O’ZBEKISTON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI FARG’ONA FILIALI

“ KOMPYUTER INJINIRINGI ” fakulteti

“DASTURIY INJINIRING” kafedrasi

“ALGORITMGA KIRISH”

fanidan

5330500-"Kompyuter injiniringi”

MARUZALAR MATNI

Farg’ona - 2015

Ushbu o’quv uslubiy ko’rsatma O’zbekiston Respublikasi Oliy va O’rta Maxsus ta’lim vazirligining Oliy O’quv yurtlari boshqarmasi 201__ yil _____da tasdiqlangan (ro’yxat t/r _____) namunaviy dasturi va “Dasturiy injiniringi” kafedrasida ishlab chiqilgan o’quv dasturi asosida tuzilgan va unga mos keladi

O’quv uslubiy ko’rsatma 5330500- Kompyuter injiniringi yo’nalishi talabalari uchun mo’ljallangan.

Tuzuvchilar: ass. Xusanov B

«Жаҳон цивилизациясига дахлдор бўлган энг замонавий илмларни эгалламай туриб, мамалакат тараққиётини таъминлаш қийин»
Ислом Каримов

Сўз боши

Инсоният тарихининг кўп асрлик тажрибаси эзгу ғоялардан ва соғлом мафкурадан маҳрум бирон-бир жамиятнинг узоққа бора олмаслигини кўрсатди. Шу боис, мустақиллик туфайли мамлакатимиз ўз олдига озод ва обод Ватан, эркин ва фаровон ҳаёт барпо этиш, ривожланган мамлакатлар қаторидан ўрин олиш, демократик жамият қуриш каби эзгу мақсадларни қўйди.

Бу эса келажагимизни яққол тасаввур этиш, жамиятимизнинг ижтимоий-маънавий пойдеворини мустаҳкамлаш эҳтиёжини туғдиради. Демак, галдаги энг асосий вазифа: ёш авлодни Ватан равнақи, юрт тинчлиги, халқ фаровонлиги каби олижаноб туйғулар руҳида тарбиялаш, юксак фазилатларга эга, эзгу ғоялар билан қуролланган комил инсонларни вояга етказиш, жаҳон андозаларига мос, кучли билимли, рақобатбардош кадрлар тайёрлашдир.

«Жаҳон цивилизациясига дахлдор бўлган энг замонавий илмларни эгалламай туриб, мамалакат тараққиётини таъминлаш қийин», - деган эдилар президентимиз И. Каримов. Ўзбекистоннинг иқтисодий ва ижтимоий соҳаларда юқори натижаларга эришиши, жаҳон иқтисодий тизимида тўлақонли шериклик ўрнини эгаллай бориши, инсон фаолиятининг барча жабҳаларида замонавий ахборот технологияларидан юқори даражада фойдаланишнинг кўламлари қандай бўлишига ҳамда бу технологиялар ижтимоий меҳнат самарадорлигини ошишида қандай рол ўйнашига боғлиқ.

Президентимиз Ислом Каримовнинг кўп йиллик изланишлари, асарларидаги фикр-мулоҳазаларига таяниб яратилган «Миллий истиқлол ғояси: асосий тушунча ва тамойиллар» номли рисолада таълим-тарбия жараёнининг ихтиёрий босқичида амал қилиш лозим бўлган қуйидаги мезон ва талаблар келтирилган:

✓ ўқув машғулотларини олиб боришда талабаларнинг ёши, тафаккури, дунёқараши ва қизиқишларини ҳисобга олиш;

✓ таълим-тарбиянинг илғор, таъсирчан воситаларидан, замонавий ўқитиш технологияси имкониятларидан кенг фойдаланиш;

✓ айрим тушунчаларни ҳаддан зиёд соддалаштириш, таълимнинг эскича услуб ва тамойилларини қўллаш натижасида фаннинг қадрсизланишига йўл қўймаслик;

✓ таълим жараёнида тазйиқ ўтказмасдан маърифий асосда иш тутиш, ёшларнинг мустақил ва эркин фикрлаш, баҳс-мунозара юритиш кўникмаларини оширишга эътибор қаратиш;

✓ ўқитувчи ва тингловчилар орасида ўзаро ҳамфикрлик ва ҳамкорлик муҳитини шакллантириш, мавзунинг тушунча ва тамойилларини шарҳлашда ҳаётий мисоллар, бугунги дунёда рўй бераётган воқеалар таҳлилидан, матбуот материалларидан кенг фойдаланиш;

✓ ёшларда ғоялар, ўз маъно-моҳиятига кўра бунёдкор ёки вайронкор бўлиши ҳақидаги ҳаётий ва ҳаққоний тасаввурларни шакллантириш;

✓ миллий истиқлол ғоясининг инсонпарварлик моҳиятини кўрсатиш асосида мустақиллик биз учун энг олий қадрият, уни асраб-авайлаш эса ҳар биримизнинг муқаддас бурчимиз эканини талабаларнинг қалби ва онгига сингдириш.

Юқорида айтилган мезон ва талабларга риоя қилган холда Республикамизда, замонавий ҳисоблаш техникаси воситаларидан самарали фойдаланишни уддалай оладиган, замонавий компьютерлардан амалий иш фаолиятида кенг фойдалана оладиган етук кадрлар тайёрлаш долзарб вазифалардан ҳисобланади. Шунинг учун, кадрлар тайёрлаш миллий дастурининг иккинчи босқичида юқори малака, рақобатбардош кадрлар тайёрлаш учун сифатли, жаҳон андозаларига мос дарсликлар, ўқув қўлланмалари ва маъруза матнларини тайёрлаб, чоп эттириш масаласига жуда катта эътибор берилган.

Мазкур маърузалар матни ҳам давлатимиз чақириғига муносиб жавоб сифатида ва узоқ йиллик педагогик фаолиятимизнинг махсули тарзида яратилди.

1–маъруза: Алгоритмлар. Ҳисоблашда алгоритмларнинг ўрни. Алгоритмлар орқали ечилувчи масалалар.

Режа:

1. Алгоритмлар;
2. Ҳисоблашда алгоритмларнинг ўрни;
3. Алгоритмлар орқали ечилувчи масалалар;

Алгоритм тушунчаси, масалаларни алгоритмлаш.

Компьютер ўз ҳисоблаш кучлилиги билан бирга тезкор, озода, аниқ ва шу билан бирга “бутунлай бефаҳм бажарувчи” ҳисобланади. Турли масалаларни ечишда ундан фойдаланганимизда компьютер бирор нимани ўзи ўйлаб топади деган фикримиз хато, компьютер ишлаши учун аниқ ва тўлиқ инструкция керак бўлади. Бу ерда биз алгоритмни аниқлаш тўпламидан бирига келяпмиз. ***АЛГОРИТМ – сўнгги натижани ҳосил қилиш учун керакли бўлган, бирор ҳаракатни амалга оширувчи қатъий ўрнатилган тартиб***. Бу ғалати туюлиши мумкин, лекин биз реал ҳаётда алгоритмга ҳар доим дуч келамиз. Омадли телефон қўнғироғи учун керакли бўлган амаллар тартибини ўз ичига олувчи телефон-автоматдан фойдаланиш инструкцияси. Маиший техникадан фойдаланиш қоидалари ва бошқалар қисқа, тушунарли шаклда бизга у ёки бу ҳолда нима қилишимиз кераклигини хабар қилиб, ҳаракатларимиз алгоритмини белгилаб беради. Тарихчи математикларнинг таъкидлашича, (H. Zemanek ишларига қаранг, Lecture Notes in Computer Sciece **122** (1981), 1-81), «алгоритм» сўзи буюк аждодимиз Абу Абдуллоҳ Муҳаммад ибн Мусо ал-Хоразмий исмидан келиб чиққан, унинг машҳур “Китоб ал-жабр ва ал-муқобола” трактаси эса яна бир машҳур “алгебра” атамасининг вужудга келишига асос бўлди. Компьютер иши жараёнида бошқариладиган инструкцияларни ишлаб чиқаришнинг асоси алгоритм ҳисобланади. Бироқ, биз алгоритмдан ўз ёзувларимизни тўғридан-тўғри компьютерга ўтказа олмаймиз, чунки улар компьютер тушунмайдиган, фақатгина инсонлар тушунадиган тилда ёзилган. Компьютер алгоритмни тушуниши учун у машина тилига ўгирилади, айнан шундай машина тилида ёзилган алгоритмлар *дастур* ёки *компьютер дастури* деб аталади. Қуйида биз бу тушунчани жорий курс асосида ётувчи алгоритм тушунчаси ёрдамида аниқлаштиришга ҳаракат қиламиз. Шуни таъкидлаш керакки, адабиётда умумэътироф этилган алгоритмни аниқлаш тушунчаси йўқ. *Компьютер технологиялари* тушунчасига адекват бўлган алгоритм ифодасини берамиз:

<u>Алгоритм</u> – бу масала ечимини ҳосил қилиш учун бошлангич информацияда амалга ошириш керак бўлган аниқ белгиланган амаллар кетма-кетлиги.

Ихтиёрий **алгоритм** муҳим хоссаларга эга:

- Алгоритмнинг *аниқлиги* – ҳар бир қадам бажарилишининг бир қийматлилиги.
- *Дискретлилиги* – масалани ечиш жараёнини бажарилиш вақтида компьютер ёки инсонга қийинчилик туғдирмаслиги учун бир неча содда босқичлар (бажарилиш қадамлари)га бўлиш.
- *Оммавийлик* – белгиланган масалалар синфини ечиш учун алгоритмнинг фойдалилиги.
- *Натижавийлик* – охирги қадамларда дастлабки маълумотларга эга бўлган керакли натижани олишга имкон берувчи алгоритмнинг ҳаракатлар якуни.

Амалиётда қуйидаги ***алгоритм*** турлари мавжуд***:***

<u>Чизиқли</u> – амаллар кетма-кет, бирор-бир шарт текширилмасдан бажарилувчи алгоритм.

<u>Тармоқланувчи</u> – белгиланган шартларнинг ўзгаришига боғлиқ ҳолда кўрсатмаларнинг вариантлари олдиндан мўлжалланадиган алгоритм.

<u>Циклик</u> – алоҳида жараёнлар ёки жараёнлар гуруҳи бир неча марта бажариладиган алгоритм.

Алгоритмни ёзиш усуллари: *сўзли, формулали, жадвалли, график.*

Информацион модель тушунчаси

Информация компьютерда қайта ишлаш учун хом-ашё сифатида хизмат қилади, яъни металлургик ишлаб чиқаришда металл руда хом-ашё ҳисобланиши каби. Лекин, ихтиёрий хом-ашё қайта ишловда самарали бўлиши учун дастлабки тайёргарликка эга бўлиши керак. Бу тўлалигича информацияга тегишли. Демак, бизда ўрганиш учун ҳисоблаш техникасига жалб этмоқчи бўлган бирор бир ҳодиса мавжуд. Биринчи навбатда биз қизиқтираётган ҳодиса ҳақидаги информацияларни йиғамиз, сўнг бу информацияни системалаштирамиз ва синфларга ажратамиз. Бундан кейин берилган ҳодисани ифодаловчи моделни қурамиз. **Модель** *сўзи француzчадан келиб чиққан бўлиб, дастлабки маъноси бу – бирор физик обьект ёки ҳодисанинг аниқ кўринишини берувчи намунадир.* Биз физик эмас, балки **информацион модел**дан фойдаланамиз. У ҳодисани махсус математик аппарат, график, диаграммалар ёрдамида ифодалайди. Модель ҳодисанинг характерли белгилари ва асосий томонларини очиб бериш имконини беради. Математик ва имитацион моделлаштириш мавжуд.

Математик моделлаштириш – ҳодиса тадқиқоти ва ифодаси учун математик аппаратни қўллаш. Аниқ математик модель объектнинг ҳолатини кузатиш ва уни таҳлил қилиш имконини беради.

Имитацион моделлаштириш – асосан саноатда қўлланилади, ҳисоблаш техникаси ва махсус программа таъминоти ёрдамида реал мавжуд бўлмаган қурилмада бир қатор текширишлар ўтказиш имконини беради. Бундай моделлаштиришни қўллаш хом-ашё ишлаб чиқаришни тезлаштиради, чунки қуриш ва тадқиқ қилиш жараёни қисқаради, хатолар миқдори ва уларнинг баҳоси камаяди. Масалан, «Боинг» фирмаси узоқ йиллар давомида қўлланиб келинган жойни режалаштиришни ишлаб чиқиш, йўловчилар ўриндиқларининг жойлашуви, самолёт салонининг натурал моделларини яратишдан бош тортди, уларни компьютер моделларига алмашди. Бу миллионлаб долларларнинг тежалиши ва самолётларнинг янги моделларини ишлаб чиқиш муддатларини қисқартирди.

Моделни қуриб бўлгандан сўнг унга мос *алгоритм*ни яратиш босқичига ўтилади.

Алгоритмлар орқали ечилувчи масалалар.

Ҳисоблаш машинаси тилида (машина кодларида) кетма-кет буйруқлар кўринишида берилган масала ечимининг алгоритми машинавий дастур деб аталади.

Машинавий дастур буйруғи ёки машинавий буйруқ–қўшимча кўрсатма ва тушунчаларсиз автоматик ҳолда бажарилувчи элементар машина инструкцияси.

Дастурлаш – дастур тузиш билан боғлиқ бўлган назарий ва амалий фаолият.

Алгоритмни машина тилига ўгириш жараёни **трансляция** деб аталади. Машина тилини «одамийлаштиришнинг» биринчи қадами рамзий номларни машина кодига ўтказувчи дастурлар тузиш бўлди. Сўнг арифметик ифодаларни ўгирувчи дастурлар яратилди ва ниҳоят, 1958 йилда дастурлаш тилида кенг фойдаланиладиган Фортран транслятори кириб келди. Шундан сўнг кўплаб дастурлаш тиллари яратилди.

Компьютер машинавий дастур буйруқларини бошқарган ҳолда информацияга ишлов беради, бунинг учун иш жараёнида турли берилганлардан фойдаланади.

Фойдаланилган берилганлар қуйидагиларга бўлинади:

1. *Кирувчи* – компьютерга киради ва масалани ечиш учун шарт сифатида фойдаланилади.

2. *Жорий ёки ички* – дастур ичида информацияни сақлаш ва ишлов бериш учун ишлатилади.

3. *Чиқувчи* – информацияга ишлов бериш натижасида дастурда ҳосил бўлган берилганлар. Матн, график, видеотасвир ва ҳ. к. кўринишда бўлиши мумкин.

Ҳодиса тадқиқоти, масала ечими учун ҳисоблаш техникаси ёрдамида қабул қилиш керак бўлган амалларнинг умумий тартибини қуйидагича схема сифатида тасвирлаш мумкин:

Ҳодиса, жараён, масала ➔ модель➔алгоритм➔дастур➔компьютер➔ натижа.

Назорат саволлари

1. Алгоритмлар турларини айтинг.
2. Ҳисоблашда алгоритмларнинг ўрни.
3. Алгоритмлар орқали ечилувчи масалалар.

2–маъруза: Қўшилган саралаш. Қўшилган саралаш корректлиги. Псевдокод тузишда келишув. Алгоритм таҳлили.

Режа:

1. Қўшилган саралаш;
2. Қўшилган саралаш корректлиги;
3. Псевдокод тузишда келишув;
4. Алгоритм таҳлили;

Қўшимчалар билан саралаш

Қўшимчалар билан саралашнинг асосий ғояси шундаки, янги элементни сараланган рўйхатга қўшишда уни ихтиёрий жойга эмас, балки керакли жойга жойлаштириш лозим, сўнг бутун рўйхатни бошқатдан саралаш керак. Қўшимчалар билан саралашда ихтиёрий рўйхатнинг биринчи элементи сараланган рўйхатнинг 1 узунлиги деб ҳисобланади. 2 элементли сараланган рўйхат бошланғич рўйхатдаги 2-элементни 1-элемент жойлашган рўйхатнинг керакли ўрнига жойлаштириш орқали яратилади. Энди бошланғич рўйхатнинг 3-элементини сараланган 2 элементли рўйхатга қўйиш мумкин. Бу жараён бошланғич рўйхатнинг барча элементлари кенгаювчи сараланган рўйхатга жойлашгунга қадар давом этади.

Бу жараённи ифодаловчи алгоритм қуйида келтирилган:

```
InsertionSort(list,N)
List N
For i=2 to N do
        newElement=list[i]
        location=i-1
        while(location>=1) and (list[location]>newElement) do
                list[location+1]=list[location]
                location=location-1
        end while
        list[location+1]=newElement
end for
```

Бу алгоритм янги қўйиладиган элементни newElement ўзгарувчисига юклайди. Сўнг барча катта элементларни 1 позицияга суриб, бу элементга жой ажратади. Циклнинг охирги итерацияси элементни location+1 номер билан location+2 позиция ўтказади.

Энг ёмон ҳолат таҳлили

Агар ички while циклини кўриб чиқадиган бўлсак, яна бошқатдан қўшилаётган элемент барча элементларда кичик ва рўйхатнинг сараланган қисмида жойлашган бўлса, жараённинг катта қисми бажарилади. Бу ҳолатда location ўзгарувчисининг қиймати 0 га тенг бўлганда цикл тўхтайди. Шунинг учун алгоритм бажарилишининг асосий қисми ҳар бир янги элемент рўйхат бошига қўшилганда амалга оширилади. Бу фақатгина бошланғич рўйхат элементлари камайиш тартибда бўлганда бажарилади. Бу энг ёмон ҳодисалардан бири, лекин бошқалари ҳам бор.

Бундай рўйхатни қайта ишлашни қандай амалга ошириш кераклигини кўриб чиқамиз. Дастлаб рўйхатнинг 2-елементи қўйилади. У фақатгина битта элемент билан таққосланади. Иккинчи қўйилган элемент олдинги иккита элемент билан, учинчи қўйиладиган элемент учта элемент билан ва ҳоказо, i-қўйиладиган элемент i та элемент билан таққосланади ҳамда бу жараён N-1 марта такрорланади. Шундай қилиб, энг ёмон ҳолатдаги саралаш мураккаблиги қуйидагига тенг:

$$W(N) = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \frac{N^2 - N}{2}$$

$$W(N) = O(N^2)$$

Ўрта ҳолат таҳлили

Ўрта ҳол таҳлилини 2 босқичга ажратамиз. Дастлаб навбатдаги элемент ҳолатини аниқлаш учун керак бўладиган таққослашларнинг ўрта қийматини ҳисоблаймиз. Кейин, 1-қадам натижасидан фойдаланиб, барча керакли амалларнинг ўрта қийматини ҳисоблаш мумкин. Дастлаб, і-елементнинг жойлашиш ўрнини аниқлаш учун таққослашларнинг ўрта қийматини ҳисоблаймиз. Юқорида айтиб ўтганимиздек, рўйхатга і-елементни қўшганда у керакли жойда жойлашган бўлса ҳам, ҳеч бўлмаганда битта таққослаш бажарилади.

і-елемент учун нечта мумкин бўлган ҳолат мавжуд? Қисқа рўйхатларни кўриб чиқамиз ва ихтиёрий ҳолатда натижаларни умумлаштиришга ҳаракат қиламиз. 1-қўшиладиган элемент учун 2 та имконият бор: икки элементдан биринчиси ёки иккинчиси бўлиши мумкин. 2-қўшиладиган элементнинг 3 ҳолатдан бирида бўлиши мумкин: 1,2,3 рақамлари билан. Демак, і-қўшиладиган элемент і+1 та ҳолатдан бирини эгаллаши мумкин. Барча имкониятлар эҳтимоллик даражасини тенг деб оламиз.

Ҳар бир і+1 позицияга етиб олиш учун нечта таққослаш талаб этилади? і нинг кичик қийматларини кўриб чиқамиз ва натижани бирлаштиришга ҳаракат қиламиз. Агар 4-қўшилаётган элемент 5-позицияга тушса, у ҳолда таққослаш ёлғон бўлади. Агар унинг 4-позицияси тўғри бўлса, у ҳолда 1-таққослаш рост ҳисобланади, иккинчиси эса – ёлғон. 3-позицсияга тушса, 1- ва 2-таққослаш рост, 3-си эса ёлғон бўлади. 2-позицияда 1-,2- ва 3-таққослаш рост ва 4-си ёлғон бўлади. 1-позицияда барча таққослашлар рост бўлади, ёлғон таққослашлар бўлмайди. Бундан эса і+1, і, і-1,…, 2 позицияларга тушувчи и-елемент учун таққослашлар мос ҳолда 1,2,3 …і эканлиги келиб чиқади. 1-позицияга тушганда эса таққослашлар сони і га тенг бўлади. і-қўйиладиган элемент учун таққослашларнинг ўрта қиймати қуйидаги тенглик билан берилади:

$$A_i = \frac{1}{i+1}\left(\sum_{p=1}^{i} p + i\right) = \frac{1}{i+1}\left(\frac{i(i+1)}{2} + i\right) = \frac{i}{2} + \frac{i}{i+1} = \frac{i}{2} + 1 - \frac{1}{i+1}$$

Биз і-елементни қўйишдаги таққослашларнинг ўрта қийматини ҳисобладик. Энди бу натижани рўйхатдаги ҳар бир N-1 элемент учун йиғиш керак:

$$A(N)=\sum_{i=1}^{N-1}A_i=\sum_{i=1}^{N-1}(\frac{i}{2}+1-\frac{1}{i+1})$$

$$A(N)=\sum_{i=1}^{N-1}\frac{i}{2}+\sum_{i=1}^{N-1}1-\sum_{i=1}^{N-1}\frac{1}{i+1}$$

$$\sum_{i=1}^{N-1}\frac{i}{2}=\sum_{i=1}^{N-1}1=\sum_{i=1}^{N-1}\frac{1}{i+1}$$

$$A(N)\approx\frac{1}{2}\frac{(N-1)N}{2}+(N-1)-(\ln N-1)=\frac{N^2-N}{4}+(N-1)-(\ln N-1)$$

$$=\frac{N^2+3N-4}{4}-(\ln N-1)\approx\frac{N^2}{4};$$

$$A(N)=O(N^2).$$

3–маъруза: Функциянинг ўсиши. Асимптотик белгилашлар. Тенгламалар ва тенгсизликларда асимптотик белгилашлар.

Режа:

1. Функциянинг ўсиши;
2. Асимптотик белгилашлар;
3. Тенгламалар ва тенгсизликларда асимптотик белгилашлар;

Алгоритм таҳлилини, қўйилган масалани ушбу алгоритм билан ечиш қанча вақт талаб қилиши деб тасаввур қилиш мумкин.

Ҳар бир қаралаётган алгоримтни N ўлчовли бошланғич маълумотлар массивидаги масалаланинг қанчалик тез ечилиши билан баҳолаймиз.

Масалан, саралаш алгоритми N та қийматдан иборат рўйхатни ўсиш тартибида жойлаштириш учун қанча таққослаш талаб қилади ёки N*N ўлчамли иккита матрицани кўпайтиришда қанча арифметик амаллар зарурлигини ҳисоблаш.

Битта масалани турли алгоритмлар билан ечиш мумкин. Алгоритмлар таҳлили бизга алгоритмни танлаш учун курол бўлади. Тўртта қийматдан энг каттасини танлайдиган иккита алгоритмни қараймиз:

largest = a
if b > largest then
largest = b
end if
return a
if c > largest then
largest = c end if if d > largest then

largest = d end if return largest
if a > b then if a > c then I f a > d then
return a else
return d end if else
if c > d then
return c else
return d end if end if else
if b > c then if b > d then
return b else
return d end if else
if c > d then
return c else
return d end if end if end if

Кўриниб турибдики, қаралаётган алгоритмларнинг ҳар бирида учта таққослаш бажарилади. Биринчи алгоритмни ўқиш ва тушуниш осон, аммо компьютерда бажарилиш нуқтаи назаридан уларнинг мураккаблик даражалари тенг. Бу икки алгоритм вақт нуқтаи назаридан тенг, лекин биринчи алгоритм largest номли қўшимча ўзгарувчи ҳисобига кўпроқ хотира талаб қилади. Агарда сон ёки белгилар таққосланса, ушбу қўшимча ўзгарувчи катта аҳамиятга эга бўлмайди, лекин бошқа турдаги маълумотлар билан ишлаганда бу муҳим аҳамиятга эга. Кўплаб замонавий дастурлаш тиллари катта ва мураккаб объектларни ёки ёзувларни таққослаш операторларини аниқлаш имконини беради. Бундай ҳолларда қўшимча ўзгарувчиларни жойлаштириш катта жой талаб қилади. Алгоритмларнинг эффективлигини таҳлили қилишда бизни биринчи навбатда вақт масаласи қизиқтиради, аммо хотира муҳим роль ўйнайдиган вазиятда уни ҳам муҳокама қиламиз.

Алгоритмларинг турли хоссалари битта масалани ечувчи икки турдаги алгоритмларнинг эффективлигини таққослаш учун хизмат қилади.
Биз шунинг учун ҳеч қачон матрицаларни кўпайтириш алгоритми билан саралаш алгоритмини эмас, балки иккита турли саралаш алгоритмларини бир-бири билан таққослаймиз.
Алгоритм таҳлилининг натижаси – белгиланган алгоритмнинг компьютердан қанча вақт ёки такрорлаш талаб қилишини аниқ ҳисобловчи формула эмас.

Бундай маълумот муҳим эмас, бу ҳолатда компьютер тури, у битта ёки ундан ортиқ фойдаланувчи томонидан ишлатиляптими, унинг процессори ва частотаси қанақа, процессор чипида командалар тўлиқми ва компилятор

бажарилаётган кодни қай даражада амалга оширмоқда каби томонларни назарда тутиш керак.

Бу шартлар алгоритм бажарилиш натижасида дастурнинг ишлаш тезлигига таъсир қилади. Юқоридаги шартлар ҳисобига дастурни бошқа тез ишлайдиган компьютерга ўтказилганда алгоритм яхши ишлагандай бажарилиши тезроқ амалга ошади. Аслида эса ундай эмас, биз шунинг учун таҳлилимизда компьютернинг имкониятларини инобатга олмаймиз.

Оддий ва катта бўлмаган дастурларда бажариладиган амаллар сонини N нинг функцияси кўринишида аниқ ҳисоблаш мумкин.

Аксарият ҳолатларда бунга зарурият қолмайди. 8 .4 § да келтирилган $N + 5$ та ва $N + 250$ та амал бажариладиган икки алгоритм орасида N нинг етарлича катта қийматларида деярли фарқ бўлмайди. Шунга қарамай, биз алгоритмларни бажариладиган амаллар сонига қараб таҳлил қиламиз.

Алгоритм томонидан бажариладиган жараёнлар борки, биз уларнинг ҳаммасини ҳисоблаб ўтирмаймиз, бунинг сабаби шундаки, ҳатто унинг энг кичик созлаши ҳам самарадорликнинг сезилмас яхшиланишига олиб келади. Файлдаги турли белгилар сонини ҳисобловчи алгоритмни қараймиз. Бу масала ечими учун алгоритмнинг тахминий кўриниши қуйидагича бўлади:

for all 256 белгиларни do
ҳисоблагични нолга тенглаш end for
while агар файлда белги қолса do
навбатдаги белгини кўрсат ва ҳисоблагични биттага ошир end while
do
ҳисоблагични нолга тенглаштириш end for
while файлда белги мавжуд бўлса do
навбатдаги белгини кўрсат ва ҳисоблагични биттага ошир
end while

Ушбу алгоритмни кўриб чиқамиз. У такрорланиш бажарилишида 256 та ўтиш қилади. Агар берилган файлда N та белги бўлса унда иккинчи такрорланишда N та ўтиш қилинади. «Бу қандай ҳисоблаш?» деган савол туғилади. **For** циклида аввал цикл ўзгарувчиси бажарилади, кейин хар бир ўтишда унинг цикл чегарасидан чиқмаётганлиги текширилади ва ўзгарувчи қийматини оширади. Бу эса цикл бажарилишида 257 юклаш бажарилади (бири цикл ўзгарувчиси, 256 таси ҳисоблагич учун), яъни 256 та ошириш ва 257 та цикл чегарасидан чиқмаганлигини текшириш (битта амал циклни тўхтатиш учун қўшилган). Иккинчи циклда $N + 1$ марта шарт текширилади (+1 файл бўш бўлгандаги охирги текширув), ва N ҳисоблагични ошириш. Жами амаллар:

Ошириш $N + 256$

Юклаш	*257*
Шартларни текшириш	*N + 258*

Шундай қилиб 500 белгидан иборат файл берилса алгоритмда 1771 та амал бажарилади, улардан 770 таси натижа беради (43%). Энди N нинг қиймати ошганда нима бўлишини кўрамиз. Агар файл 50 000 белгидан иборат бўлса, унда алгоритм 100 771 амал бажаради, уларнинг 770 таси натижа учун (жами амаллар сонининг 1% ини ташкил этади). Ечимга қаратилган амаллар сони ошмаяпти, лекин *N* катта бўлганда уларнинг фоизи жуда кам.

Энди бошқа томонига эътибор қаратамиз. Компьютерда маълумотлар билан шундай ишлашга мўлжалланганки, катта хажмдаги маълумотлар блокини кўчириш ва юклаш бир хил тезликда амалга оширилади. Шунинг учун биз аввал 16 та ҳисоблагичга бошланғич қиймат 0 ни юклаймиз, кейин қолган ҳисоблагичларни тўлдириш учун шу блокдан 15 та нусха оламиз.Бу эса цикл бажарилиш давомида текширишлар сонини 33 га, юклашлар сонини 33 ва оширишлар сонини 31 га камайишига олиб келади. Демак амал бажарилишлар сони 770 дан 97 гача камайди, яъни 87%. Агар эришилган натижани 50000 белгидан иборат файл устида бажарсак, тежамкорлик 0.7% ни ташкил қилади (100771 та амал ўрнига 100098 амал бажарамиз).

Агарда барча амалларни циклдан фойдаланмай 31 та юклашлар орқали бажарганимизда, вақтни янада тежаган бўлардик, аммо бу усул 0.07 фойда келтиради. Ишимиз унумли бўлмайди.

Кўриб турганимиздек, алгоритмнинг бажарилиш вақти билан боғлиқ барча амаллар бефойда. Таҳлил тили билан айтганда, бошланғич маълумотлар ҳажмининг ортишига алохида эътибор қаратиш керак.

Аввалги ишларда алгоритмларни таҳлил қилишда алгоритмларни Тьюринг машинасида ҳисоблаш аниқланган. Таҳлилда масалани ечиш учун зарур бўлган ўтишлар сони ҳисобланган. Бу турдаги таҳлил тўғри бўлиб, икки алгоритмнинг нисбий тезликларини аниқлаш имконини беради, аммо унинг амалиётда қўлланилиши кам, чунки кўп вақт талаб қилади.

Аввал бажариладиган алгоритмнинг Тьюринг машинасидаги ўтиш функцияларини ёзиш, кейин эса бажарилиш вақти ҳисобланади.

Алгоритмларни таҳлил қилишнинг бошқа яхшироқ усули - уни бирор юқори босқичли тил Pascal, C, C++, JAVA да ёзиш ёки оддий псевдокодларда ёзишдир. Барча алгоритмларнинг асосий бошқарув структурасини ифодалаганда псевдокодларнинг хоссалари аҳамиятга эга эмас. Ихтиёрий тил бизнинг талабимизга жавоб беради, чунки **for ёки while** шаклидаги цикллар, **if, case ёки switch** кўринишидаги тармоқланиш механизмлари барча

дастурлаш тилларида мавжуд. Ҳар гал биз битта аниқ алгоритмни кўриб чиқишимизга тўғри келади – унда бирдан ортиқ функция ёки программа фрагменти киритилган бўлади, шунинг учун юқорида келтирилган тилларнинг тезлиги умуман муҳим эмас. Псевдокодлардан фойдаланишимизнинг сабаби шунда.

Кўплаб дастурлаш тилларида мантиқий ифоданинг қийматлари қисқартирилган шаклда ҳисобланади. Бу A and B ифодадаги B ҳаднинг қиймати қачонки A рост бўлсагина ҳисобланади, акс ҳолда натижа B га боғлиқ бўлмаган тарзда ёлғон бўлади. Худди шундай A or B ифодада A нинг қиймати рост бўлса, B ҳаднинг қиймати ҳисобланмайди. Кўриниб турибдики, мураккаб шартларнинг 1 ёки 2 га тенглигидаги таққослашларининг сонини ҳисоблаш шарт эмас. Шунинг учун бу бобни ўрганишда мантиқий ифодаларнинг қийматини қисқартириб ҳисобланишини эътиборсиз қолдирамиз.

Кирувчи берилганлар синфи

Алгоритмларнинг таҳлилида кирувчи маълумотларнинг роли юқори, чунки алгоритм ҳаракатларининг кетма-кетлиги кирувчи маълумотлар билан белгиланади. Масалан, *N* та элементдан ташкил топган рўйхатнинг энг катта элементини топиш учун қуйидаги алгоритмдан фойдаланиш мумкин:

```
largest = list [l]
for i = 2 to N do
if (list [i] > largest) then
 largest = list[i]
end if
end for
```

Агар рўйхат камайиш тартибида бўлса, у ҳолда цикл бошланишидан аввал битта ўзлаштириш бажарилади, цикл танасида эса ўзлаштириш бўлмайди. Агар рўйхат ўсиш тартибида бўлса, у ҳолда *N* та ўзлаштириш бажарилади (цикл бошланишидан аввал битта ва *N*-1 та циклда). Биз таҳлил қилиш давомида кирувчи қийматлар тўпламининг турли имкониятларини кўриб чиқишимиз керак, агар битта тўплам билан чегаралансак, бу ечим энг тез (ёки энг секин) бўлган тўплам бўлиб чиқиши мумкин. Натижада биз алгоритм ҳақидаги ёлғон тасаввурга эга бўламиз. Бунинг ўрнига кирувчи тўпламлар туринининг барчасини кўриб чиқамиз.

Биз кирувчи тўпламларни ҳар бир тўпламдаги алгоритм ҳолатига боғлиқ ҳолда синфларга бўлиб чиқамиз. Бундай бўлиниш кўриб чиқилаётган тўпламлар миқдорини камайтириш имконини беради. Масалан, 10 та сондан иборат рўйхат учун энг катта элементни топиш алгоритмини қўллаймиз.

Биринчи сони энг катта бўлган 362 880 кирувчи тўпламлар мавжуд, уларни битта синфга жойлаштириш мумкин. Агар қиймати бўйича энг катта сон иккинчи ўринда турган бўлса, у ҳолда алгоритм иккита ўзлаштиришни амалга оширади. Энг ката сон иккинчи ўринда турган тўплам 362 880. Уларни бошқа синфга киритиш мумкин. 1 дан N гача бўлган сонлар орасида энг катта соннинг ўзгариш ҳолида ўзлаштиришлар сонининг қандай ўзгаришини кўришимиз мумкин.

Шундай қилиб, барча кирувчи тўпламларни бажарилган ўзлаштиришлар сони бўйича N та турли синфга бўлиш керак. Кўриб турганингиздек, ҳар бир синфда жойлашган тўпламларни бирма-бир ёзиш ёки ёзиб олиш шарт эмас. Фақатгина ҳар бир тўпламдаги синфлар миқдори ва иш ҳажмини билиш етарли.

Кирувчи берилганларнинг мумкин бўлган тўплами N катталашганда жудаям катта бўлиши мумкин. Масалан, 10 та турли сони рўйхатда 3 628 800 усулда жойлаштириш мумкин. Бу усулларнинг барчасини кўриб чиқишнинг имкони йўқ. Биз бунинг ўрнига алгоритм бажарилишига кўра рўйхатни синфларга бўламиз. Юқорида кўрсатилган алгоритм учун бўлиниш энг катта қийматнинг жойлашиши ўрнига асосланади. Натижада 10 та турли синф ҳосил бўлади. Бошқа алгоритм учун, масалан, энг катта ва энг кичик сонни топиш алгоритмида бўлиниш энг катта ва энг кичик соннинг жойлашувига асосланади. Бундай бўлинишда 90 та синф бўлади. Синфларни ажратиб бўлгач, ҳар бир синфдан битта тўпламда алгоритм ҳолатини кўриш мумкин. Агар синфлар тўғри танланган бўлса, у ҳолда бир синфдаги кирувчи берилганлар тўпламида алгоритм бир хил миқдордаги амалларни бажаради, бошқа синфнинг тўпламлари учун эса амаллар миқдори бошқача бўлади.

Хотира бўйича мураккаблик

Биз асосан алгоритмларнинг вақт бўйича мураккаблигини муҳокама қиламиз, аммо иш бажариш учун у ёки бу алгоритмга қанча хотира кераклиги ҳақида ҳам айтиш мумкин. Компьютер хотираси (ҳам ички, ҳам ташқи) ҳажми чегараланган. Компьютерлар ривожланишининг дастлабки босқичларида бу таҳлил услубий характерга эга эди. Барча алгоритмлар чегараланган хотира етарли ёки қўшимча майдонни талаб қилувчи алгоритмларга бўлинади. Кўпинча дастурловчилар хотирасига эга ва ташқи қурилмалар талаб қилмайдиган секин ишловчи алгоритмни танлашар эди.

Компьютер хотирасига бўлган талаб жуда катта эди, шунинг учун қайси маълумотлар сақланиб қолади, бундай сақлашнинг самарали усуллари қандай каби саволлар ўрганилар эди. Фараз қилайлик, масалан, биз -10 дан +10 гача интервалдаги вергулдан кейин битта ўнли белгига эга бўлган

моддий сон ёзяпмиз. Моддий сони ёзишда кўпчилик компьютерлар 4 дан 8 байтгача хотира сарфлайди, леки нагар бу сони аввалдан 10 га кўпайтирсак, -100 дан +100 гача интервалдаги бутун сон ҳосил қиламиз ва уни сақлаш учун бор йўғи бир байт сарфланади. Биринчи вариант билан солиштирсак, 3-7 байт тежашга эришилди. 1000 та шундай сон сақлайдиган дастур 3000 дан 7000 байтгача тежайди. Агар ўтган асрнинг 80-йилларида компьютерларнинг хотираси 65536 байт бўлганлигини эътиборга олсак, жиддий тежаш кўзга ташланади. Айнан шу компьютер дастурларининг узоқ йил ишлаши хотирани тежаш зарурияти Билан бир қаторда 2000 йил муаммо туғдирди. Агар сизнинг дастурингиз турли саналардан фойдаланса, йилни ёзиш учун 1999 ўрнига 99 ифодасини сақлаган ҳолда жойнинг ярмини тежаса бўлади. 80-йиллардаги дастур муаллифлари маҳсулотлари 2000 йилгача яшашини тахмин ҳам қилишмаган эди.

Ҳозирги кунда бозорларда таклиф қилинаётган дастурий таъминотга назар ташласак, хотиранинг бундай таҳлили ўтказилмаганлиги аён бўлади. Оддий дастурлар учун зарур хотира ҳажми мегабайтларда ўлчанади. Дастур тузувчилар жойни тежаш эҳтиёжини ҳис қилмаётганга ўхшайдилар, уларнинг фикрича, агар фойдаланувчида етарли хотира бўлмаса, у дастур бажарилиши учун етмаётган 32 ёки ундан ортиқ мегабайт хотира ёки уни сақлаш учун янги қаттиқ диск сотиб олади. Натижада компьютерлар ўзининг белгиланган муддатидан аввал яроқсиз ҳолга келиб қолади.

Яқинда тарқалган чўнтак компьютерлари (PDA – personal digital assistant) янги оҳанг олиб кирди. Бундай қурилманинг хотираси ҳам маълумотлар, ҳам дастурлар учун 2 дан 8 мегабайтгача. Шунинг учун ҳам маълумотларни ихчам сақлашни таъминловчи кичик дастурларни яратиш қийин бўлиб қолмоқда.

Нимани ҳисоблаш ва нимани инобатга олиш лозим

Нимани ҳисоблаш масаласи иккита қадамдан иборат. Биринчи қадамда аҳамиятли жараён ёки жараёнлар гуруҳи танланади, иккинчи қадамда шу жараёнлардан қай бири алгоритмда жойлашган, қайсилари эса қўшимча ҳаражатларни ёки маълумотларни қайд эти шва ҳисобга олишга кетиши ташкил этади. Икки хил аҳамиятли жараён тури мавжуд: таққослаш ва арифметик амаллар. Барча таққослаш операторлари эквивалент ҳисобланади ва уларни излаш ҳамда ажратув алгоритмларида инобатга олинади. Таққослаш миқдори бундай алгоритмларнинг муҳим элементи ҳисобланади, излашда ушбу миқдор изланган катталик билан мос тушадими, саралашда эса берилган ораликдан чиқиши аниқланади. Солиштирувчи операторлар бир катталикнинг иккинчиси билан тенг ёки тенг эмаслиги, катта ёки кичиклиги, кичик ёки тенглиги, катта ёки тенглигини текширади.

Биз арифметик амалларни икки гуруҳга бўламиз: аддитив ва мультипликатив. Аддитив операторлар (қисқа қилиб айтганда қўшув) қўшиш, айириш, орттириш ва қисқартиришни ўз ичига олади. Мультипликатив операторлар (ёки қисқача кўпайтиришлар) кўпайтириш, бўлиш ва модул бўйича қолдиқ олишни ўз ичига олади. Икки гуруҳга ажратиш кўпайтиришнинг қўшишдан кўпроқ ишлатилишига боғлиқ. Амалиётда баъзи алгоритмлар уларда кўпайтириш кам бўлса, қўшишлар сони пропорционал даражада ўсса ҳам афзалроқ ҳисобланади. Биз ўз китобимизда яна бир, кўпайтиришдан ҳам кўп вақт талаб қилувчи амаллар гуруҳини ҳосил қилувчи логарифмлар ва тригонаметрик функциялардан фойдаланувчи алгоритмларга тўхталмадик (одатда компьютерлар бу ифодаларни бир қаторга ажратиш ёрдамида ҳисоблайдилар). Бутун сонли икки даражага кўпайтириш ёки бўлиш алоҳида ҳолатни ташкил қилади. Бу жараён силжишга олиб келади, кейингиси эса тезлик жиҳатдан қўшишнинг эквиваленти ҳисобланади. Бироқ, бу тафовут сезиларли бўлган ҳоллар жуда кам, чунки 2 га кўпайтириш ва бўлиш биринчи навбатда таққослаш операторлари аҳамиятга эга бўлган «тақсимла ва бошқар» сингари алгоритмларда учрайди.

Кирувчи маълумотларнинг синфлари

Алгоритмни таҳлил қилишда кирувчи маълумотларни танлаш унинг бажарилишига таъсир қилиши мумкин. Айтайлик, баъзи саралаш алгоритмлари, агар кириш рўйхати сараланган бўлса, жуда тез ишлаши мумкин, бошқа алгоритмлар шундай рўйхатда унча катта бўлмаган натижани кўрсатади. Тасодифий рўйхатда эса натижа бунинг тескариси бўлиши мумкин. Шунинг учун биз маълумотларнинг бир кириш рўйхатидаги алгоритмлар ҳаракатини таҳлил қилиш билан чегараланмаймиз. Биз алгоритмни ҳам энг тез, ҳам энг секин ишлашини таъминловчи маълумотларни қидирамиз. Бундан ташқари, биз барча мавжуд маълумотлар тўпламидаги алгоритмларнинг ўртача самарасини ҳам баҳолаймиз.

Энг яхши ҳолат

Бўлимнинг номланишидан ҳам кўриниб турибдики, алгоритмлар учун энг яхши ҳолат бу қисқа вақт ичида амалга ошириладиган алгоритмнинг маълумотлар жамланмаси. Бундай жамланма алгоритм энг амал бажарадиган қийматлар комбинациясини ифодалайди. Агар биз излаш алгоритмини текширсак, изланган қиймат биринчи алгоритм текшираётган катакка ёзилган бўлса (одатда мақсадли қиймат ёки калит деб аталади), маълумотлар тўплами энг яхши ҳисобланади. Бундай алгоритмга унинг мураккаблигидан қатъий назар, битта таққослаш керак бўлади. Шуни эслатиш керакки, рўйхатдан излашда, унинг қанчалик узун бўлишидан қатъий назар, энг яхши ҳолат

доимий вақтни талаб қилади. Умуман, энг яхши ҳолатда алгоритмни бажариш вақти кичик ёки доимий бўлади, шунинг учун биз бундай таҳлилни кам ўтказамиз.

Энг ёмон ҳолат

Энг ёмон ҳолатни таҳлил қилиш жуда муҳим, чунки у алгоритм ишининг максимал вақтини тасаввур қилишга ёрдам беради. Энг ёмон ҳолатни таҳлил қилганда алгоритм энг кўп иш бажарадиган кириш маълумотларини топиш зарур. Изловчи алгоритм учун бу каби кирувчи маълумотлар – бу шундай рўйхатки, унда изланган калит охирида келади ёки умуман бўлмайди. Натижада N таққослаш керак бўлади. Энг ёмон ҳолатнинг таҳлили танланган алгоритмга қараб дастурнинг ишлаш вақти учун юқори баҳони беради.

Ўртача ҳолат

Ўрта ҳолатнинг таҳлили энг мураккаб ҳисобланади, чунки у кўпгина деталларни ҳисобга олишни талаб қилади. Таҳлил асосини мавжуд бўлган кирувчи маълумотлар тўпламини бўлиб чиқиш лозим бўлган турли гуруҳларни аниқлаш ташкил қилади. Иккинчи қадамда кирувчи маълумотлар тўплами қайси гуруҳга тегишли бўлиш эҳтимоли аниқланади. Учинчи қадамда ҳар бир гуруҳдаги маълумотларга алгоритмнинг иш вақти ҳисобланади. Алгоритмнинг бир гуруҳдаги ҳамма кирувчи маълумотлар учун ишлаш вақти бир хил бўлиши керак, акс ҳолда гуруҳни бўлиш лозим. Ўртача иш вақти

$$A(n)\sum_{i=1}^{m} p_i t_i, \quad (7.1)$$

формула орқали ҳисобланади. Бу ерда n - кирувчи маълумотлар ўлчами, m - гуруҳлар сони, p_i - кирувчи маълумотларнинг i сонли гуруҳга тегишлилик эҳтимоли, t_i – i сонли гуруҳдаги маълумотни қайта ишлаш учун алгоритмга керак бўладиган вақт деб белгиланган.

Баъзи ҳолларда биз кирувчи маълумотларнинг ҳар бир гуруҳга тушиш эҳтимолини бир Хилл деб тахмин қиламиз. Бошқача айтганда, агар гуруҳ 5 та бўлса, биринчи гуруҳга тушиш эхтимоли иккинчи ёки бошқа гуруҳга тушиш эхтимолидек, яъни ҳар бир гуруҳга тушиш эхтимоли 0,2 га тенг. Бу ҳолда ишнинг ўртача вақтини аввалги формула Билан ёки унга эквивалент соддалаштирилган барча гуруҳларнинг тенг эхтимоллигида ҳақиқий бўлган

$$A(n)\sum_{i=1}^{m} p_i t_i, \mathbf{(7.2)}$$

формуладан фойдаланишимиз мумкин.

Логарифмлар

Бизнинг таҳлилимизда логарифмлар сезиларли ўрин эгаллайди, шунинг учун уларнинг хоссаларини кўриб чиқишимиз лозим. *x* сонининг *y* асос бўйича логарифми деб шундай даражага айтиладики, *x* ни олиш учун *y* ни кўтариш керак бўлади. Масалан, $\log_{10}45$ тахминан 1,653 га тенг, чунки $10^{1.653} \sim 45$. Логарифмнинг асоси ихтиёрий сон бўлиши мумкин, лекин бизнинг таҳлилимизда кўпроқ 10 ва 2 асосли логарифмлар учрайди.

Логарифм – ўсиб борувчи функция. Бу дегани, агар $X > Y$ бўлса, ҳар бир B асос учун $\log_B X > \log_B Y$. Логарифм – ўзаро бир ифодали функция. Бу дегани, агар $\log_B X = \log_B Y$ бўлса $X=Y$ бўлади. Шунингдек, логарифмнинг унга кирувчи ўзгарувчиларнинг мусбат қийматида тўғри бўлган қуйидаги муҳим хоссаларини билиш лозим:

$$\log_B 1 = 0; \quad (7.3)$$

$$\log_B B = 1; \quad (1.4)$$

$$\log_B(XY) = \log_B X + \log_B Y; \quad (1.5)$$

$$\log_B X^Y = Y\log_B X; \quad (1.6)$$

$$\log_A X = \frac{\log_B X}{\log_B A} \quad (7.7)$$

шу хоссалар ёрдамида функцияни соддалаштириш мумкин. (7.7) хоссаси логарифмнинг асосини алмаштириш имконини беради. Кўплаб калькуляторлар 10 асосли логарифмлар ва натурал логарифмларни ҳисоблайди. $\log_{42}75$ ни ҳисоблаш учун нима қиласиз? (7.7) тенглиги ёрдамида 1.155 жавобини олишингиз мумкин.

Бинар дарахтлар

бинар дарахти шундай тузилишга эгаки, ундаги ҳар бир тугун иккита тугундан ортиқ бўлмаган бир аждод наслдан иборат бўлади. Дарахтнинг энг юқори тугуни ягона аждодсиз тугун ҳисобланади; у илдизли тугун деб аталади. N тугунли бинар дарахти кам $[\log_2 N+1]$ тугунга эга (тугунларнинг максимал зичлигида). Масалан, 15 тугунли тўла бинар дарахтида бир илдиз, иккинчи даражада 2 та тугун, 3-даражада 4 та тугун ва 4-даражада 8 та тугун бор; бизнинг тенглигимиз ҳам $[\log_2 15]+1=[3.9]+1=4$ даражани беради. Дарахтга яна бир тугуннинг қўшилиши янги даражанинг ҳосил бўлишига олиб келади ва уларнинг сони тенг бўлади $[\log_2 16] + 1 = [4] + 1 = 5$. N тугунли энг катта бинар дарахти N даражага эга: бу дарахтнинг ҳар бир тугунида битта насл бор (дарахтнинг ўзи ҳам оддий рўйхат кўринишига эга).

Агар дарахтнинг даражаларини рақамлаб чиқсак, илдиз 1 даражада деб ҳисоблаб, *K* рақамли даражада 2^{K-1} тугуни ётади. *J* даражали (1 дан *J* гача рақамланган) тўла бинар дарахтида ҳамма барглар *J* рақамли даражада ётади ва ҳар бир тугунда бирдан *J-1* даражада иккита тўғридан-тўғри насл бор. *J*

даражали тўла бинар дарахтида 2^J - 1 тугун бор. Бу ахборот кейинчалик ҳам сизга асқотиши мумкин. Бу формулаларни яхшироқ тушуниш учун бинар дарахтларини чизишни ва натижаларингизни юқоридаги формулалар билан солиштиришингизни маслаҳат берамиз.

Эхтимолликлар

Биз алгоритмларни кирувчи маълумотларга кўра таҳлил қилмоқчимиз, бунинг учун эса у ёки бу кирувчи маълумотлар тўплами қанчалик кўп учрашини баҳолашимиз керак. Шу билан бирга, биз кирувчи маълумотлар у ёки бу шароитларга тўгри келиш эхтимоллиги билан ишлашимизга тўғри келади. У ёки бу ҳодисанинг эхтимоллиги нол ва бир оралиғидаги сондан иборат, 0 эхтимоллиги ҳодиса ҳеч қачон содир бўлмаслиги,1 эхтимоли эса бўлиши мумкинлигини билдиради. Агар бизга турли кирувчи қийматларнинг сони аниқ 10 га тенглиги маълум бўлса, ишонч Билан айтишимиз мумкинки, ҳар қандай бундай киришнинг эхтимоллиги 0 ва 1 оралиғида бўлади, барча эхтимолликларнинг йиғиндиси 1 га тенг, чунки улардан биттаси амалга ошиши мумкин. Агар ҳар бир киришнинг амалга ошиш эхтимоллиги бир хил бўлса, улардан ҳар бирининг эхтимоллиги 0.1 га тенг бўлади (10 дан 1 ёки 1/10).

Бизнинг таҳлилимиз, асосан барча имкониятларни кўриб чиқишдан иборат бўлади, кейин эса биз уларнинг ҳаммаси тенг эхтимолли деб фараз қиламиз. Агар имкониятларнинг умумий сони N га тенг бўлса, улардан ҳар бирининг амалга ошиши эхтимоллиги 1/N га тенг бўлади.

Қўшиш формулалари

Алгоритмларни таҳлил қилганда биз баъзи катталиклар йиғиндисини қўшишимизга тўғри келади. Айтайлик, бизда цикли алгоритм бор. Агар цикл ўзгарувчиси 5 қийматини олса, цикл 5 марта бажарилади, агар унинг қиймати 20 га тенг бўлса 20 бўлади. Агар цикл ўзгарувчиси M га тенг бўлса, цикл M марта бажарилади. Агар цикл ўзгарувчиси 1 дан N гача ҳамма қийматларга ўтса, цикл бажарилишининг жами сони 1 дан N гача бўлган ҳамма натурал сонлар йиғиндисига тенг бўлади. Бу йиғиндини биз $\sum_{i=1}^{N} i$ кўринишида ёзамиз. Йиғинди белгисининг пастки қисмида ўзгарувчи йиғиндининг бошланғич қиймати, юқори қисмида эса – охирги қиймати турибди. Бундай ифодаланиш бизни қизиқтирган йиғинди билан қандай боғлиқлиги тушунарли.

Агар бирор қиймат шу каби йиғинди кўринишида ёзилса, натижани бошқа шу каби ифодалар билан солиштириш мумкин бўлиши учун уни соддалаштириш керак. Икки сондан каттаси $\sum_{i=11}^{N}(i^2-i)u\sum_{i=0}^{N}(i^2-20i)$. Шунинг

учун йиғиндини соддалаштириш учун биз қуйидаги формулалардан фойдаланамиз, бунда C - i га боғлиқ бўлмаган ўзгарувчи.

$$\sum_{i=1}^{N} Ci = C\sum_{i=1}^{N} i \qquad (7.8)$$

$$\sum_{i=L}^{N} i = \sum_{i=0}^{N-L} (i+L) \qquad (7.9)$$

$$\sum_{i=L}^{N} i = \sum_{i=0}^{N} i - \sum_{i=0}^{L-1} i \qquad (7.10)$$

$$\sum_{i=1}^{N} (A+B) = \sum_{i=1}^{N} A + \sum_{i=1}^{N} B \qquad (7.11)$$

$$\sum_{i=0}^{N} (N-i) = \sum_{i=0}^{N} i \qquad (7.12)$$

(7.12) тенглиги 0 дан N гача сонлар йиғиндиси N дан 0 гача сонлар йиғиндисига тенглигини билдиради. Баъзи ҳолларда бу тенгликни қўллаш ҳисоблашларни енгиллаштиради.

$$\sum_{i=1}^{N} 1 = N \quad (7.13)$$

$$\sum_{i=1}^{N} C = CN \quad (7.14)$$

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2} \quad (7.15)$$

(7.15) тенглигини ёдлаб олиш осон, агар 1 дан N гача бўлган сонларни жуфтликка ажратсак. 1 ни N га қўшиб, 2 ни N-1 га ва ҳоказо, биз ҳар бири N+1 га тенг сонлар тўпламини оламиз. Бундай сонлар нечта бўлади? Албатта, уларнинг сони жуфт қилиб ажратилган элементлар сонининг ярмига тенг, яъни N/2. Шунинг учун барча N сонларининг йиғиндиси (7.18) га тенг.

$$\frac{N}{2}(N+1) = \frac{N(N+1)}{2} \qquad (7.16)$$

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} = \frac{2N^3+3N^2+N}{6} \qquad (7.17)$$

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1 \qquad (7.18)$$

(7.17) тенглигини икки бўлинадиган сонлар орқали эслаб қолиши мумкин. 0 дан 10 гача бўлган икки даражасининг йиғиндиси 11111111111 иккиланган сонига тенг. Бу сонга 1 ни қўшиб 100000000000, яъни 2^{11} ни ҳосил қиламиз. Лекин бу натижа нолдан ўнггача бўлган икки даражаларининг йиғиндисидан 1 га кўп, шунинг учун йиғиндининг ўзи 2^{11}-1 га тенг. Энди агар 10 ўрнига N қўйсак, биз (7.17) тенгликка келамиз.

Ҳар қандай сон учун

$$\sum_{i=1}^{N} A^i = \frac{A^{N+1}-1}{A-1} \text{ ихтиёрий А сони учун} \qquad (7.19)$$

$$\sum_{i=1}^{N} i2^i = (N-1)2^{N+1}+2 \qquad (7.20)$$

$$\sum_{i=1}^{N} \frac{1}{i} \approx \ln N \qquad (7.21)$$

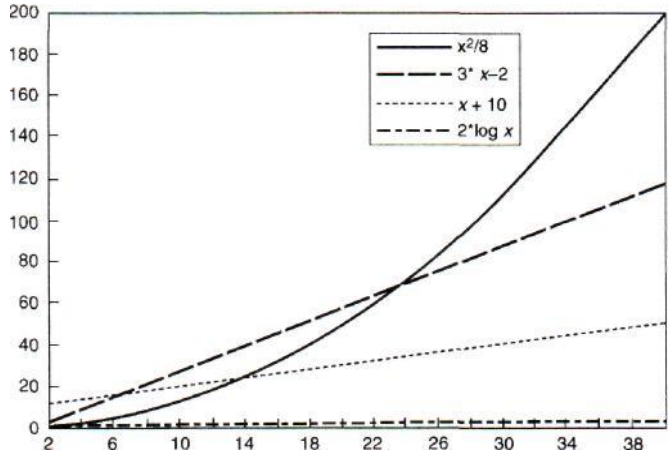
$$\sum_{i=1}^{N} \log_2 i \approx N \log_2 N - 1.5 \qquad (7.22)$$

Йиғиндиларни соддалаштиришда аввал уларни (7.8)-(7.12) тенглик ёрдамида янада оддий сонларга ажратиш, сўнгра йиғиндиларни бошқа айниятлар ёрдамида алмаштириш мумкин.

Ўсиш тезликлари

Алгоритм билан бажариладиган жараёнлар сонини аниқ билиш алгоритмларни таҳлил қилишда муҳим роль ўйнамайди. Кирувчи маълумотларнинг ҳажми кўпайганида бу соннинг ўсиш тезлиги муҳимроқ ҳисобланади. У алгоритмнинг ўсиш тезлиги деб аталади.

Агар 7.1-расмга диққат Билан қарасак, функция графикларининг қуйидаги хусусиятларини кўрсатиш мумкин. x^2 функция аввал секин ўсади, лекин x ўсганда унинг ўсиш тезлиги ҳам ошади. x функциясининг ўсиш тезлиги ўзгарувчининг ҳамма қийматлари оралиғида доимийдир. 2 log x функцияси умуман ўсмайди, лекин бу ёлғон тасаввур. Ҳақиқатда эса у ўсади, фақат жуда секин.



1.1. – расм. Тўрт функциянинг графиги.

Функция қийматларининг нисбий баландлиги биз кўриб чиқаётган ўзгарувчининг қийматлари ката ёки кичиклигига боғлиқ. x=2 да функция қийматини таққослаймиз. Бу нуқтадаги энг кичик қийматли функция $x^2/8$; энг ката қийматли функция x+10 ҳисобланади. Бироқ x ортиши билан $x^2/8$ функция оша бошлайди.

Алгоритмларни таҳлил қилиш пайтида бизни алгоритмнинг ўсиш тезлиги қизиқтиради. Функциянинг нисбий «ўлчами» бизни *x* ўзгарувчининг ката қийматларида қизиқтиради.

Баъзи кўп учрайдиган функция синфлари 7.2-расмдаги жадвалда келтирилган. Бу жадвалда биз берилган синфнинг функция қийматини эркин ўзгарувчан катталик қийматининг кенг диапазонида келтирдик. Кўриниб турибдики, кирувчи маълумотларнинг оз ўлчамларида функция қийматлари сезиларли фарқ қилмайди, аммо бу ўлчамлари ўсганда фарқ сезиларли ошади. бу жадвал 7.1-расмдан таассуротни кучайтиради. Шунинг учун биз кирувчи маълумотларнинг ката ҳажмларида нима содир бўлишини ўрганамиз, кичик ҳажмлардаги фарқ кўринмас бўлганлиги сабабли.

7.1 ва 7.2-расмлардаги маълумотлар функцияларнинг Яна бир хоссасини намойиш этади. Тез ўсувчи функциялар секин ўсувчи функциялардан устунлик қилади. Шунинг учун биз агар алгоритм мураккаблиги икки ёки бир неча функциялар йиғиндисидан иборат бўлса, тез ўсувчи функциялардан ташқари барча функцияларни олиб ташлаймиз.

	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	$2n^2$
1	0.0	1.0	0.0	1.0	1.0	2.0
2	1.0	2.0	2.0	4.0	8.0	4.0
5	2.3	5.0	11.6	25.0	125.0	32.0
10	3.3	10.0	33.2	100.0	1000.0	1024.0
15	3.9	15.0	58.6	225.0	3375.0	32768.0
20	4.3	20.0	86.4	400.0	8000.0	1048576.0
30	4.9	30.0	147.2	900.0	27000.0	1073741824.0
40	5.3	40.0	212.9	1600.0	64000.0	1099511627776.0
50	5.6	50.0	282.2	2500.0	125000.0	1125899906842620.0
60	5.9	60.0	354.4	3600.0	216000.0	1152921504606850000.0
70	6.1	70.0	429.0	4900.0	343000.0	1180591620717410000000.0
80	6.3	80.0	505.8	6400.0	512000.0	1208925819614630000000000.0
90	6.5	90.0	584.3	8100.0	729000.0	1237940039285380000000000000.0
100	6.6	100.0	664.4	10000.0	1000000.0	1267650600228230000000000000000.0

Функцияларнинг ўсиш синфлари

масалан, агар биз алгоритмни таҳлил қилганда, унинг x^3-30x таққослаш қилишини билсак, алгоритмнинг мураккаблиги x^3 каби ўсади, деймиз. Бунинг сабаби шундаки, 100 та кирувчи рақамларда x^3 ва орасидаги фарқ атига 0,3 % ни ташкил қилади. Кейинги бўлимда биз бу фикрни формаллаштирамиз.

Ўсиш тезликларини таснифлаш

Алгоритм мураккаблигининг ўсиш тезлиги муҳим роль ўйнайди ва биз ўсиш тезлиги формуласи ката устунликка эга ҳади билан аниқланишини кўрдик. Шунинг учун биз секин ўсадиган кичик ҳадларга эътибор қаратмаймиз. Барча кичик ҳадларни олиб ташлаб, мураккабликнинг ўсиш тезлиги ҳисобланувчи алгоритм ёки функциянинг тартибига эга бўламиз. Алгоритмларни улар мураккаблигининг ўсиш тезлигига қараб гуруҳларга ажратиш мумкин. Биз 3 тоифани киритамиз: мураккаблиги мазкур функция каби тез ўсувчи алгоритмлар, мураккабликлари ўша тезликда ўсувчи алгоритмлар ва мураккаблиги бу функциядан секин ўсувчи алгоритмлар.

Катта омега

f сингари тез ўсувчи функциялар синфини биз Ω(f) орқали белгилаймиз (катта омега деб ўқилади). Агар ҳамма қийматларда эркин ўзгарувчан катталик n, баъзи ката чегарада n_0 , $g(n) > cf(n)$ қиймати баъзи мусбат с сон учун бўлса, g функцияси шу синфга тегишли бўлади. Ω(f) синфи ўзининг қуйи чегараси билан изоҳланса, ундаги ҳамма функциялар f каби тез ўсади.

Биз алгоритмларнинг самарадорлиги билан қизиқамиз, шунинг учун Ω(f) синфи бизни у даражада қизиқтирмайди: масалан, $\Omega(n^2)$ га n^2 дан тез ўсувчи ҳамма функциялар киради, айтайлик n^3 ва2^n.

Катта О

Спектрнинг бошқа тарафида О(f) синфи жойлашган (катта О деб ўқилади). Бу синф f дан тез ўсмайдиган функциялардан ташкил топган. Функция О(f) синфлари учун юқори чегарани ҳосил қилади. Формал нуқтаи назардан f функцияси О(f) синфига тегишли, агар барча n учун $g(n) \le cf(n)$, баъзи чегара катта О ва баъзи мусбат с константа учун бўлса.

Бу синф биз учун жуда муҳим. Иккита алгоритмдан қайси бирининг мураккаблиги катта О синфига тегишлиги бизни қизиқтиради.

Катта тета

в(Θ) орқали биз f сингари тезликда ўсувчи функциялар синфини белгилаймиз (катта тета деб ўқилади). Формал нуқтаи назардан бу синф икки аввалги синфларнинг кесишувидан иборат, Θ (f) = Ω (f) ∩О(f).

Алгоритмларни таққослаганда бизни ўрганилган масалалардан тезроқ ечувчилари қизиқтиради. Шунинг учун агар топилган алгоритм Θ синфига тегишли бўлса, биз уни кўрб чиқмаймиз.

Катта О синфининг тавсифи

Берилган функция О(f) га тегишли эканлигини икки хил йўл билан текшириш мумкин: юқоридаги тавсиф орқали ёки қуйидаги тавсиф ёрдамида:

$$g \in O(f), агар \lim_{n\to\infty} \frac{g(n)}{f(n)} = \text{с ихтиёрий с константа учун.} \qquad (1.23)$$

Бу шуни англатадики, *g(n)/f(n)* нинг муносабатлар чегараси мавжуд бўлса ва у чексизликдан кичик бўлса, g€O *(f)* бўлади. Баъзи функциялар учун буни текшириш осон эмас. Лопиталь қонунига кўра, бундай ҳолда функциялар чегарасини уларнинг ҳосиласи чегарасида алмаштириш мумкин.

Белгилаш

Θ(f),Ω(f), ва O(f) синфларининг ҳар бири тўплам ҳисобланади, шунинг учун «g – шу тўплам элементи» ибораси аҳамиятлидир. Бироқ, таҳлилларда g = O*(f)*деб ёзишади, аслида эса g€O *(f)***.**

«Тақсимла ва бошқар» кўринишидаги алгоритмлар

Кириш қисмида айтиб ўтилганидек, «тақсимла ва бошқар» кўринишидаги алгоритмлар турли масалаларни ечиш учун ихчам ва кучли қуролни таъминлайди. Бу бўлимда биз бундай алгоритмларни ишлаб чиқиш эмас, балки уларнинг таҳлили билан шуғулланамиз. Цикларда таққослашлар сонини ҳисоблашда циклнинг бир итерациясидаги таққослашлар сонини ҳисоблаб, уни итерациялар сонига кўпайтириш етарли. Агар ички циклнинг итерациялар сони ташқи параметрларга боғлиқ бўлса, ҳисоблаш қийинроқ бўлади. «Тақсимла ва бошқар» кўринишидаги алгоритмлар итерациясини ҳисоблаш оддий эмас: у рекурсив чақириқлар, тайёрлов ва якунланувчи ҳаракатларга боғлиқ. рекурсив чақирувда у ёки бу функциянинг неча марта бажарилиши, одатда, аниқ эмас. Мисол сифатида қуйидаги «тақсимла ва бошқар» кўринишидаги алгоритмни кўриб чиқамиз:

```
DivideAndConquer (data, N, solution)
data        //кирувчи маълумотлар тўплами
N       //тўпламдаги қийматлар сони
solution    //масала ечими
if  (N <= SizeLimit)  then
DirectSolution(data,N,solution) else
DivideInput(data,N,smallerSets,smallerSizes,numberSmaller) for i=l to
numberSmaller do
DivideAndConquer(smallerSets[i],smallerSizes[i],smallSol[i]) end for
CombineSolutions(smallSol.numberSmaller,solution)
end if
```

Бу алгоритм ечимини оддий рекурсив бўлмаган алгоритм (матнда DirectSolution деб аталувчи) ёрдамида топиш учун вазифанинг ҳажми кичик эмаслигини аввал текширади ва агар шундай бўлса, унинг чақириғи содир бўлади. Агар вазифа жуда ката бўлса, аввал Divide Input амали чақирилади, у кирувчи рақамларни бир қанча кичик тўпламларга бўлади (уларнинг сони numberSmaller га тенг). Кирувчи дастлабки тўпламнинг ҳар бир қисми

кичикроқ тўпламлардан бирига тушади, аммо бир элементнинг ўзи бир нечта тўпламга тушиши мумкин. Сўнг ҳар бир кичик кирувчи тўпламда DivideAndConquer алгоритмининг рекурсив чақируви содир бўлади, CombineSolutions функцияси олинган натижаларни бирлаштиради.

Натурал соннинг факториалини циклда ҳисоблаш қийин эмас, бироқ қуйида берилган мисолда бизга ушбу ҳисоблашнинг рекурсив варианти керак бўлади. JV сонининг факториали N ни, N — 1 сонининг факториалига кўпайтирилганига тенг. Шунинг учун қуйидаги алгоритм керак бўлади:

```
Factorial(N)
N       // число, факториал которого нам нужен
Factorial  // возвращает целое число
if (N=l) then
return 1 else
smaller = N-l
answer=Factorial(smaller)
return (N*answer) end if
```

Бу алгоритмнинг қадамлари оддий ва тушунарли ва биз уларни юқорида келтирилган стандарт алгоритм Билан солиштиришимиз мумкин. Биз аввалроқ бу бўлимда кўпайтириш амали қўшишдан мураккаброқ, деб эслатган бўлсакда, кўпайтиришни алоҳида ҳисоблаш керак. Мисолни соддалаштириш учун биз бу чегарани ҳисобга олмаймиз.

Бу икки алгоритмни таққосласак, рақамлар ўлчамининг чегараси 1 эканлиги кўринади ва бунда ҳеч қандай арифметик амал бажарилмайди. Бошқа барча ҳолларда биз else даражасига ўтамиз. Умумий алгоритмдаги биринчи қадам «киришнинг кичик қисмларга бўлиниши» бўлади; ҳисоблаш алгоритмида бу қадамнинг факториалига битта айиришни талаб қилувчи smaller ўзгарувчисини ҳисоблаш тўғри келади. Умумий ва бошқар алгоритмининг кейинги қадами кичикроқ рақамларни қайта ишлаш амалини рекурсив чақириш ҳисобланади; факториални ҳисоблаш алгоритмида – бу бир рекурсив чақириқ ва ундагивазифанинг ҳажми аввалгидан битта кам.

Умумий алгоритмдаги охирги қадам тенгламаларни бирлаштиришдан иборат; факториални ҳисоблаш алгоритмида бу кўпайтириш охирги return операторида бажарилади.

Рекурсив алгоритм самарадорлиги

Рекурсив алгоритм қанчалик самарадор? Агар биз алгоритмни ҳисоблаш бевосита иккинчи даражали, кирувчи берилганларнинг бўлиниши

логарифмик, ечимларнинг бирлашиши эса чизиқли (ҳаммаси кирувчи берилганларнинг ҳажмига боғлиқ) десак, кирувчи берилганлар саккиз қисмга бўлинади ва уларнинг ҳар бири бошланғич берилганларнинг тўртдан бир қисмига тенг эканлигини билиб унинг самарадорлигини кўрсата оламизми? Бу масалани ечиш осон эмас, ҳатто унга қандай киришиш ҳам аниқмас. Бироқ, «тақсимла ва бошқар» кўринишидаги алгоритмлар таҳлили жуда содда экан, агар алгоритмдаги қадамлар юқорида кўрсатилган умумий ҳолдаги алгоритм қадамларига мос бўлса, бевосита ҳисоблаш, киришнинг бўлиниши, рекурсив чақирувларнинг баъзи миқдорлари ва ҳосил қилинган натижаларнинг бирлашуви, агар бу қадамлар бир-бири билан қандай мос тушишини ва ҳар бир қадамнинг мураккаблиги аниқ бўлса, у ҳолда «тақсимла ва бошқар» кўринишидаги алгоритм мураккаблигини аниқлаш учун қуйидаги формуладан фойдаланиш мумкин:

$$DAC(N)=\begin{cases} DIR(N) & \text{при } N \le SizeLimit, \\ DIV(N)+\sum_{i=1}^{numberSubdir} DAC(smallerSize[i])+COM(N) & \end{cases}$$

бу ерда DAC —**DivideAndConquer** алгоритм мураккаблиги**,**
DIR —**Direct Solution** алгоритм мураккаблиги,
DIV —**Divide Input** алгоритм мураккаблиги**,**
COM —**CombineSolutions** алгоритм мураккаблиги.

Бу умумий формула асосида юқорида қўйилган саволга жавоб бериш жуда осон. Биз фақат умумий формулага ҳар бир қисмнинг маълум мураккабликларини қўйиб чиқишимиз керак. Натижада қуйидагига эришамиз:

$$DAC(N)=\begin{cases} N^2 & \text{при } N \le SizeLimit \\ \log_2 N+\sum_{i=1}^{8} DAC(N/4)+N & \text{при } N > SizeLimet \end{cases}$$

ёки, барча кичик тўпламлар ҳажми бир хил бўлганлиги учун бундан ҳам осонроқ усули:

$$DAC(N)=\begin{cases} N^2 & \text{при } N \le SizeLimit \\ \log_2 N+8DAC(N/4)+N & \text{при } N > SizeLimet \end{cases}$$

Бундай кўринишдаги тенглик реккурент деб номланади, чунки функция қиймати ўз атамасида ифодаланган. Биз фақат N га боғлиқ бўлган ва худди

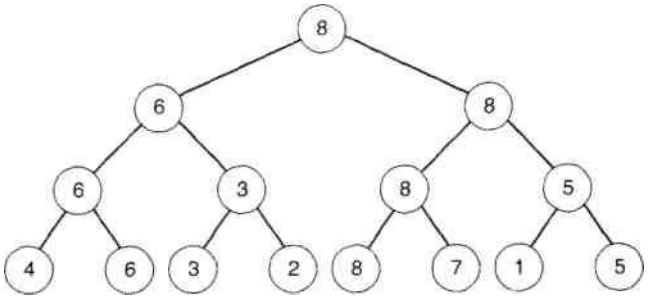
шу функциянинг бошқа элементларига боғлиқ бўлмаган мураккаблик учун ифодани топишимиз керак. Бундай тенгликлардан рекурсияни ўчириш § 1.6 да кўриб чиқилади, у ерда реккурент муносабатлар тўлиқ ўрганилади.

Факториал билан боғлиқ мисолга қайтамиз. Биз факториал билан ҳисобланадиган алгоритмнинг барча босқичларини умумий DivideAndConquer алгоритми билан таққосладик. Бу таққослашдан фойдаланамиз ва юқорида келтирилган умумий формулага қўйиш керак бўлган қийматни аниқлаймиз. Factorial функциясидаги бевосита ҳисоблашлар амалларни талаб қилмайди, кирувчи берилганларнинг бўлиниши ва натижаларнинг бирлашуви алгоритмларининг ҳар бири биттадан амални талаб этади ва рекурсив чақирув масалани ечади, берилганлар ҳажми бошланғич берилганлардан биттага камаяди. Натижада Factorial функциясида ҳисоблашлар миқдори учун қуйидаги реккурент муносабатни ҳосил қиламиз:

$$Calc(N)=\begin{cases}0 & при N=1,\\ 1+calc(N-1)+1 & при N>1.\end{cases}$$

Турнирлар методи

Турнирлар методи рекурсияга асосланган ва унинг ёрдамида кўплаб масалаларни ечиш мумкин, маълумотлар бўйича биринчи ўтиш натижасида олинган ахборот кейинги про-ютларни енгиллаштиради. Агар биз энг катта қийматни излаш учун ундан фойдалансак, ҳамма элементлари і inn г г іт баргли бинар дарахтни тузиш талаб қилинади. Ҳар бир босқичда икки элемент жуфтларга бирлашган бўлиб, улардан энг каттаси оталик релига нусхаланади. Жараён илдиз тугунга етгунча такрорланади. Қайд этилган берилганлар тўплами учун тўлиқ турнир дарахти 1.3. расмда тасвирланган.



Саккиз қийматдан иборат тўплам учун турнир дарахти

N тартибли таққослашлар миқдори ҳисобига рўйхатдаги катталиги бўйича иккинчи элементни топиш алгоритми ишлаб чиқилади. Бунда бизга турнирлар методи ёрдам беради. Ҳар қандай таққослаш натижасида биз «ғолиб» ва «мағлуб»га эга бўламиз. Мағлубларни унутамиз, дарахт бўйлаб юқорига фақат ғолиблар ҳаракат қилади. Энг ката элементдан ташқари ҳар қайси элемент ҳеч бўлмаганда битта таққослашда мағлубиятга учрайди.

Шунинг учун турнир дарахтини қуриш учун N — 1 та таққослаш талаб этилади.

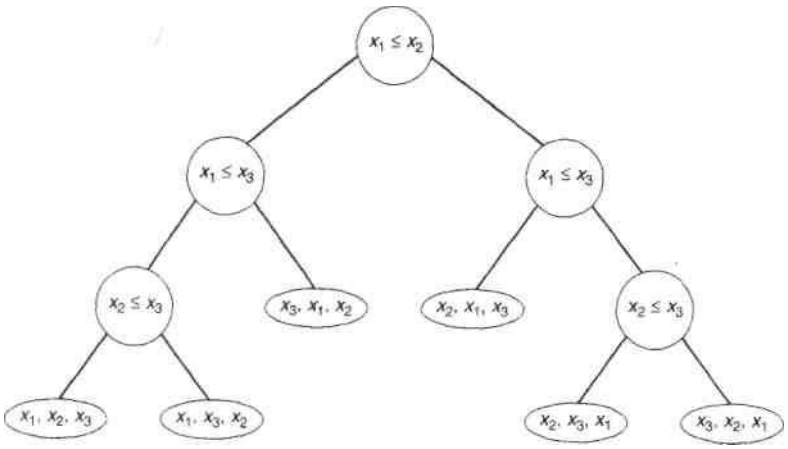
Катталиги бўйича иккинчи элемент энг катта элементдан мағлуб бўлиши мумкин. Дарахтдан пастга томон юриш давомида энг катта элементга ютқазган элементлар рўйхатини тузамиз. 1.3.2 бўлимдаги дарахт учун формула бундай элементлар сони $[\log_2 N]$ дан ортиб кетмаслигини кўрсатади. Уларнинг дарахт бўйича қидируви $[\log_2 N]$ таққослашни талаб қилади; яна $[\log_2 N]$ - 1 таққослашлар улардан энг каттасини танлаш учун керак. Ҳаммаси бўлиб бу ишга $N + 2[\log_2 N] —2$ кетади, яъни $O(N)$ та таққослаш.

Турнирлар методи ёрдамида қийматлар рўйхатини саралаш мумкин. 3-бобда турнирлар методига асосланган уюмларни саралашни кўриб чиқамиз.

Қуйи чегаралар

Алгоритм, агар ундан тезроқ ишлайдиган алгоритм мавжуд бўлмаса, оптимал ҳисобланади. Бизнинг алгоритм оптимал эканлигини қандай билишимиз мумкин? Ёки оптимал эмас, фақатгина етарлича яхши бўлиши ҳам мумкин? Бу саволларга жавоб бериш учун биз аниқ масалани ечиш учун керак бўлган амаллар миқдорини энг кам ҳолатда бўлса ҳам билишимиз керак. Бунинг учун масалани ечувчи алгоритмни эмас, балки айнан масалани ўрганиш керак. Натижада ҳосил қилинган қуйи чегара бу масалани ечиш учун керак бўлган иш ҳажмини кўрсатади ва бунга номзод бўлган тезроқ ишловчи ихтиёрий алгоритм баъзи жараёнларга нотўғри ишлов бериши шарт.

Учта сондан иборат рўйхатни саралаш жараёнини таҳлил қилиш учун яна бинар дарахтидан фойдаланамиз. Дарахтнинг ички тугунларини таққосланган элементлар жуфтлиги билан белгилаймиз. Дарахт новдасидан ўтиши таъминланган элементлар рўйхати дарахтнинг мос баргида тасвирланади. Уч элементдан иборат рўйхат учун дарахт **7.4 расмда тасвирланган.** Бундай дарахт ечим дарахти деб номланади.



Уч элементли рўйхатни саралаш учун ечим дарахти

Ҳар бир саралаш алгоритми қайси элементларини таққослашига боғлиқ ҳолда ўзининг ечимлар дарахтини яратади. Ечимлар дарахтидаги илдиздан баргга боришнинг энг узун йўли энг ёмон ҳолга мос келади. Қисқа йўл энг яхши ҳол ҳисобланади. Ўрта ҳол ечимлар дарахтидаги қирралар сонини ундаги барглар сонига бўлиш билан ифодаланади. Бир қарашда ечимлар дарахтини чизиш ва керакли сонларни ҳисоблаш қийинмасдек туюлади. Лекин 10 та сони саралашдаги ечимлар дарахтининг ҳажмини тасаввур қилинг. Таъкидланганидек, мумкин бўлган тартиблар сони 3 628 800 га тенг. Шунинг учун дарахтда камида 3 628 800 та барг бўлади, ундан кўпроқ ҳам бўлиши мумкин, чунки турли таққослашлар кетма-кетлиги натижасида битта тартибга такрорий келиш мумкин. Бундай дарахтда даражалар сони 22 дан кам бўлмаслиги керак.

Алгоритм мураккаблиги учун ечимлар дарахти ёрдамида қандай қилиб чегараларни ҳосил қилиш мумкин? Биламизки, аниқ алгоритм элементларнинг бошланғич тартибига боғлиқ бўлмаган ҳолда ихтиёрий рўйхатни тартиблаши керак. Кирувчи қийматларнинг ихтиёрий ўрин алмашуви учун камида битта барг мавжуд бўлиши керак, бу эса ечимлар дарахтида барглар сони $N!$ дан кам бўлмаслиги кераклигини билдиради. Агар алгоритм ҳақиқатда самарали бўлса, у ҳолда ундаги ҳар бир ўрин алмаштириш бир марта учрайди. $N!$ баргли дарахтда нечта даража бўлиши керак? Биз ҳар бир навбатдаги даражада олдингисига қараганда тугунлар икки маротаба кўплигини кўрдик. K даражадаги тугунлар сони 2^{K-1} га тенг, шунинг учун ечимлар дарахтимизда L та даража бўлади, бу ерда L — $N! \le 2^{L-1}$ учун энг кичик сон. Бу тенгсизликни логарифмлаб, қуйидагини ҳосил қиламиз:

$$log_2 N! \le L - 1.$$

L нинг энг кичик қийматини топиш учун логарифмдан қутулиш мумкинми? Факториал хоссаларига мурожаат қиламиз. Қуйидагидан фойдаланамиз

$$Log_2N! = log_2(N(N-1)(N-2)...1),$$

Бу ерда (1.5) га тенг

$$log_2(N(N-1)(JV-2)...1) = \log_2 N + \log_2 (N-1) + \bullet\bullet\bullet + \log_2 1 = \sum_{i=1}^{N} \log_2 i .$$

(7.21) тенгликдан қуйидагини ҳосил қиламиз:

$$\sum_{i=1}^{N} \log_2 i \approx N \log_2 N - 1.5, \log_2 (N!) \approx N \log_2 N$$

Бундан келиб чиқадики, ечимлар дарахтидаги L нинг минимал қуйи чегараси саралаш учун $0(N \log_2 N)$ тартибга эга. Энди биз биламизки, $0(N \log N)$ тартибга эга бўлган саралаш алгоритми энг яхши ҳисобланади ва уни оптимал деса бўлади. Бундан ташқари, $0(N \log N)$ та амалдан кўра тезроқ ишлайдиган алгоритм тўғри ишлаши мумкинмас.

Саралаш алгоритми учун қуйи чегарани чиқаришда алгоритм рўйхат элементлари жуфтликларини кема-кет таққослаш асосида ишлайди деб фараз қилинган эди. 3-бобда чизиқли вақтни талаб қиладиган бошқа саралаш алгоритми (илдизсимон саралаш) билан танишамиз. Бу алгоритмда мақсадга эришиш учун калит қийматлари таққосланмайди, балки уларни уюмларга бўлиб ташлайди.

4–маъруза: Рекуррент муносабатлар. Ўрнига қўйиш усули. Ечимни топиш. Нозик нюанслар. Ўзгарувчиларни алмаштириш. Рекурсиялар дарахти усули.

Режа:

1. Рекуррент муносабатлар;
2. Ўрнига қўйиш усули ва ечимни топиш;
3. Нозик нюанслар, ўзгарувчиларни алмаштириш;
4. Ўзгарувчиларни алмаштириш, рекурсиялар дарахти усули;

Рекуррент муносабат

Алгоритм мураккаблиги учун реккурент муносабатлар алгоритм кўринишидан чиқарилади, лекин улар ёрдамида бу мураккабликни дарҳол ҳал этиш осон эмас. Бунинг учун реккурент муносабатни реккурент табиатидан воз кечувчи ёпиқ кўринишга келтириш керак. Бу усулни тушуниш учун бир қатор мисоллар кўриб чиқамиз.

Рекуррент муносабат икки кўринишда берилиши мумкин. Биринчиси содда ҳолатлар кам бўлганда:

$T(n) = 2T(n-2) - 15;$
$T(2) = 40;$
$T(1) = 40.$

Иккинчи шакли содда ҳолатлар миқдори нисбатан кўпроқ бўлган ҳолда ишлатилади:

$$T(n) = \begin{cases} 4, & агарn \le 4 \\ 4T(n/2) - 1 & акс\ холда \end{cases}$$

Бу шакллар эквивалент. Оддий қийматлар рўйхатини чиқариб ташлаб, иккинчисидан биринчисига ўтиш мумкин. Бундан келиб чиқадики, юқорида келтирилган реккурент муносабатлардан иккинчисини қуйидаги кўринишда ёзиш мумкин:

$$T(n) = 4T(n/2) - 1;$$
$$T(4) = 4;$$
$$T(3) = 4;$$
$$T(2) = 4;$$
$$T(1) = 4;$$

Қуйидаги реккурент муносабатни кўриб чиқамиз:

$$T(n) = 2T(n-2)-15;$$
$$T(2) = 40;$$
$$T(1) = 40.$$

Биз биринчи тенгликка $T(n - 2)$ учун эквивалент бўлган ифодани қўймоқчимиз. Бунинг учун ҳар бир n ни $n - 2$ билан алмаштирамиз:

$$T(n - 2) = 2T(n - 2 - 2) - 15 = 2T(n - 4) - 15.$$

Энди $T(n - 4)$ ни чиқариб ташлаш керак. Худди шундай алмаштиришларни бажаришимиз керак. Буни катта бўлмаган қийматлар кетма-кетлигида бажарамиз:

$$T(n-2) = 2T(n-4)-15;$$
$$T(n-4) = 2T(n-6)-15;$$
$$T(n-6) = 2T(n-8)-15;$$
$$T(n-8) = 2T(n-10)-15;$$
$$T(n - 10) = 2T(n - 12) - 15.$$

Энди ҳисоблаш натижаларини бошланғич тенгламага қўямиз. Бунда натижани охиригача қисқартирмаймиз, акс ҳолда умумий схемага эга бўлмаслигимиз мумкин. Ўрин алмаштиришлар қуйидагини беради:

$T(n) = 2T(n-2)-15 = 2(2T(n- 4)-15)-15,$
$T(n) = 4T(n-4)-2 \cdot 15-15;$

T(n)= 4(2T(n-6)-15)-2•15-15,
T(n)= 8T(n-6)-4•15-2•15-15;

T(n)= 8(2T(n-8)-15)- 4•15- 2•15-15,
T(n) = 16T(n-8)- 8 • 15 - 4 • 15 - 2 • 15-15;

T(n)= 16(2T(n-10)-15)- 8•15- 4•15-2•15 - 15,
T(n) = 32T(n-10)-16•15-8•15-4•15-2•15-15;

T(n)= 32(2T(n-12)-15)-16•15-8•15-4•15-2•15-15,
T(n) = 64T(n-12)-32•15-16•15-8•15-4•15-2•15-15.

Балки сиз умумий схемани тушуниб етгандирсиз. Аввалambор, ҳар бир тенгликдаги қўшилувчи иккининг навбатдаги даражасига кўпайтирилган -15 ни кўрсатиб турибди. Иккинчидан, коэффициент T функцияни рекурсив чақирувда иккининг даражаси бўлиб ҳисобланади. Бундан ташқари, T функция аргументи ҳар сафар 2 тага камайиб бормоқда.

Бу жараён қачон тугашини ўзимиздан сўрашимиз мумкин. Бошланғич тенгликка қайтиб T(1) ва T(2) қийматларни қайд қилиб олганимизни эслашимиз мумкин. Бу катталиклардан бирига етиш учун нечта алмаштириш бажариш керак? Жуфт *n* да 2=*n*-(n- 2) га эга бўламиз. Биз (n-2)/2-1 та алмаштириш бажаришимиз керак, бу n/2-1 та -15 кўпайтувчили қўшилувчи ва олдинда иккининг n/2-1 даражасини беради. n=14 да нима бўлишини кўриб чиқамиз. Бу ҳолда бешта алмаштириш бажаришимиз керак: 15 кўпайтувчиси билан олтита қўшилувчига эга бўламиз ва T(2) да коэффициент 2^6 га тенг бўлади. Охирги тенгликда *n*=14 та алмаштиришда худди шундай натижа ҳосил бўлади.

Агар *n* тоқ сон бўлсачи? Бу формулалар илгаригидек ишлайдими? *n*=13 бўлганда кўриб чиқамиз. Тенгликда битта ўзгариш содир бўлади —T функция аргументи 2 эмас 1 бўлади, лекин n/2-1 қиймати 5 га тенг бўлади (6 га эмас). Тоқ n да n/2-1 ўрнига n/2 ни оламиз. Жавобда икки ҳолатни кўриб чиқишимиз керак.

$$T(n) = 2^{2(n/2)-1} - T(2) - 15\sum_{i=0}^{(n/2)-1} 2^i,$$

жуфт n да;

$$T(n) = 2^{2(n/2)-1} \cdot 40 - 15\sum_{i=0}^{(n/2)-1} 2^i$$

$$T(n) = 2^{2(n/} T(2) - 15\sum_{i=0}^{(n/2)} 2^i,$$

тоқ *n* да.

$$T(n)= 2^{2(n/2)} \cdot 40 - 15 \sum_{i=0}^{(n/2)} 2^i$$

Энди, жуфт n сони учун (7.17) тенгликни қўллаб, қуйидагига эга бўламиз

$$T(n) = 2^{(n-2)-1} \cdot 40\text{-}15 \cdot (2^{n/2}\text{-}1)$$
$$= 2^{n/2}20\text{-}2^{n/2} \cdot 30+15$$
$$= 2^{n/2}\ (20\text{-}15)+15$$
$$= 2^{\,n/2}\ \cdot 5+15,$$

тоқ *n* да

$$T(n) = 2^{\,n/2} \cdot 40\text{-}15 \cdot (2^{n/2}\text{-}1)$$
$$= 2^{\,n/2} \cdot 40\text{-}2^{\,n/2}\ \cdot 30+15$$
$$= 2^{\,n/2}(40\text{-}30)+15$$
$$= 2^{\,n/2}\ \cdot 10+15.$$

Яна бир реккурент муносабатни кўриб чиқамиз:

$$T(n) = \begin{cases} 5, & \text{агар} \le 4; \\ 4T(n/2)-1 & \text{— 1 акс ҳолда .} \end{cases}$$

Юқоридаги ҳолдагидек иш юритамиз. Аввал фақат *n* қийматларни қўямиз; бунда ҳар бир навбатдаги тенгликда олдинги аргументнинг ярми ҳосил бўлади. Натижада қуйидаги тенгликларга эга бўламиз:

$$T(n/2) = 4T(n/4)\text{-}1;$$
$$T(n/4) = 4T(n/8)\text{-}1;$$
$$T(n/8) = 4T(n/16) - 1;$$
$$T(n/16) = 4T(n/32) - 1;$$
$$T(n/32) = 4T(n/64) - 1.$$

Бу тенгликларни бошланғич тенгламага қўйсак:

$T(n) = 4T(n/2)\text{-}1 = 4(T(n/4)\text{-}1)\text{-}1,$

$T(n) = 16T(n/4)\ \text{-}4 \cdot 1\text{-}1;$

$T(n) = 16(4T(n/8)\text{-}1)\text{-}4 \cdot 1\text{-}1,$

$T(n) = 64T(n/8)\text{-}16 \cdot 1\text{-}4 \cdot 1\text{-}1;$

$T(n) = 64(T(n/16)\text{-}1)\text{-}16 \cdot 1\text{-}4 \cdot 1\text{-}1,$

$T(n) = 256T(n/16)\text{-}64 \cdot 1\text{-}16 \cdot 1\text{-}4 \cdot 1\text{-}1;$

$T(n) = 256(4T(n/32)\text{-}1)\text{-}64 \cdot 1\text{-}16 \cdot 1\text{-}4 \cdot 1\text{-}1,$

$T(n) = 1024T(n/32)\text{-}256 \cdot 1\text{-}64 \cdot 1\text{-}16 \cdot 1\text{-}4 \cdot 1\text{-}1;$

$T(n)$ = 1024(4T(n/64)-1)-256•1-64•1-16•1-4•1-1,
$T(n)$ = 4096T(n/64)-1024•1-256•1-64•1-16•1-4•1-1.

Кўриниб турибдики, что при -1да коэффициент ҳар бир алмаштиришда бирор тўрт даражага ошади, аргументга бўлаётган иккилик даражаси -1да энг катта тўрт даражадан ҳар сафар 1га кўп бўлади. Бундан ташқари, T да коэффициентдаги тўртнинг даражаси биз аргументни бўлаётган иккининг даражаси каби $T(n/2^i)$ да бу коэффициент 4^r га тенг, кейин эса -4^{i-1} дан -1гача бўлган қўшилувчилар келади. і нинг қандай қийматида алмаштиришлар тўхтатилиши керак? Қийматлар $n \le 4$ да берилганлиги учун, $T(4) = T(n/2^{\log_2 n-2})$ га етиб келганда тўхташи мумкин. Натижада:

$$T(n) = 4^{\log_2 n-2} T(4) - \sum_{i=0}^{\log_2 n-3} 4^i$$

Энди аниқ қийматлар ва (7.18) тенгликдан фойдаланамиз:

$$T(N) = 4^{\log_2 n-2} \cdot 5\frac{4^{\log_2 n-2}-1}{4-1};$$

$$T(N) = 4^{\log_2 n-2} \cdot 5\frac{4^{\log_2 n-2}-1}{3};$$

$$T(N) = \frac{15 \cdot 4^{\log_2 n-2} - 4^{\log_2 n-2} + 1}{3};$$

$$T(N) = \frac{4^{\log_2 n-2}(15-1)+1}{3};$$

$$T(N) = \frac{4^{\log_2 n-2} \cdot 14 + 1}{3}.$$

Кўриниб турибдики, что в замкнутом виде рекуррент муносабатнинг ёпиқ кўриниши жуда ҳам содда бўлмаслиги мумкин, бироқ унга ўтишда рекурсив «чақирув»дан холи бўламиз ва шунинг учун ҳосил қилинган натижаларни тезда таққослаб, уларнинг тартибини аниқлай оламиз.

Дастурлар таҳлили

Фараз қилинг, бизда катта мураккаб дастур бор, у ўйлаганимиздан ҳам секин ишлайди. Унинг ишлашини сезиларли даражада тезлаштирадиган дастурнинг созловчи қисмларини қандай билиш мумкин?

Дастурга қараб кўп ҳисоблашлар ёки цикллар мавжуд бўлган қисмдастурларни (процедура ёки функция деб ҳам номланади) топиш мумкин. Эътиборлироқ бўлсак, танланган қисмдастурлар доим ишлатилмаганлиги учун самара сезиларли эмаслигини англашимиз мумкин. Дастлаб доимий ишлатиладиган қисмдастурларни топиб, уларни яхшилашга

ҳаракат қилиш маъқулроқ. Излаш усулларидн бири бу ҳар бир қисмдастурга биттадан глобал ҳисоблагичлар тўпламини киритишдан иборат. Дастур иш бошлашида ҳисоблагичлар нолга тенглаштирилади. Сўнгра ҳар бир қисмдастурнинг биринчи қаторига мос ҳисоблагични 1 га ошириш буйруғи қўйилади. Қисмдастурга ҳар қандай мурожаатда ҳисоблагич ортиб боради ва иш якунида ҳисоблагич ҳар бир қисмдастурга нечта чақирув бўлганлигини кўрсатади. Шунда қайси қисмдастурлар кўпроқ, қайсилари кам чақирилганлигини кўриш мумкин.

Фараз қиламиз, бизнинг дастурда баъзи оддий қисмдастурлар 50 000 марта, мураккаб қисмдастурлар эса бир марта чақирилди. У ҳолда оддий дастурда битта амални ўчирганда ҳосил бўладиган натижага эришиш учун мураккаб дастурларда амаллар сонини 50 000 мартага камайтириш керак. Битта дастурда оддий яхшиланишни топиш қисм дастурлар гуруҳидаги 50 000 яхшиланишни топишга кўра енгилроқ эканлиги тушунарли.

Ҳисоблагичларни қисмдастурлар даражаларида ҳам ишлатиш мумкин. Бу ҳолатда биз олдиндан топа олишимиз мумкин бўлган ҳар бир керакли нуқтада биттадан глобал ҳисоблагичлар тўпламини яратамиз. Фараз қиламиз, бирор if операторининг қисмлари бўлган then ва else нинг ҳар бири неча марта бажарилишини билмоқчимиз. У ҳолда иккита ҳисоблагич яратиш мумкин ва улардан биринчисини then қисмига тушганда, иккинчисини – else қисмига тушганда катталаштириш мумкин. Дастур охирида ҳисоблагичларда бизни қизиқтирадиган информация бўлади. Умуман олганда, ҳисоблагичларни бошқариш имконияти бўлган ихтиёрий жойда ўрнатиш керак.

Дастур ўз ишини тугатгач ҳисоблагичларда ҳар бир қисмдастурларнинг бажарилиш сони ҳақидаги маълумот бўлади. Кейин ишнинг энг катта ҳажмини бажараётган қисмдастурларни яхшилаш имкониятини кўриб чиқиш керак.

Бу жараён кўплаб компьютер ва тизимлар ишлаб чиқишда жуда муҳим, дастурлар ҳақида маълумот олишнинг автоматик воситаси дастурий таъминоти бор.

5–маъруза: Индикаторли тасодифий миқдор. Хизматчи ёллаш ҳақидаги масалани индикаторли тасодифий миқдорлар ёрдамида таҳлили

Режа:

1. Индикаторли тасодифий миқдор;
2. Хизматчи ёллаш ҳақидаги масалани индикаторли тасодифий миқдорлар ёрдамида таҳлили;
3. Мавзуга доир масалалар ечиш;

Кўпҳадларни ҳисоблашнинг аҳамияти. Қуйидаги умумий кўпҳадларнинг турини оламиз.

$p(x)=a_nx^n+a_{n-1}x^{n-1}+ a_{n-2}x^{n-2} +\dots a_2x^2 + a_1x + a_0$.

Тахмин қиламизки $a_n, \dots, a_0$ нинг ҳамма коэффициэнти аниқ, доимий ва массивга ёзилган. Бу шуни англатадики кўпҳадларни хисоблаш учун ягона кирувчи маълумот сифатида x ҳисобланади, дастур натижаси эса x нуқтасидаги кўпҳадларнинг аҳамияти ҳисобланади .

Стандарт алгоритм тўғри чизиқли :

```
Evaluate(x)
x // кўпҳаднинг аҳамиятини ҳисоблаш нуқтаси.
result=a[0] + a[1]*x
xPower=x
for i=2 to n do
xPower=xPower*x
result=result+a[i]*xPower
end for
return result
```

Бу алгоритм мутлақо шаффоф ва унинг таҳлили аниқ. For циклида иккита кўпайтириш мавжуд, бунда $n - 1$ марта бажарилади. Бундан ташқари бир кўпайтириш циклдан олдин бажарилади. Шу сабабли кўпайтиришлар умумий сони $2n - 1$. Циклда битта қўшув бажарилади. Яна битта қўшув циклдан олдин бажарилади. Шу сабабли қўшувлар умумий сони n.

Горнер схемаси.

Горнер схемаси орқали ҳисоблаш самаралироқ, бунда у мураккаблашмайди. Ушбу чизма кўпҳаднинг қуйидаги кўринишига асосланган:

$p(x) = ((\dots ((a_nx+a_{n-1})x+a_{n-2})x+\dots + a_2)x+a_1)x +a_0$.

Китобхон ушбу кўринишни айнан ўша кўпҳад бераётганини осон текширади. Тегишли алгоритм қуйидаги кўринишга эга :

HornersMethod(x)
x // кўпҳаднинг аҳамиятини ҳисоблаш нуқтаси.
for i=n-1down to 0 do
*result=result*x*
result=result+a[i]
end for
return result

Цикл n марта бажарилади, бунда цикл ичида битта кўпайтириш қўшиш берилган. Шу сабабли Горнер схемаси орқали ҳисоблаш n кўпайтиришларни талаб этади – стандарт алгоритмларга нисбатан кўпайтиришлар сони икки марта камроқ.

Коэффициэнтларни дастлабки қайта ишлаш.

Кўпҳад коэффициэнтларини алгоритм ишидан олдинроқ қайта ишлаш орқали натижани яхшилаш мумкин. Асосий ғоя шундан иборатки, кўпҳадни тасвирлаш кичик даражали кўпҳадлар орқали бўлади. Масалан, x^{256} даражани ҳисоблаш учун параграф бошидаги Evaluate функциясига ўхшаш циклли функциядан фойдаланиш мумкин. Натижада 255 та кўпайтириш бажарилади. Муқобил ёндошув шундан иборатки, *result=x*x* қўйилади ва операция 7 марта бажарилади. Биринчи бажаришдан сўнг ўзгарувчан *result* x^4, иккинчидан кейин x^8 ни, учинчидан кейин x^{26} тўртинчидан кейин x^{32} ни, бешинчидан кейин x^{125} ва еттинчидан сўнг x^{256} ни ташкил этади.

Коэффициэнтларни дастлабки қайта ишлаш услуби ишлаши учун кўпҳадлар унимодал (яъни, катта коэффициэнт a_n га тенг) бўлиши , кўпҳад даражаси эса бирга кам яъни ($n = r^k - 1$ баъзида $k>1$) бўлиши керак. Бу ҳолатлар кўпҳад қуйидаги кўпҳад кўринишда бўлади:

$p(x)= (x^j+b)q(x)+r(x)$, бунда $j=2^{к-1}$

Ҳар икки кўпҳад q ва r аъзолар p га қараганда икки марта кам бўлади. Зарур натижани олиш учун q(x) ва n(x) ни алоҳида - алоҳида ҳисоблаймиз , шундан кейин битта қўшимча кўпайтириш ва иккита қўшишни бажарамиз. Агар бунда *b* нинг аҳамияти тўғри танланса, *q* ҳар икки *q* ва *r* кўпҳадлар унимодал бўлади. Ҳар бирининг даражаси уларнинг ҳар бирига ушбу процедурани қўллаш имкониятини яратади.

Шунда биз кетма – кет қўллаш ҳисоблашни татбиқ этишга олиб келганини кўрамиз. Умумий кўринишдаги кўпҳадлар тўғрисида сўз юритмасдан мисолга эътибор қаратамиз. Кўпҳадни кўриб чиқамиз:

$p(x) = x^7 + 4x^6 – 8x^4 + 6x^3 + 9x^2 + 2x – 3$

Аввал кўпайтирувчи x^j+b ни аниқлаштирамиз. Кўпҳад даражаси 7 ёки $2^3 - 1$, иш сабабли $k = 3$. Шундан келиб чиқадики, $j=2^2= 4$ b нинг аҳамиятини шундай танлаймизки, ҳар икки q ва r кўпҳади унимодал бўлсин. Бунинг учун P даги a_{j-1} коэффициентни саралаш керак ва $b = a_{j-1}-1$ ни қўйиш керак. Бизнинг ҳолатда бу шуни англатадики, $b=a_3-1=5$. Энди биз q ва r аҳамиятни топишни истаймиз, бу қуйидаги тенгламани қондириши керак:

$$x^7+4x^6 – 8x^4 + 6x^3 + 9x^2 + 2x - 3 = (x^4 + 5)q(x) + r(x)$$

q ва r кўпҳадларни g ни $x^4 + 5$ бўлишда ҳосил бўлган алоҳида қолдиққа мос келади, қолдиқларни бўлиш эса шуни беради:

$$P(x) = (x^4 + 5)(x^3 + 4x^2 + 0x + 8) + (x^3 – 11x^2 + 2x - 37)$$

Кейинги қадамда биз худди шу процедурани ҳар бир q ва r кўпҳадларга қўллашимиз мумкин:

$$q(x) = (x^2 - 1)(x + 4)+(x+12), \quad r(x)=(x^2+1)(x-11)+(x -26)$$

Натижада эса қуйидагига эга бўламиз:

$$p(x) = (x^4 + 5)((x^2 - 1)(x + 4) + (x + 12)) + ((x^2 + 1)(x - 11) + (x - 26))$$

Ушбу кўпҳадга қараб кўрамизда, x^2 ни ҳисоблаш учун битта кўпайтириш керак; яна бир кўпайтириш $x = x^2 \cdot x^2$ ни ҳисоблаш учун зарур. Бу икки кўпайтиришлардан ташқари тенгламанинг ўнг томонини ҳисоблашда яна 3 та кўпайтириш иштирок этади.

Яна 10 та қўшиш операциялаш таъкидланади. 1 – жадвалда ушбу ҳисоблаш усули ва бошқа усулларнинг таққослов натижалари берилган.

Ушбу процедурани диққат билан ўрганиб умумий мурракаблик формуласини чиқариш мумкин. Аввало шуни кўрамизки, тенгламада битта кўпайтириш ва иккита қўшиш қатнашмоқда. Шу сабабли $M=M(к)$ кўпайтириши ва $A = A(к)$ қўшиш учун қуйидаги рекурент солиштирмалар боғланишини оламиз:

$M(1)= 0$ $\qquad$ $A(1)= 0$

$M(к)=2M(к-1)+1$ при $к >1$ $\qquad$ $A(k)= 2A(k - 1) + 2$ при $к > 1$.

Кўринишлар	**Кўпайтириш**	**Қўшиш**
Стандарт	13	13
Горнер схемаси	7	7
Дастлабки қайта ишлаш	5	5

Матрицаларни кўпайтириш.

Матрица – математик объект бўлиб, иккилик массивга эквивалентдир. Матрицада сонлар қатор ва устунларда жойлашади. Иккита бир хил матрицани элементар даражада қўшиш ёки айириш мумкин. Агар биринчи матрица устуни сонлари иккинчи матрица қатори сонлари билан тўғри келса, бу матрицаларни кўпайтириш мумкин. Натижада, матрица қатори биринчи

матрица қатори ва устуни эса иккинчи матрица устунига тенг бўлади. 3 х 4 ҳажмдаги матрицани 4 х 7 ҳажмдаги матрицага кўпайтирса, у ҳолда биз 3 х 7 ҳажмдаги матрицага эга бўламиз.

Матрицаларни кўпайтириш: иккала кўриниш яъни АВ ва ВА, иккита бир хил ҳажмдаги квадрат матрицани айириш мумкин, аммо умуман олганда жавоблар бир – биридан фарқли бўлади. Икки матрицани кўпайтириш ёки айириш учун ҳар бир биринчи қатор кўпҳадлари ва иккинчи ҳар бир матрица устунига кўпайтирилади. Кейин бундай кўпайтма йиғиндиси ҳисобланади ва натижа тўғри келган катакка ёзилади. Расмда икки матрицани кўпайтириш келтирилган (1- расм).

Матрицани кўпайтириш 24 марта кўпайтириш 16 марта қўшишни талаб қилади. Умуман стандарт ҳажмдаги *a x b* матрицани а х с матрицага кўпайтирилса, ab кўпайтириш ва a(b-1) қўшиш бажарилади.

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} \qquad \begin{pmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \end{pmatrix}$$

2 х 3 – матрицани на 3 х 4 – матрицага кўпайтириш.

G ҳажмдаги а х b матрицани Н расмдаги b х с матрицани стандарт алгоритми кўпайтмаси қуйидаги кўринишда бўлади. Натижа эса а х с ҳажмдаги R матрицага ёзилади.

for i=1 to a do
for j=1 to c do
R[i,j]=0 for k=l to b do
*R[i,j>Rti.J]+G[i.k]*H[k,j] end for k end for j end for i*

Биринчи қарашда бу минимал ҳажмдаги иш, икки матрицани қайта кўпайтириш зарур. Аммо олимлар буни минимал иш эканлигини исботлаша олмади ва натижада улар бошқа алгоритмларни топишди ва улар жуда қулай бўлган кўпайтириш матрицаларидир.

Матрицаларни Виноград усулида кўпайтириш

Агар икки матрица кўпайтмаларининг натижасига қарасак, ҳар бир элемент унга аҳамиятли бўлган скаляр кўпайтмага муносиб қатор ва устун матрицаларга асос қилиб олинади. Биз ишнинг олдиндан бажарилишига имкон берувчи дастлабки қайта ишлаш кўпайтмасини сезишимиз мумкин.

Икки векторни қараймиз:

V = (v_1,v_2,v_3,v_4) и W=(w_1,w_2,w_3,w_4)

Уларниг скаляр кўпайтмаси

$\mathbf{V} \bullet \mathbf{W} = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4$ га тенг.

Бу тенгликни қуйидаги кўринишда ёзишимиз мумкин:

$\mathbf{V} \bullet \mathbf{W} = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_4)$

$— v_1v_2 — v_3v_4 — w_1w_2 — w_3w_4.$

Китобхон охирги икки ифоданинг эквивалентлигига осонгина ишонади. Балки иккинчи ифода биринчи ифодага қараганда кўпроқ ишни талаб қилади: тўртта кўпайтириш ўрнига биз олтита эканлигини, 3та қўшилувчи ўрнига эса 10 тасини кўрамиз. Бу ифоданинг ўнг қисми охиридаги тенглик дастлабки қайта ишлаш имконини бергани аниқ эмас: унинг қисмларини ҳар бир матрица қаторининг ва иккинчи матрицанинг ҳар бир устунини олдиндан ҳисоблаш ва эслаб қолиш мумкин. *G* матрицанинг *a* х *b* ҳажмдаги Н матрицанинг *b* х с ҳажмига кўпайтириш учун Виноградниг тўлиқ алгоритми шундай кўринишга эга. Натижа *a* х с ҳажмдаги *R* матрицага ёзилади.

```
d=b/2
// ҳисоблаш  rowFactors для G
for i=l to a do
rowFactor[i]=G[i,1]*G[i,2]
for j=2 to d do
rowFactor[i]=rowFactor[i]+G[i,2j-1]*G[i,2j]
end for j end for i
// ҳисоблаш columnFactors для H for i=l to c do
columnFactor[i]=H[1,i]*H[2,i]
for j=2 to d do
columnFactor[i]=columnFactor[i]+H[2j-1,i] *H[2j,i]
end for j end for i
// матрицаларни ҳисоблаш R for i=l to a do for j=l to c do
R[i,j]=-rowFactor[i]-columnFactor[j J for k=l to d do
R[i,j>RCi»JJ + (Gti,2k-l]+H[2k,j])*(G[i,2k]+H[2k-l,j]) end for к ena^udry ~ end for i
// умумий ҳажмдаги тоқ кўпҳадларни қўшиш if (2*(Ъ/2) /= Ъ)
 then for i=l to a do for j=l to c do
R[i,j]-RCi,j]+G[i,b]*H[b,j]
end for j end for i end if
```

Виноград алгоритмининг анализи

Умумий рост ҳажмдаги *b* ҳолатни жадвал кўринишида кўриб чиқамиз.

	Кўпайтириш	**Қўшиш**
G матрицани дастлабки қайта ишлаш	*ad*	a(d-1)
H матрицани дастлабки қайта ишлаш	*cd*	c(d-1)
R матрицани ҳисоблаш	*acd*	*ac(2d+d+1)*
Умумий	*(abc+ab+bc)/2*	*(a(b*-2)+*c(b*-2) +ac(3b + 2))/2

Матрицаларини Штрассен усулида кўпайтириш.

Штрассен алгоритми эквивалентлари матрицалар билан ишлайди. Умуман олганда бу жуда фойдали. Штрассен алгоритмининг 2 х 2 матрицасини кўпайтириш учун 7 та формуладан фойдаланамиз. Бу формулалар фавқулотда кўрилмаган ва, афсуски, Штрассен алгоритми манбаларида унга қай йўл билан келганлиги ҳам тушунтирилмаган.

Матрица элементларининг кўпайтмаси коммутив бўлиши учун унинг формулаларини кўриш ёки ишлатиш талаб қилинмайди. Бунинг маъноси шуки, бъзида бу элементлар матрица бўлиши мумкин, бу дегани Штрассен алоритмини рекурсив равишда қўллашимиз мумкин.

Бу эса Штрассен формуласи:

$x_1 = (G_{1,1} + G_{2,2})(H_{1,1}+H_{2,2});$ $x_5 = (G_{1,1} + G_{2,2})H_{2,2};$

$x2 = (G_{2,1} + G_{2,2})H_{1,1};$ $x_6 = (G_{2,1} - G_{1,1})(H_{1,1} + H_{1,2});$

$x_3 = G_{1,1}(H_{1,2} - H_{2,2});$ $x_7 = (G_{2,1} - C_{2,2})(H_{2,1}+H_{2,2}).$

$x_4 = G_{2,2}(H_{2,1} — H_{1,1});$

Энди R матрица элементлари формула бўлади:

$R_{1,1} = x_1+x_4-x_5-x_7;$ $R_{1,2} = x_3 + x_5;$

$R_{2,1}=x_2+x_4;$ $R_{2,2} = x_1 + x_3-x_2+x_6.$

2 х 2 матрицани кўпайтмасида алгоритм 7 та кўпайтирув ва 18 та қўшиш амалини бажаради. Умумий кўринишининг анализи, кўпайтиришлар сони иккита марта қайта кўпайтириш *N* х Л/" – тахминан $N^{2.81}$ қўйишлар сони эса $6N^{2.81}-6N^2$ кўринишга эга.

Иккита 16 х 16 матрицанинг кўпайтмасида Штрассен алгоритми 1677 кўпайтирув ва қўшимча 9138 та қўшиш амалларини тежайди.

Уччала натижа бирлаштирилса, қуйидаги натижа ҳосил бўлади:

	Кўпайтириш	Қўшиш
Стандарт алгоритми	N^3	N^3-N^2
Виноград алгоритми	$\frac{N^3+2N^2}{2}$	$\frac{3N^3+4N^2-4N}{2}$
Штрассен алгоритми	$N^{2.81}$	$6N^{2.81}$-6 N^2

Штрассен алгоритми амалиётда кам қўлланилади: уни ишлатишда эҳтиёткорлик талаб қилинади. Унинг муҳим томони шундаки, у матрицаларни кўпайтиришда $0(ЛГ^3)$ кам операция талаб қилинадиган биринчи алгоритмдир. Матрицаларни кўпайтиришнинг самарадорлигини ошириш ва унинг камчиликлари устида ишлаш ишлари давом эттирилмоқда.

Чизиқли тенгламани ечиш

Чизиқли тенгламалар системаси N ноъмалумли N та тенгламадан ташкил топади. Системанинг умумий кўриниши қуйидаги кўринишга эга:

$$a_{11}x_1+a_{12}x_2+a_{13}x_3+\ldots+a_{1N}x_N=b_1$$
$$a_{21}x_1+a_{22}x_2+a_{23}x_3+\ldots+a_{2N}x_N=b_2$$
$$\vdots$$
$$a_{N1}x_1+a_{N2}x_2+a_{N3}x_3+\ldots+a_{NN}x_N=b_N$$

Бундай системаларнинг келиб чиқиши турлича бўлиши мумкин, бироқ одатда константалар (a коэффициент деб кўрсатилган) тенгламанинг ўнг тарафида келтирилган b қиймати берилган кўрсатмалар қурилмалари ўлчамларининг баъзи қийматлари ҳисобланади. Бизни ўлчамларнинг қандай қийматларида бундай натижаларга эришилиши қизиқтиради. Қуйидаги мисолни кўриб чиқамиз:

$$2x_1+7x_2+1x_3+5x_4=70$$
$$1x_1+5x_2+3x_3+2x_4=45$$
$$3x_1+2x_2+5x_3+1x_4=33$$
$$8x_1+1x_2+5x_3+3x_4=56.$$

Чизиқли тенгламалар системасини ечиш усулларидан бири ўрин алмаштиришни бажаришдан иборат. Масалан, иккинчи тенгламани $x_1=45-5x_2-3x_3-2x_4$ кўринишида қайта ёзиш мумкин, сўнгра ўнг тарафдаги ифодани x_1 ўрнига бошқа учта тенгламага алмаштириш мумкин. Натижада биз учта ноъмалумли учта тенгламани оламиз. Кейин қолган тенгламалардан бирида x_2 ноъмалум билан худди шуни бажариш мумкин, шунда иккита ноъмалумли иккита тенглама ҳосил бўлади. x_3 ўзгарувчи билан яна бир ўзгартириш ва биз бир ноъмалумли битта тенглама ҳосил қиламиз (яъни x_4). Бу тенгламани x_4

ноъмалум қийматини олиш орқали ечиш мумкин, олинган ечимни аввалги босқичдаги икки тенгламадан бирига алмаштирсак, x_3 қийматини топишимиз мумкин. Топилган x_3 ва x_4 қийматларини учта тенгламадан бирига алмаштиргандан сўнг, биз x_2 ни топишимиз мумкин ва ниҳоят, дастлабки тенгмалардан ҳар бири x_1 қийматини беради.

Бундай амал жуда яхши ишлайди, бироқ унинг бажарилишида жуда кўп алгебраик ўзгаришлар амалга оширилади, бунда эса хатолик юз бериши мумкин. Тенгламалар ва ноъмалумлар сони ошганда бу ўзгаришларнинг бажарилишига кетадиган вақт тез ўсади, уларни дастурлаш эса осон эмас. Бироқ айнан улар Гаусс-Жордан усулининг асосини ташкил қилади. Биз ҳозир шу усулни ёритиб берамиз.

Гаусс-Жордан усули

Чизиқли тенгламалар системасини N қаторли ва N+1 устунли матрица деб тасаввур қилиш мумкин. Аввалги мисолда матрица қуйидаги кўринишга эга:

$$\begin{pmatrix} 2 & 7 & 1 & 5 & 70 \\ 1 & 5 & 3 & 2 & 45 \\ 3 & 2 & 4 & 1 & 33 \\ 8 & 1 & 5 & 3 & 56 \end{pmatrix}$$

Энди бу матрицанинг қаторлари билан талаб этилувчи натижаларга олиб келувчи баъзи ўзгаришларни бажариш мумкин. Агар биринчи N устунлар ва матрица қаторлари ягона матрицани ҳосил қилса, охирги устун х изланувчи қийматдан ташкил топади:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & x_1 \\ 0 & 1 & 0 & 0 & x_2 \\ 0 & 0 & 1 & 0 & x_3 \\ 0 & 0 & 0 & 1 & x_4 \end{pmatrix}$$

Ҳаракатлар режасининг асосида қуйидаги ўзгариш ётади: биринчи қаторни унинг биринчи элементига ажратиш мумкин, сўнгра биринчи қатордаги карали сонларни кейинги қаторлардан чиқариш мумкин.

Бизнинг мисолда иккинчи қатордан биринчисини чиқариш, учинчисидан – учланган биринчини, тўртинчисидан – 8 га кўпайтирилган

биринчи қаторни чиқариш мумкин. Натижада биринчи устун керакли кўринишга эга бўлади. Янги матрица қуйидаги кўринишга эга:

$$\begin{pmatrix} 1 & 3.5 & 0.5 & 2.5 & 35 \\ 0 & 1.5 & 2.5 & -0.5 & 10 \\ 0 & -8.5 & 2.5 & -6.5 & -72 \\ 0 & -27 & 1 & -17 & -224 \end{pmatrix}$$

Энди бу ҳаракатларни иккинчи қатор учун такрорлаймиз. Бу қаторни иккинчи элементнинг қийматига (яъни 1,5 га) бўлганимиздан кейин биринчи, учинчи ва тўртинчи қаторларнинг иккинчи элементлари қийматини Янги иккинчи қаторни чиқаришдан олдин нечага кўпайтириш кераклиги аниқланади. Янги матрица қуйидаги кўринишга эга:

$$\begin{pmatrix} 1 & 0 & -5.33 & 3.66 & 11.7 \\ 0 & 1 & 1.6 & -0.33 & 6.67 \\ 0 & 0 & 16.7 & -9.3 & 15.3 \\ 0 & 0 & 46 & -26 & -44 \end{pmatrix}$$

Бу жараённи учинчи қатор учун такрорлаймиз ва керакли учинчи устунни, сўнгра тўртинчи устунни ҳосил қиламиз:

$$\begin{pmatrix} 1 & 0 & 0 & 0.68 & 6.76 \\ 0 & 1 & 1 & 0.6 & 8.2 \\ 0 & 0 & 0 & -0.56 & -0.96 \\ 0 & 0 & 0 & -0.24 & -1.68 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 & 3 \\ 0 & 0 & 0 & 1 & 7 \end{pmatrix}$$

Изланаётган ноъмалумлар қийматларини олдик: x_1=2, x_2=4, x_3=3 ва x_4=7. компьютерда бу жараённи амалга ошириш учун яхлитлаш муаммосига дуч келамиз. Циклни итерациялашда яхлитлашнинг хатолари йиғилиб

боради, бунинг ҳисобига натижалар унга аниқ бўлмаслиги мумкин. Чизиқли тенгламаларнинг ката системаларини ечишда хатолар сезиларли бўлиши мумкин. Яхлитлашдаги хатоларни назорат қилишга имкон берувчи чизиқли тенгламаларни ечишнинг бошқа алгоритмлари ҳам мавжуд, бироқ бу алгоритмлар сонли таҳлил курсида ўрганилади ва биз уларга тўхталмаймиз.

Агар икки қатор даражаланган бўлса, муаммо пайдо бўлади. Бундай ҳолда ноли қатор ҳосил қиламиз ва алгоритм нолга бўлишдаги хатоликка дуч келади. Бундай ҳолат «хусусият» деб аталади; хусусиятларни қайта ишлаш бизнинг китобимизда ўрганилмайди.

Детерминалланмаган алгоритмлар

NP нима?

Аввалги бўлимларимизда кўздан кечирилган барча алгоритмлар полиномиал мураккабликка эга эди. Ундан ташқари, уларнинг ҳаммасининг мураккаблиги $O(N^3)^1$ эди, комплекс кўпайтириш алгоритми эса энг кўп вақт талаб қилувчи эди. Бироқ, асосийси, биз бу масалаларнинг аниқ ечимини идрокли вақт ичида топа олдик. Бу масалаларнинг ҳаммаси P синфига – полиномиал мураккабликка эга масалалар синфига тегишли. Бундай масалалар деярли ечиладиган ҳисобланади.

Яна бошқа масалалар синфи ҳам бор: улар деярли ҳал қилинмайди ва биз уларни идрокли вақтда еча оладиган алгоритмларни билмаймиз. Бу масалалар NP синфини ҳосил қилади, яъни детерминаллашмаган полиномиал мураккабликка эга. Бу сўзларнинг маъноси параграф охирида ойдинлашади. Фақат шуни айтиш мумкинки, бу масалаларни ечувчи барча маълум детерминаллашган алгоритмларнинг мураккаблиги ёки экспоненциал, ёки факториал. Улардан баъзиларининг мураккаблиги 2^N га тенг, бу ерда N – кирувчи рақамларнинг сони. Бу ҳолда кирувчи рақамлар рўйхатига бир элементни қўшсак, алгоритмнинг ишлаш вақти икки баробар ортади. Агар бундай вазифани ечиш учун кирувчи ўнта элементдан алгоритмга 1024 та амал талаб қилинган бўлса, кирувчи 11 та элементдан амаллар сони 2048 ни ташкил қилади. Бу вақтнинг киришни қисқа узунлаштиришдаги сезиларли ортишидир.

NP синфидаги вазифаларни характерловчи «Детерминиллашган полиномиал» сўз бирикмаси уни ечишда қуйидаги икки қадамли ёндашув ҳисобланади.

Биринчи қадамда бундай вазифанинг мавжуд ечимини асосий деб ҳисоблаб, яъни ечимни топишга ҳаракат қилувчи детерминаллашмаган алгоритм мавжуд; баъзан бундай уриниш муваффақиятли бўлиб, биз оптимал ёки оптималга яқин жавобни оламиз, баъзида эса муваффақиятсиз (оптималдан узоқ жавоб) олинади. Иккинчи қадамда биринчи қадамда

олинган жавоб дастлабки вазифанинг ечими эканлиги текширилди. Бу қадамларнинг ҳар бири алиҳида полиномиал вақтни талаб қилади. Муаммо шундаки, изланган ечимни олиш учун бу қадамларни неча марта такрорлашимиз кераклигини биз билмаймиз. Иккала қадам полиномиал бўлса ҳам, уларга мурожаат экспенциал ёки фактириал бўлиши мумкин.

Коммивояжер масаласи NP синфига тегишли. Бизга шаҳарлар тўплами ва улар орасидаги саёҳат «нархи» берилган. Шундай тартибни аниқлаш керакки, ҳамма шаҳарларга ташриф буюриб, дастлабки шаҳарга қайтиб келганда, саёҳатнинг умумий нархи минимал бўлсин. Бу вазифани, масалан, шахар кўчаларидаги ахлатни самарали йиғиштириш тартибини аниқлаш ёки компьютер тармоғининг ҳамма бўғимларида ахборотни тез тарқатиш йўлини танлашда қўллаш мумкин. Саккизта шаҳарни 40.320 мавжуд усул билан соддалаштириш мумикн, ўнта шаҳар учун бу сон 3628800 га ортади. Қисқа йўлни танлаш ушбу ҳамма имкониятларни кўриб чиқишни талаб қилади. Фараз қилайлик, бизда кўрсатилган тартибда 15 та шаҳар орқали саёҳат баҳосини ҳисобловчи алгоритм бор. Агар бир сонияда бундай алгоритм ўзидан 100 та вариант ўтказса, ҳамма имкониятлар ва қисқа йўлни топиш учун унга тўрт аср керак бўлади. Агар бизнинг хизматимизда 400 та компьютер бўлса ҳам, барибир бунга бир йил кетади, биз атига 15 та шаҳарни назарда тутяпмиз. 20 та шаҳар учун қисқа йўлни топиш учун миллиард компьютерлар 9 ой ичида параллел равишда ишлаши лозим. Компьютерлар оптимал ечимни топишини кутгандан кўра қандай бўлса ҳам саёҳат қилиш тезроқ ва арзонроқ эканлиги аниқ кўриниб турибди.

Уларнинг барчасини кўздан кечирмай қисқа йўлни топса бўладими? Барча йўлларни кўриб чиқувчи алгоритмни ўйлаб топиш ҳали ҳеч кимнинг қўлидан келмаган. Шаҳарлар сони кам бўлса, вазифа осон ҳал бўлади, бироқ бу ҳар доим шундай бўлади дегани эмас, бизни умумий вазифанинг ечими қизиқтиради.

Коммивояжер ҳақидаги масала биз 6-бобда кўриб чиққан дастур масалаларига ўхшаш. Ҳар бир шаҳарни графнинг чўққиси, шаҳарлар орасидаги йўлни қирралар, улар орасидаги саёҳат баҳосини шу қирранинг оғирлиги деб тасаввур қилиши мумкин. Бундан шундай хулоса чиқариш мумкинки, қисқа йўлни қидириш алгоритми коммивояжер масаласини ҳал қилади, бироқ бу ундай эмас. Коммивояжер ҳақидаги масаланинг қандай икки шарти уни қисқа йўл ҳақидаги масаладан фарқлайди? Биринчидан биз ҳамма шаҳарларга ташриф буюришимиз керак, қисқа йўлни излаш алгоритми эса берилган икки шаҳарлар орасидаги йўлни беради. Агар қисқа йўлни танлаш алгоритми берган қисқа бўлакли йўлни йиғсак, бу йўл баъзи

шаҳарлардан бир неча марта ўтади. Иккинчи фарқ қисқа йўлни излашда дастлабки нуқтага қайтишни талаб қилишдан иборат.

Бу масала NP синфига тегишли эканлигини кўрсатиш учун уни юқорида ёритилган икки қадамли амал ёрдамида қандай ечиш мумкинлигини тушуниш зарур. Коммивояжер ҳақидаги масаланинг биричи қадамида шаҳарларнинг баъзи тартиблаштирилиши асосий ўринга чиқади. Бу детерминаллашмаган жараён бўлганлиги учун ҳар сафар янги тартиб пайдо бўлади. Генерация жараёнини полиномиал вақтда амалга ошириш мумкин: биз шаҳарлар рўйхатини сақлашимиз мумкин, тасодифий рақамни асосийлаштиришимиз, шу номдаги шаҳарни рўйхатдан танлашимиз ва у Яна пайдо бўлмаслиги учун рўйхатдан чиқаришимиз мумкин. Бундай жараён O(N) ичида амалга оширилади, бу ерда N – шаҳарлар сони. Иккинчи қадамда берилган тартибдаги шаҳарлар бўйича саёҳатлар баҳосини ҳисоблаш амалга оширилади. Бунинг учун биз рўйхатдаги шаҳарларнинг кетма-кет жуфтлари орасидаги саёҳат баҳосини жамлашимиз керак, бунинг учун ҳам O(N) жараёни талаб қилинади. Икала қадам ҳам полиномиал, чунки коммивояжер масаласи NP синфига тегишли.

Бу ерда шуни таъкидлаш лозимки, бундай икки қадамли амални биз аввал кўриб чиққан масалаларнинг ихтиёрийсига қўллаш мумкин. Масалан, рўйхатни саралашни дастлабки рўйхат элементларининг эркин тартибини асосийлаштириб, бу тартиб ўсувчи эканлигини текширган ҳолда бажариш мумкин. Саралаш масаласининг бундай талқини NP синфига тегишли эмасми? Албатта, тегишли. P ва NP синфлари орасидаги фарқ шундаки, биринчи ҳолда бизда полиномиал вақтда масалани ечувчи детерминаллашган алгоритм бўлса, иккинчи ҳолда биз бундай алгоритмни билмаймиз. Бу масалага биз 11.3 § да қайтамиз.

Масалани бошқа масалага келтириш

Масалани ечишнинг усулларидан Яна бири бир масалани иккинчисига келтириш ёки редуциялашдир. Иккинчи масалани ечиш алгоритмини шундай ўзгартириш керакки, у биринчисини еча олсин. Агар ўзгартириш ва иккинчи масала полиномиал вақтда бажарилса, бизнинг янги масаламиз ҳам полиномиал вақтда ечилади.

Мулоҳазамизни мисол орқали тушунтирамиз. Биринчи масала шундан иборат бўлсинки, агар берилган Буль ўзгарувчилари «рост» қийматига эга бўлса, «ҳа» ифодасини қайтариш ва тескари ҳолда «йўқ» ифодасини қайтариш бўлсин. Иккинчи масала бутун сонлар рўйхатидан максимал қийматни топиш бўлсин. Уларнинг ҳар бири оддий аниқ ечимни талаб қилади, лекин фараз қилайлик, биз максимумни излашдаги масала ечимини биламиз, Буль ўзгарувчиси ҳақидаги масалани ечишни билмаймиз. Биз Буль

ўзгарувчилари ҳақидаги масалани бутун сонларнинг максимуми ҳақидаги масалага келтирмоқчимиз. «Ёлғон» ифодасига 0 сонини, «рост» ифодасига 1 сонини таққословчи бутун сонлар рўйхатига Буль ўзгарувчилари қийматининг ўзгаришлари алгоритмини ёзамиз. Сўнгра рўйхатдаги максимал элементни излаш алгоритмидан фойдаланамиз. Рўйхат қандай тузилганлигига қараб, бу максимал элемент ёки 0, ёки 1 бўлиши мумкинлигини аниқлаймиз. Бундай жавобни Буль ўзгарувчиси ҳақидаги жавобга ўзгартириб, агар максимал қиймат 1 га тенг бўлса «ҳа», 0 га тенг бўлса «йўқ» ни қайтариш мумкин.

1-бобда кўрганимиздек, максимал қийматни излаш чизиқли вақт ичида амалга ошади, биринчи масалани иккинчисига редукциялаш ҳам чизиқли вақтни талаб қилади, шунинг учун Буль ўзгарувчиси ҳақидаги масалани ҳам чизиқли вақтда ечиш мумкин.

NP масалалари ҳақида баъзи нарсаларни билиш учун кейинги бўлимда маълумотлар техникасидан фойдаланамиз.

NP – тўла масалалар

NP синфини муҳокама қилишда уларни ечиш ката вақт талаб қилиши ҳақидаги фикримизга кўра, уларни ечишда самарали алгоритмларни топа олмаганлигимизни назарда тутишимиз керак. балки коммивояжер масаласига бошқа нуқтаи назардан қараб, биз уни ечишнинг полиномиал алгоритмини ишлаб чиққан бўлар эдик. Кейинги параграфда ўрганадиган бошқа масалалар ҳақида ҳам шуни айтиш мумкин.

NP тўла атамаси NP синфидаги мураккаб масалаларга киради. Бу масалалар шуниси Билан ажралиб турадики, агар биз улардан бирининг полиномиал алгоритм ечимини топсак, NP синфидаги ҳамма масалалар полиномиал алгоритм ечимига рухсат беришини билдиради. Биз NP масалани тўла деб ҳисоблаймиз ва NP синфидаги бошқа барча масалаларни унга келтириш йўлларини кўрсатамиз. Амалиётда бу фаолият унчалик ваҳимали эмас – ҳар бир NP масаласи учун редукцияни амалга ошириш зарурияти йўқ. А масаланинг баъзи NP нинг NP- тўлалигини исботлаш учун унга бирор NP – тўла В масалани келтириш етарли. В ва А масалани редукциялаб, ҳар бир NP масала А га икки қадамда келтирилиши мумкин, улардан бири – унинг В га редукцияси.

Аввалги бўлимда биз полиномиал алгоритмнинг редукциясини бажарган эдик. Энди NP масалани ечувчи алгоритм редукциясига қараймиз. Бизга масаланинг ҳамма таркибий қисмларини бошқа масаланинг эквивалент таркибий қисмларига ўзгартирувчи амал керак бўлади. Бундай ўзгартириш ахборотни сақлаши керак: ҳар сафар, биринчи масаланинг ечими ижобий жавоб берса, бундай жавоб иккинчи масалада ҳам бўлиши керак ва аксинча.

Графдаги Гамильтон йўл деб ҳар бир чўққидан аниқ бир марта ўтувчи йўлга айтилади. Агар бунда йўл дастлабки чўққига қайтиб келса, у Гамильтон цикли деб аталади. Гамильтон йўл ёки циклга эга граф тўла бўлиши шарт эмас. Гамильтон циклини излаш масаласи коммивояжер масаласига қуйидагича келтирилади. Графнинг ҳар бир чўққиси – бу шаҳар. Графнинг ҳар бир қирраси орқали ўтувчи йўлнинг баҳоси 1 га тенг деймиз. Қирра Билан боғланмаган икки шаҳар орасидаги йўлнинг баҳоси 2 га тенг деймиз. Энди коммивояжер масаласига мос келувчи масалани ечамиз. Агар графда Гамильтон цикли бўлса, коммивояжер масаласи ечимининг алгоритми 1 оғирлик қиррасидан иборат цикли йўлини топади. Агар Гамильтон цикли бўлмаса, топилган йўлда 2 оғирликдаги 1 қирра бўлади. Агар графда N чўққи бўлса, унда Гамильтон цикли мавжуд, агар топилган йўлнинг узунлиги N га тенг бўлса ва бундай цикл бўлмаса, топилган йўлнинг узунлиги N дан ката бўлади.

1971 йилда Кун конъюктив нормал форма ҳақида масаланинг NP – тўлалигини исботлади, бу масала кейинги параграфда муҳокама қилинади. Катта сонли масалаларнинг NP – тўлалиги конъюктив нормал форма масаласини уларга редукция қилиш йўли Билан исботланди. 1979 йилда Чоп этилган Гэри ва Джонсонннинг китобида NP – тўлалиги исботланган юзлаб масалалар келтирилган.

Редукция шундай қудратли нарсаки, агар NP – тўла масалалардан бирортасини P синфли масалаларга келтира олсак, барча NP масалалар полиномиал ечимга эга бўлади. Бугунги кунгача бундай келтиршни юзага чиқарувчи уринишларнинг ҳеч бири амалга ошмаган.

Типик NP масалалар

Биз муҳокама қиладиган масалаларнинг ҳар бири ёки оптималлаштирилган, ёки ечимни қабул қилиш масаласи ҳисобланади. Оптималлаштириш масаласининг мақсади, одатда, минимал ёки максимал қийматга эга аниқ натижа ҳисобланади. Ечимни қабул қилиш масаласида,одатда, чегарали қиймат берилади. Бизни кўрсатилган чегарадан ката (максималлаштириш масалаларида) ёки кичик (минималлаштириш масалаларида) ечим мавжудлиги қизиқтиради. Оптималлаштириш масалаларида олинган аниқ натижа жавоб бўлиб хизмат қилади, ечимни қабул қилиш масалаларида – «ҳа» ёки «йўқ» жавоби бўлади.

11.1 § да биз коммивояжер масаласининг оптималлашган варианти билан шуғулланган эдик. Бу минималлаштириш масаласи эди ва бизни минимал нарх йўли қизиқтирган эди. Ечимни қабул қилиш вариантида биз берилган C константадан кичик баҳоли коммивояжер йўли мавжудлигини сўрашимиз мумкин эди. Ечимни қабул қилиш масаласидаги жавоб танланган

чегарага боғлиқлиги аниқ. Агар бу чегара жуда ката бўлса (масалан, у ҳамма йўлларнинг жамланма баҳосидан ортса), «ҳа» жавобини олиш қийин эмас. Агар бу чегара жуда кичик бўлса (масалан, у ихтиёрий икки шаҳар орасидаги йўллар баҳосидан кичик бўлса), «йўқ» жавоби ҳам тез олинади. Бошқа оралиқ ҳолларда жавобни излаш вақти жуда узоқ ва у оптималлашган масалани ечиш вақтига тенг. Шунинг учун биз оптималлашган ва ечимни қабул қилиш масалалари ҳақида навбатма-навбат гапириб, жорий мақсадларимизга жавоб берадиган масаладан фойдаланамиз.

Кейинги бир нечта бўлимларда биз ҳам оптималлашган вариантда, ҳам ечимни қабул қилиш вариантида яна олтита NP масалани кўриб чиқамиз.

Графни бўяш

6-бобда айтиб ўтганимиздек, $G = (V, E)$ графи чўққиларни жуфтлаб боғловчи E қирралар тўплами, V тугунлар ёки чўққилар тўпламидан иборат. Бу ерда биз фақат йўналтирилган графлар Билан шуғулланамиз. Графнинг чўққиларини бутун сонлар билан белгиланадиган турли рангларга бўяш мумкин. Бизни ҳар бир қирранинг учи турли рангларга бўялиши қизиқтиради. N чўққили графда турли N рангларга бўяш мумкин, лекин бунда бўёқни кам миқдорда ишлатиш мумкинми? Оптималлаштириш масаласида бизни граф чўққисини бўяш учун зарур бўладиган бўёқларнинг минимал миқдори қизиқтиради. Ечимни қабул қилиш масаласида бизни чўққиларни с ёки камроқ бўёқларга бўяш мумкинлиги қизиқтиради.

Графни бўяш масаласида амалий иловалар бор. Агар графнинг ҳар бир чўққиси коллежда ўқиладиган курсни билдирса ва чўққилар қирралар билан боғланса, агар иккала курсни танловчи талаба бўлса, ғоят мураккаб граф ҳосил бўлади.

Агар ҳар бир талаба 5 та курс тингласа, бир талабага 10 та қирра тўғри келади. 35000 талабага 500 курс тўғри келади, деб фараз қилайлик. У ҳолда ҳосил бўлган графда 500 чўққи ва 35000 қирра бўлади. Агар имтиҳонларга 20 кун ажратилган бўлса, талабага бир кунда иккита имтиҳон тўғри келмаслиги учун граф чўққиларини 20 та ранга бўяш керак бўлади.

Имтиҳонлар жадвалини тишлаб чиқиш графларни бўяшга эквивалентдар. Аммо графларни бўяш масаласи NP синфига тегишли, шунинг учун жадвални идрокли вақтда тўғри ишлаб чиқиш мумкин эмас. Бундан ташқари имтиҳонларни режалаштирганда, одатда, талабаларда бир кунда иккитадан ортиқ имтиҳон бўлмаслиги кераклиги талаб этилади, курснинг турли бўлимлари бўйича имтиҳонлар эса бир кунга белгиланади. Кўриниб турибдики, имтиҳонларнинг «мукаммал» режасини ишлаб чиқиш мумкин эмас, шунинг учун қулай режаларни ишлаб чиқиш учун бошқа техника зарур. Яқинлаштирилган алгоритмлар 9-бобда ёритилган.

Қутиларга жойлаштириш

Бизда бир қанча бирлик хотира қутилари ва $s_1, s_2, \ldots\ldots, s_N$ турли ўлчамли объектлар тўплами мавжуд бўлсин. Оптималлаштириш масаласида бизни барча объектларни жойлаштириш учун зарур бўлган қутиларнинг энг кам сони, ечимни қабул қилиш масаласида эса барча объектларни В ёки оз қутиларга жойлаш мумкинлиги қизиқтиради.

Бу масала ахборотни дискка ёзиш, компьютернинг бўлак хотирасига жойлаштиришда, кемага юкни самарали тақсимлашда, мижозларнинг буюртмаси бўйича материалнинг стандарт қисмларидан бўлакларга кесишда пайдо бўлади. Агар, масалан, бизда катта метил листлар ва буюртма рўйхатидаги кичик листлар бўлса, буюртмаларни иложи борича зичроқ тақсимлашни хоҳлаймиз.

Рюкзакни жойлаштириш

Бизда $s_1, s_2, \ldots\ldots, s_N$ ҳажмли, $\omega_1, \ldots, \omega_N$ баҳоли объектлар тўплами бор. Биз оптималлаштириш маласида биз К ўлчамли рюкзакни шундай жойлаштиршимиз керакки, унинг баҳоси максимал бўлсин. Ечимни қабул қилиш масаласида бизни жойлаштирилган объектларнинг жамланма баҳоси W бўлишига эришиш мумкинлиги қизиқтиради.

Бу масала пул қўйиш стратегияларини танлашда пайдо бўлади: турли пул қўйишлар бу ерда ҳажм, кўзланган фойданинг миқдори – баҳо, рюкзакнинг ҳажми режалаштирлган капитал қўйилмалар ҳажми билан белгиланади.

Тўпламости элементлар йиғиндиси масаласи

Бизда $s_1, \ldots, s_N$ турли ўлчамли кўп объектлар ва бирор L мусбат юқори чегара бўлсин. Оптималлаштириш масаласида биз шундай объектлар тўпламини топишимиз керакки, улар ўлчамининг йиғиндиси L га яқин бўлиб, бу юқори чегарадан ошмасин. Ечимни қабул қилиш масаласида объектлар тўплами L ўлчамлари йиғиндиси билан мавжудлик ўрнатилиши керак. Бу рюкзакни жойлаштириш масаласининг соддалаштирилган кўринишидир.

КНФ – ифодасининг ҳаққий масаласи

Конъюктив нормал форма (КНФ) ANP (^ билан белгиланади) операторлар билан боғланган Буль ифодаларининг кетма-кетлигидан иборат, ҳар бир ифода OR (V орқали белгиланади) операторлар билан боғлиқ Буль ўзгарувчилари ёки уларнинг инкоридан иборат ҳисобланади. Қуйида конъюктив нормал форма (инкор ўзгарувчининг тепасига чизиқча қўйиш билан белгиланади) да ифодаланган Буль ифодасига мисол берилган:

$$(a \vee b) \wedge (\bar{a} \vee c) \wedge (a \vee b \vee \acute{c} \vee d) \wedge (b \vee \acute{c} \vee d) \wedge (a \vee b \vee c \vee \eth \vee e).$$

Конъюктив нормал формадаги Буль ифодасининг ҳақиқийлиги масаласи ечимни қабул қилиш вариантида қўйилади: ифодага кирувчи

ўзгарувчиларда барча ифодаларни ҳақиқий қилувчи ҳақиқий қийматлар мавжудми. Ўзгарувчилар сони ҳам, ифодаларнинг мураккаблиги ҳам чекланмаган, шунинг учун қийматларнинг ҳақиқийлик комбинациялари сони жуда ката бўлиши мумкин.

Ишларни режалаштириш масаласи

Бизда ишлар тўплами бўлсин ва бизга уларнинг ҳар бирининг якунланиши учун $t_1,t_2,\ldots,t_N$ зарур вақт, бу ишлар албатта тугаши керак бўлган d_1 ,d_2,...,d_n муддат ҳамда жарима маълум бўлсин. Оптималлаштириш масаласида солинадиган жарималарни минималлаштирувчи ишлар тартиби ўрнатилиши талаб қилинади. Ечимни қабул қилиш масаласида жариманинг миқдори P дан ката бўлмаган ишлар тартиби мавжудлиги сўралади.

NP синфига қандай масалалар тегишли?

Биз P синфидаги кўпгина масалалар ва NP синфидаги бир қанча масалаларни кўриб чиқдик. Масала NP синфига тегишли бўлади, агар у полиномиал вақтда детерминалланмаган алгоритм билан ечилса. Аввал айтилганидек, саралаш жараёнини қуйидагича амалга ошириш мумкин:

1) рўйхатдаги элементларни тасодифий ҳолда чиқариш;
2) 1 дан N-1 гача барча i учун $s_i<s_{i+1}$ эканлигини текшириш.

Бу детерминаллашмаган икки босқичли жараён. Биринчи босқич таққослашни талаб қилмайди ва уни N қадамда бажариш мумкин – рўйхатнинг чиқувчи элементига бир қадамдан. Иккинчи босқич ҳам полиномиал: уни бажариш учун N-1 солиштиришни бажариш лозим. Бундай амал NP синфини аниқлашимиз учун тўғри келади ва шундай хулосага келиш мумкинки, саралаш масаласи ҳам P синфига, ҳам NP синфига тегишли. Худди шуни ҳар қандай полиномиал алгоритмга қўллаш мумкин, шунинг учун P синфидаги ҳамма масалалр NP синфида ҳам бор, яъни P NP нинг тўпламостиси ҳисобланади. Бироқ NP синфида шундай масалалар борки, биз улар ечимининг полиномиал детерминаллашган алгоритмини билмаймиз.

Бу фарқнинг моҳияти NP масалаларда кўриб чиқишимиз зарур бўлган вариантларнинг ката сони ҳисобланади. Бироқ бу сон кирувчи қийматлар комбинациясининг сонидан бир оз ортади. Бизда 30 та турли элемент ёки шаҳарлардан иборат рўйхат бўлиши мумкин, фақат 30 тадан биттаси шу рўйхатда ўсувчи бўлади ёки қисқа йўлни кўрсатади. Фарқи шундаки, рўйхатни полиномиал алгоритм билан тартиблаштириш мумкин – улардан зиларига бо-йўғи 150 та таққослаш кера бўлади. Пуфаксимон саралаш алгоритмида биринчи ўтишда рўйхат бўйича энг ката элемент, 1/30 мавжуд комбинациялардан 29 тасини олиб ташлаб, ўз ўрнига туради. Иккинчи ўтишда 28 та таққослаш қолган имкониятларнинг 1/29 тасини олиб ташлайди. Диққат билан кузатилса, олиб ташланган имкониятлар сони ката

бўлиши мумкинлигини кўриш мумкин: ҳар бир ўтиш натижаси рўйхатдаги энг ката элементнинг ўрнига силжиши эмас, балки бошқа элементларнинг қайта тартибланиши ҳам бўлиши мумкин.

Бизга маълум энг қисқа йўлни излаш усули барча мавжуд йўл вариантларини кўриб чиқиш ва уларнинг узунлигини солиштиришдан иборат. Бизда вариантларнинг маълум қисмини муваффақиятли олиб ташлашга имкон берувчи алгоритм йўқ. Шунинг учун биз уларнинг ҳаммасини кўриб чиқишимизга тўғри келади. Ҳатто секундига 30 дан 1000000000 йўлни кўриб чиқиш учун 840 миллиарддан ортиқ аср керак бўлади. Икки шаҳар орасидаги қимматбаҳо йўлни ташлаб юбориш мумкиндек. Лекин ҳатто бу оддий фикр ҳам ўтмайди: ташлаб юборилган йўлнинг жамланма баҳоси юқори бўлган иккита йўл билан алмаштириш мумкин, лекин ҳеч қандай яхшиланиш юз бермайди. Графдаги қиррани ташлаб юборишимизмумкинлигини текшириш бизни мураккаб алгоритмга қайтаради. Биз оптималга яқин жавобни олиш йўлларини кўриб чиқамиз.

Ечим қабул қилиш масаласида бирор P қиймат берилади, умумий жарима P дан ошмайдиган иш тартиби мавжудлигини аниқлашимиз керак. оптималлашган масалада бизни умумий йиғиндининг минимал қиймати қизиқтиради. Биз ечим қабул қилиш масаласи билан шуғулланамиз: агар тасдиқ жавобини олгунимизча P ўсувчи чегарали Ушбу масаланинг ечим алгоритмини бир неча марта чақирсак, оптималлаштириш масаласини ҳам ечган бўламиз. Бошқача айтганда, 0 жаримали иш тартиби мавжудлигини сўралади. Агар бундай тартиб бўлмаса, 1 жаримасига ўтилади ва тасдиқ жавобини олмагунча жарима миқдори оширилади. Кейинги алгоритм ишни режалаштириш масаласининг бўлиши мумкин бўлган ечимини босқич билан солиштиради:

PenaltyLess(list,N,limit)
list ишнинг тартибланган руйхати
N ишнинг умумий сони
limit предельная величина штрафа
currentTime=0
currentPenalty=0
currentJob=l
while(currentJob<=N)and(currentPenalty<=limit)do
currentTime=currentTime+list[currentJob.time
if (list[currentJob].deadline<currentTime(then
currentPenalty=currentPenalty+list[currentJob].penalty end if
current Job=currentJob+1 end while
if currentPenalty<=limit then

return да else
return нет end if

Масаланинг NP синфига тегишли эканлиги полиномиал вақт ичида берилган ечимни текширишимизни талаб қилади. Кўрсатилган алгоритм ҳақиқатдан умумий жаримани ҳисоблайди. вақт мураккаблиги таҳлили шуни кўрсатадики, *while* цикли N дан ортиқ ўтиш содир қилмайди; агар *currentPenalty* ўзгарувчи охирги қиймати ошмаса, максималликка эришилади. Ҳамма жараёнларни ҳисобга олиш таққослашларнинг умумий сони 3N+1 га, қўшишлар сони 3N га тенглигини хулоса қилишга имкон беради. Бу эса алгоритм мураккаблиги O (N) га тенглиги, у полиномиал ва N синфига мос келишини билдиради.

Графни бўяш

Графни бўяш масаласида граф чегарасини бир неча рангларга бутун сон билан рақамланган бўяшнинг шундай йўлини топиш керакки, ҳар қайси икки қўшни чўққилар турли рангларга бўялсин. Қарор қабул қилиш шундан иборатки, кўрсатилган талабларга риоя қилган ҳолда чўққиларни с ёки ундан кам рангга бўяш, оптималлаштириш эса – рангларнинг зарур минимал сонини аниқлаш.

Детерминалланмаган босқичда мумкин бўлган чўққилар рўйхати ва уларга берилган ранглар пайдо бўлади. Айнан шу босқичда нечта ранг ишлатилиши ҳал қиланади, шунинг учун назорат босқичи танланган ранглар сонига жавоб бермайди. Ечимни танлашда детерминалланмаган босқич С ранглар билан қаноатланади. Оптималлаштириш масаласида у рангларнинг сонини кетма-кет камайтириб, талаб этилувчи бўяш давом этгунча рангларнинг ката сонидан бошлаши мумкин. Графни х ранга бўяш мумкин эмаслигига ишонч ҳосил қилгач, биз х+1 рангга бўяш минимал даражада мумкин деб ҳисоблаймиз.

Қуйидаги алгоритм ҳосил бўлган бўяшни амалга ошириш мумкинлигини текширади. Бўялаётган граф туташувлар қуймаси кўринишида берилган, бу рўйхатнинг graph [j] элементи графнинг j чўққисини кўрсатади, бу элементнинг **graph [j] .edgeCount** майдони ундан чиқадиган қирралар сонини, **graph [j] .edge** майдони j чўққисига қўшни бўлган чўққилар сақланувчи массивни ҳисобланади.

boolean ValidColoring(graph,N,colors)
graph //туташувлар рўйхати
N //графдаги чўққилар сони
colors //ҳар бир чўққига унинг рангини солиштирувчи массив
for j=1 to N do
for k=1 to graph[j].edgeCount do

if (colors[j]=colors[graph[j].edge[k]]) then
return //йўқ
end if end for k end for j return //ҳа

Кўриниб турибдики, бу алгоритм бўяшнинг мос келишини тўғри текширади. У ҳамма чўққилардан ўтади ва агар чўққи бошқа шу рангдаги чўққи билан бевосита боғлиқ бўлса, алгоритм тўхтайди ва нотўғри жавобни қайтаради. Агар қўшни чўққиларнинг ҳам жуфтликлари турли рангларга бўялган бўлса, жавоб тўғри бўлади. Бу алгоритмнинг вақт бўйича мураккаблигига келсак, *for* ташқи цикл *N* ўтишни бажаради. Икки циклда Ушбу чўққидан чиқувчи ҳар бир қирра кўздан кечирилади. Шунинг учун солиштиришларнинг умумий сони қирралар сонига тенг бўлади, яъни алгоритмнинг мураккаблиги *O*(*edges*) га тенг – полиномиал баҳо аниқ, чунки графдаги қирралар сони N^2 дан ошади. Шундай экан, шубҳасиз талаб бажарилади.

6–маъруза: Саралаш ва тартиб статистикаси. Пирамидал саралаш. Пирамидалар. Пирамидал саралаш хоссалари.

Режа:

1. Саралаш ва тартиб статистикаси;
2. Пирамидал саралаш;
3. Пирамидалар. Пирамидал саралаш хоссалари;

Қўшимчалар билан саралаш. Қўшимчалар билан саралашнинг асосий ғояси шундаки, янги элементни сараланган рўйхатга қўшишда уни ихтиёрий жойга эмас, балки керакли жойга жойлаштириш лозим, сўнг бутун рўйхатни бошқатдан саралаш керак. Қўшимчалар билан саралашда ихтиёрий рўйхатнинг биринчи элементи сараланган рўйхатнинг 1 узунлиги деб ҳисобланади. 2 элементли сараланган рўйхат бошланғич рўйхатдаги 2-элементни 1-элемент жойлашган рўйхатнинг керакли ўрнига жойлаштириш орқали яратилади. Энди бошланғич рўйхатнинг 3-элементини сараланган 2 элементли рўйхатга қўйиш мумкин. Бу жараён бошланғич рўйхатнинг барча элементлари кенгаювчи сараланган рўйхатга жойлашгунга қадар давом этади.

Бу жараённи ифодаловчи алгоритм қуйида келтирилган:

```
InsertionSort(list,N)
List N
For i=2 to N do
        newElement=list[i]
        location=i-1
        while(location>=1) and (list[location]>newElement) do
                list[location+1]=list[location]
                location=location-1
        end while
        list[location+1]=newElement
end for
```

Бу алгоритм янги қўйиладиган элементни newElement ўзгарувчисига юклайди. Сўнг барча катта элементларни 1 позицияга суриб, бу элементга жой ажратади. Циклнинг охирги итерацияси элементни location+1 номер билан location+2 позиция ўтказади.

Энг ёмон ҳолат таҳлили

Агар ички while сиклини кўриб чиқадиган бўлсак, яна бошқатдан қўшилаётган элемент барча элементларда кичик ва рўйхатнинг саралланган қисмида жойлашган бўлса, жараённинг катта қисми бажарилади. Бу ҳолатда location ўзгарувчисининг қиймати 0 га тенг бўлганда цикл тўхтайди. Шунинг учун алгоритм бажарилишининг асосий қисми ҳар бир янги элемент рўйхат бошига қўшилганда амалга оширилади. Бу фақатгина бошланғич рўйхат элементлари камайиш тартибда бўлганда бажарилади. Бу энг ёмон ҳодисалардан бири, лекин бошқалари ҳам бор.

Бундай рўйхатни қайта ишлашни қандай амалга ошириш кераклигини кўриб чиқамиз. Дастлаб рўйхатнинг 2-елементи қўйилади. У фақатгина битта элемент билан таққосланади. Иккинчи қўйилган элемент олдинги иккита элемент билан, учинчи қўйиладиган элемент учта элемент билан ва ҳоказо, i-қўйиладиган элемент i та элемент билан таққосланади ҳамда бу жараён N-1 марта такрорланади. Шундай қилиб, энг ёмон ҳолатдаги саралаш мураккаблиги қуйидагига тенг:

$$W(N) = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \frac{N^2 - N}{2}$$

$$W(N) = O(N^2)$$

Ўрта ҳолат таҳлили

Ўрта ҳол таҳлилини 2 босқичга ажратамиз. Дастлаб навбатдаги элемент ҳолатини аниқлаш учун керак бўладиган таққослашларнинг ўрта қийматини ҳисоблаймиз. Кейин, 1-қадам натижасидан фойдаланиб, барча керакли амалларнинг ўрта қийматини ҳисоблаш мумкин. Дастлаб, i-елементнинг жойлашиш ўрнини аниқлаш учун таққослашларнинг ўрта қийматини ҳисоблаймиз. Юқорида айтиб ўтганимиздек, рўйхатга i-елементни қўшганда у керакли жойда жойлашган бўлса ҳам, ҳеч бўлмаганда битта таққослаш бажарилади.

i-елемент учун нечта мумкин бўлган ҳолат мавжуд? Қисқа рўйхатларни кўриб чиқамиз ва ихтиёрий ҳолатда натижаларни умумлаштиришга ҳаракат қиламиз. 1-қўшиладиган элемент учун 2 та имконият бор: икки элементдан биринчиси ёки иккинчиси бўлиши мумкин. 2-қўшиладиган элементнинг 3 ҳолатдан бирида бўлиши мумкин: 1,2,3 рақамлари билан. Демак, i-қўшиладиган элемент i+1 та ҳолатдан бирини эгаллаши мумкин. Барча имкониятлар эҳтимоллик даражасини тенг деб оламиз.

Ҳар бир i+1 позицияга етиб олиш учун нечта таққослаш талаб этилади? i нинг кичик қийматларини кўриб чиқамиз ва натижани бирлаштиришга ҳаракат қиламиз. Агар 4-қўшилаётган элемент 5-позицияга тушса, у ҳолда таққослаш ёлғон бўлади. Агар унинг 4-позицияси тўғри бўлса, у ҳолда 1-

таққослаш рост ҳисобланади, иккинчиси эса – ёлғон. 3-позицсияга тушса, 1- ва 2-таққослаш рост, 3-си эса ёлғон бўлади. 2-позицияда 1-,2- ва 3-таққослаш рост ва 4-си ёлғон бўлади. 1-позицияда барча таққослашлар рост бўлади, ёлғон таққослашлар бўлмайди. Бундан эса i+1, i, i-1,…, 2 позицияларга тушувчи и-елемент учун таққослашлар мос ҳолда 1,2,3 …i эканлиги келиб чиқади. 1-позицияга тушганда эса таққослашлар сони i га тенг бўлади. i-қўйиладиган элемент учун таққослашларнинг ўрта қиймати қуйидаги тенглик билан берилади:

$$A_i = \frac{1}{i+1}(\sum_{p=1}^{i} p + i) = \frac{1}{i+1}(\frac{i(i+1)}{2} + i) = \frac{i}{2} + \frac{i}{i+1} = \frac{i}{2} + 1 - \frac{1}{i+1}$$

Биз i-елементни қўйишдаги таққослашларнинг ўрта қийматини ҳисобладик. Энди бу натижани рўйхатдаги ҳар бир N-1 элемент учун йиғиш керак:

$$A(N) = \sum_{i=1}^{N-1} A_i = \sum_{i=1}^{N-1} (\frac{i}{2} + 1 - \frac{1}{i+1})$$

$$A(N) = \sum_{i=1}^{N-1} \frac{i}{2} + \sum_{i=1}^{N-1} 1 - \sum_{i=1}^{N-1} \frac{1}{i+1}$$

$$\sum_{i=1}^{N-1} \frac{i}{2} = \sum_{i=1}^{N-1} 1 = \sum_{i=1}^{N-1} \frac{1}{i+1}$$

$$A(N) \approx \frac{1}{2}\frac{(N-1)N}{2} + (N-1) - (\ln N - 1) = \frac{N^2 - N}{4} + (N-1) - (\ln N - 1)$$

$$= \frac{N^2 + 3N - 4}{4} - (\ln N - 1) \approx \frac{N^2}{4};$$

$$A(N) = O(N^2).$$

Пуфаксимон саралаш

Энди пуфаксимон саралашга ўтамиз. Бу усулнинг асосий принципи кичик қийматларни суриш натижасида рўйхатнинг юқори қисмига ўтказиш ва шу вақтнинг ўзида катта элементларни пастга тушуришдан иборат. Пуфаксимон саралашнинг турли вариантлари бор. Бу параграфда вариантлардан бирини кўриб чиқамиз, қолганларини топшириқларда учратишингиз мумкин.

Пуфаксимон саралаш алгоритми рўйхат бўйича бир нечта ўтиш йўлларини амалга оширади. Ҳар бир ўтишда қўшни элементлар таққосланади. Агар қўшни элементларнинг тартиби нотўғри бўлса, у ҳолда уларнинг жойи алмашади. Ҳар бир ўтиш рўйхат бошидан бошланади. Дастлаб биринчи ва иккинчи элемент таққосланади, сўнг 2- ва 3-елемент, кейин 3- ва 4-елемент таққосланади ва ҳоказо; нотўғри тартибда турган элементлар жой алмашади. Биринчи ўтишда рўйхатнинг энг катта элементи топилса, у барча элементлар билан рўйхат охиригача ўрин алмашади. 2-ўтишда рўйхатнинг катталиги бўйича 2-елементи охиридан иккинчи ўринга ўтади. Бу жараённинг давом этиши натижасида ҳар бир элемент ўз ўрнини

эгаллайди. Бунда кичик қийматлар ҳам юқоридан ўз ўрнига жойлашади. Агар бирор ўтишда элементларнинг ўрин алмашиши бажарилмаса, у ҳолда барча элементлар керакли тартибда жойлашган бўлади ва алгоритмнинг бажарилиши тўхтатилади. Шуни таъкидлаш керакки,ҳар бир ўтишда бир вақтнинг ўзида бир нечта элемент ўз ўрни томон силжиб боради. Пуфаксимон саралаш алгоритми қуйида келтирилган:

```
BubbleSort(list,N)
list
N
number0fPairs=N
swappedElements=true
while swappedElements do
        number0fPairs=number0fPairs-1
        swappedElements=false
        for i=1 to number0fPairs do
                if list[i]>list[i+1]then
                        Swap(list[i],list[i+1])
                        swappedElements=true
                end if
        end for
end while
```

Яхши ҳолат таҳлили

SwappedElements байроқ изоҳи нотўғри эканлигини таъкидлаш учун яхши ҳолат таҳлилини қисқача кўриб чиқамиз. Бажарилаётган иш ҳажми қайси ҳолатда минимал бўлишини кўриб чиқамиз. For сикли 1-ўтишда тўлиқ бажарилиши керак ва шунинг учун алгоритмга камида N-1 та таққослаш талаб этилади. 2 та имкониятни кўриб чиқиш керак: 1-ўтишда ҳеч бўлмаганда битта ўрин алмаштириш ёки ўрин алмаштириш бажарилмайди. 1-ҳолатда ўрин алмаштириш SwappedElements байроғининг қиймати true га ўзгаришига олиб келади, демак while цикли яна такрорланади, бунда N-2 та таққослаш талаб этилади. 2-ҳолатда SwappedElements элемент байроғи false қийматини сақлаб қолади ва алгоритм бажарилиши тўхтатилади.

Шунинг учун яхши ҳолда N-1 та таққослаш бажарилади, бунда 1-ўтишда ўрин алмаштиришлар бўлмайди. Бундан эса маълумотларнинг яхши тўплами талаб қилинган тартибда элементлар рўйхатини ташкил этиши келиб чиқади.

Ёмон ҳолат таҳлили

Яхши ҳолда рўйхатдаги элементлар талаб қилинган тарибда жойлашар экан, элементларнинг тескари тартибда жойлашиши ёмон ҳолни бермасмикан, деган савол туғилади. Агар энг катта элемент 1-ўринда турса, у ҳолда у рўйхат охиригача қолган барча элементлар билан ёнма-ён қўйиб борилади.1-ўтишдан олдин катта қийматдаги 2-елемент 2-ҳолатни эгаллайди, бироқ 1-таққослаш ва 1-ўрин алмаштириш натижасида у 1-ўринга ўтказилади. 2-ўтишнинг бошида 1-ҳолатда катта қийматдаги 2-елемент жойлашган бўлади ва у рўйхатнинг охиридан 2-елементгача қолган элементлар билан ёнма-ён ўрин алмашиб боради. Бу жараён барча қолган элементлар учун бажарилади, шунинг учун for цикли N-1 марта такрорланади. Шунинг учун берилганларнинг тескари тартиби ёмон ҳолга олиб келади.

Ёмон ҳолда нечта таққослаш амалга оширилади? 1-ўтишда қўшни элементларнинг N-1 та таққослаши, 2-ўтишда эса N-2 та таққослаш бажарилади. Тадқиқотлар шуни кўрсатадики, ҳар бир навбатдаги ўтишда таққослашлар сони биттага камаяди. Шунинг учун ёмон ҳол мураккаблиги қуйидаги формула билан берилади:

$$W(N) = \sum_{i=N-1}^{1} i = \sum_{i=1}^{N-1} i = \frac{(N-1)N}{2} = \frac{N^2 - N}{2} \approx \frac{1}{2} N^2 = O(N^2).$$

Ўрта ҳолат таҳлили

Юқорида таъкидлаганимиздек, ёмон ҳолатда for цикли N-1 марта такрорланади. Ўрта ҳолда элементларнинг ўрнини алмаштирмасдан ҳосил бўлган ўтишларнинг ихтиёрийсида эҳтимоллиги тенг деб фараз қиламиз. Ҳар бир ҳолатда нечта таққослаш бажарилишини кўриб чиқамиз. 1-ўтишдан кейин таққослаш сони N-1 г тенг бўлади. 2 та ўтишдан кейин N-1+N-2 та бўлади. биринчи і та ўтишда таққослашлар сонини C(i) деб белгилаймиз. Агар бирорта ҳам ўрин алмаштириш бажарилмаса, алгоритм ўз ишини тўхтатади. Ўрта ҳол таҳлилида бундай имкониятларни кўриб чиқиш лозим. Натижада қуйидаги тенгликка эга бўламиз:

$$A(N) = \frac{1}{N-1} \sum_{i=1}^{N-1} C(i)$$

бу ерда C(i) – for циклининг дастлабки і та ўтиш оралиғидаги таққослашлар сони. Бунга нечта таққослаш талаб этилади? C(i) нинг қиймати қуйидаги тенгликда кўрсатилган:

$$C(i) = \sum_{j=N-1}^{i} j = \sum_{j=i}^{N-1} j = \sum_{i=i}^{N-1} j - \sum_{j=i}^{i-1} j$$

ёки

$$C(i) = \frac{(N-1)N}{2} - \frac{(i-1)i}{2} = \frac{N^2 - N - i^2 + i}{2}$$

Охирги тенгликни A(N) ифодага қўйиб қуйидагини ҳосил қиламиз:

$$A(N) = \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{N^2 - N - i^2 + i}{2}$$

N і га боғлиқ бўлмаганлиги учун:

$$A(N) = \frac{1}{N-1}((N-1)\frac{N^2 - N}{2} + \sum_{i=1}^{N-1} \frac{-i^2 + i}{2})$$

$$A(N) = \frac{N^2 - N}{2} + \frac{1}{2(N-1)}(\sum_{i=1}^{N-1}(-i^2) + \sum_{i=1}^{N-1} i)$$

Натижада:

$$A(N) = \frac{N^2 - N}{2} + \frac{1}{2(N-1)}(-\frac{(N-1)N(2N-1)}{6} + \frac{(N-1)N}{2})$$

$$= \frac{N^2 - N}{2} - \frac{(2N-1)N}{12} + \frac{N}{4} = \frac{6N^2 - 6N - 2N^2 + N + 3N}{12}$$

$$= \frac{4N^2 - 2N}{12} \approx \frac{1}{3} N^2 = O(N^2)$$

Шелл саралаши

Шелл саралашини Доналд Л. Шелл ўйлаб топган. Бу саралашнинг ўзига хослиги шундаки, бутун рўйхат аралаштирилган рўйхатостиларнинг мажмуаси сифатида қаралади. 1-қадамда бу рўйхат остилар жуфт элементларни ифодалайди. 2-қадамда ҳар бир гуруҳда тўрттадан элемент мавжуд бўлади. Жараённинг такрорланиб бориши натижасида ҳар бир рўйхатостисида элементлар сони ортиб боради, рўйхат остилар сони эса, мос ҳолда камаяди. 3.1-расмда 16 элементдан иборат рўйхатни саралашда фойдаланиш мумкин бўлган рўйхат остилар тасвирланган. 3.1(а) - расмда 8 та рўйхатости тасвирланган, бунда ҳар бирида 2 тадан элемент, яъни 1-рўйхатостиси 1- ва 9-елементларни ўз ичига олади, 2- рўйхатостиси 2- ва 10-елементларни ўз ичига олади ва ҳоказо. 3.1(б)расмда ҳар бири тўрттадан элементни ўз ичига олган 4 та рўйхатостисини кўришимиз мумкин. Бунда 1-рўйхатостиси 1-, 5-, 9- ва 13-елементларни ўз ичига олади. 2- рўйхатостиси 2-, 6-, 10- ва 14-елементлардан иборат. 3.1(в)-расмда жуфт ва тоқ

элементлардан иборат 2 та рўйхатостиси тасвирланган. 3.1(г)-расмда яна битта рўйхатга қайтиб келамиз.

Рўйхат остиларни саралаш 3.1 да ўрганилган қўшимчалар билан саралашни қўллаш ёрдамида амалга оширади. Натижада биз қуйидаги алгоритмга эга бўламиз:

```
Shellsort(list,N)
...
End while
```

Increment ўзгарувчиси рўйхатости элемент номерлари орасидаги қадам катталигини ўз ичига олади. Биз алгоритмни рўйхат элементининг узунлигидан кичик бўлган, иккининг энг юқори даражасидан биттага кам бўлган қадамдан бошлаймиз. Шунинг учун 1000 элементдан иборат рўйхат 1-қадамининг қиймати 511 га тенг бўлади. Шунингдек, қадам қиймати рўйхатостилар сонига тенг бўлади. 1- рўйхатостисининг элементлари 1 ва 1+increment номерларига эга; охирги рўйхат остининг 1-елементи сифатида increment номерли элемент хизмат қилади.

While циклининг охирги бажарилишида passes ўзгарувчисининг қиймати 1 га тенг бўлади, шунинг учун InsertionSort функсиясини охирги чақирувда increment ўзгарувчисининг қиймати 1 га тенг бўлади.

Бу алгоритмнинг таҳлили ички InsertionSort алгоритмининг таҳлилига асосланади. 3.1 да ҳисоблаганимиздек, қўшимчалар билан саралашнинг ёмон ҳолатида N та элементдан иборат рўйхат $(N^2-N)/2$ та амални талаб этади, ўрта ҳолда эса N^2 элементни талаб қилади.

Алгоритм таҳлили.

Алгоритм таҳлилини InsertionSort функсиясини чақиришлар сонини ва ҳар бир чақирувда рўйхатдаги элементлар сонини ҳисоблашдан бошлаймиз. 15 узунликка эга бўлган рўйхатга мурожаат қиламиз. 1-ўтишда increment ўзгарувчисининг қиймати 7 га тенг бўлади, шунинг учун 2 узунликдаги рўйхатда 7 та чақирув амалга оширилади. 2-ўтишда increment ўзгарувчисининг қиймати 3 га тенг бўлади, шунинг учун 5 узунликдаги рўйхатда 3 та чақирув амалга оширилади. 3- ва охирги ўтишда 15 узунликдаги рўйхатга битта чақирув эълон қиламиз.

Биз биламизки, ёмон ҳолда иккита элементдан иборат рўйхатда InsertionSort алгоритми битта таққослашни амалга оширади. Ёмон ҳолда 5 та элементдан ташкил топган рўйхатда 10 та таққослаш бажарилади, 15 та элементдан иборат рўйхатда таққослашлар сони 105 га тенг. Бу сонларни йиғсак 142 ни ҳосил қиламиз (7*1+3*10+1*105). Бироқ, биз яхши натижага

эришдикми? Агар 3.1.1. бўлимдаги таҳлилга қайтсак, қўшимчалар билан саралашнинг ёмон ҳолида ҳар бир янги элемент рўйхат бошига қўйилар эди. ShellSort алгоритмининг охирги ўтишида дастлабки босқичларида саралаш бажарилган бўлса ҳам, бу ёмон ҳол саралашни амалга ошира олмайди. Балки ишнинг қолган ҳажмини ҳисоблаш учун бошқача ёндашиш зарурдир?

Саралаш алгоритми таҳлилида баъзан рўйхатдаги инверсиялар миқдорини ҳисоблаб борамиз. Инверсия – бу рўйхатдаги тартибсиз элементлар жуфтлиги. Масалан, [3,2,4,1] рўйхатда 4 та инверсия мавжуд: (3,2);(3,1);(2,1) ва (4,1). Энг кўп инверсия элементлар тескари тартибда жойлашган рўйхатда бўлади, уларнинг сони $(N^2-N)/2$ га тенг.

Алгоритмдаги ҳар бир ўрин алмаштириш инверсиялар сонини камида биттага камайтиради. Масалан, пуфаксимон саралашда тартибсиз жуфт элементлар топилса, уларнинг ўрин алмашиши инверсия миқдорини биттага камайтиради. Худди шундай қўшимчалар билан саралашда ҳам бу ўринли бўлади, яъни катта элемент бир позиция юқорига сурилса, сурилган ва қўйилган элемент ҳисобига инверсия йўқолади. Шунинг учун пуфаксимон ва қўшимчалар билан саралашда (иккаласида ҳам мураккаблик $O(N^2)$га тенг) ҳар қандай таққослаш битта инверсиянинг йўқолишига олиб келади.

Шелл саралашининг асосида қўшимчалар билан саралаш ётибди, шунинг учун юқоридагилар Шелл саралаши учун ҳам ўринлидек туюлади. Аммо, Шелл саралашида аралаш рўйхат остилар тартибланишини ҳисобга олсак, элементлар ўрнини алмаштириш учун чақирилган алоҳида таққослаш (биттадан кўпроқ) инверсиялар сонини битта эмас, ундан кўпроқ камайтириши мумкин. 3.1 - расмда 1-ўтишда 16 ва 14 элементлар таққосланарди, уларнинг тартиби нотўғри бўлганлиги учун ўрни алмаштирилди. 16-елементни 1-ўриндан 9-ўринга ўтказиб, 7 та инверсиядан халос бўлдик. Шелл саралашининг таҳлили мурккабликларни келтириб чиқаради, чунки юқоридаги ўрин алмаштириш 14 элементдан иборат 7 та янги инверсиянинг пайдо бўлишига олиб келди. Шунинг учун у инверсияларнинг умумий миқдорини биттага камайтирди. Агар 7- ва 4-елементларнинг ўрин алмашишига аҳамият берсак, натижа худди шундай бўлади. лекин вазият бирмунча яхшиланади. 7 та таққослашдан кейин 1-ўтишда инверсиялар сони 36 тага камайди. 2-ўтишда 19 та таққослаш бажарилади ва 24 та инверсия йўқолади. Охирги қадамда 19 та таққослаш амалга оширилиб, 10 та охирги инверсия йўқотилади. Таққослашларнинг умумий сони 62 га тенг.

Шелл саралаш алгоритмининг тўлиқ таҳлили жуда мураккаб. Ёмон ҳолда бу алгоитмнинг мураккаблиги $O(N^{3/2})$ га тенглиги исботланди. Бу

алгоритмнинг тўлиқ таҳлили Д.Кнут “Искусство программироания” 3-томда келтирилган.

Қадамнинг самарадорликка таъсири

Қадамлар кетма-кетлигини танлаш Шелл саралаш алгоритмининг муракканлигига ҳал қилувчи таъсир кўрсатиши мумкин, лекин қадамлар кетма-кетлигининг оптимал вариантини излаш керакли натижага олиб келмади. Бир нечта вариантлар ўрганиб чиқилди; қуйида изланишлар натижаси келтирилган:

Агар ўтишлар фақатгина иккита бўлса, 1-ўтишда $1{,}72\sqrt[3]{N}$ тартибли қадам ва 2-ўтишда 1, бунда $O(N^{5/3})$ саралаш мураккаблигини ҳосил қиламиз.

Бошқа қадамлар тўплами барча j лар учун $h_j = (3^j - 1)/2$ кўринишга эга бўлади, $h_j<N$. h_j нинг бу қийматлари $h_{j+1} = 3h_j + 1$ ва $h_j = 1$ тенгликлар билан қаноатлантиради, шунинг учун улардан каттасини аниқлаб, қолганларини $h_j = (h_{j+1} - 1)/3$ формула бўйича ҳисоблашимиз мумкин. Бундай қадамлар кетма-кетлиги $O(N^{3/2})$ алгоритм мурккаблигига олиб келади.

Яна бир вариант $2^i\ 3^j$ нинг рўйхат узунлигидан кичик бўлган барча қийматларини ҳисоблашни кўриб чиқади, (I ва j лар бутун сонлар); ҳисобланган қийматлар қадамлар қиймати сифатида камайиш тартибида фойдаланилади. Масалан, N=40 да қуйидаги кетма-кетлик ўринли бўлади: $36(2^2\ 3^2)\ldots\ 1(2^0\ 3^0)$. Бундай кетма-кетлик ёрдамида Шелл саралашининг мураккаблиги $O(N(\log N)^2)$ гача камаяди.

Шелл саралаши – бу умумий саралашнинг бир кўриниши, лекин параметрларни танлаш унинг самарадолигига таъсир кўрсатиши мумкин.

7–маъруза: Тезкор саралаш. Тезкор саралаш тавсифи. Массивлврни бўлаклаш. Тезкор саралаш самарадорлиги. Балансланган бўлаклаш. Режа:

1. Тезкор саралаш. Тезкор саралаш тавсифи;
2. Массивлврни бўлаклаш;
3. Тезкор саралаш самарадорлиги;
4. Балансланган бўлаклаш;

Тезкор саралаш алгоритми

Тезкор саралаш нафақат ўқитиш учун балки амалиётда ҳам кенг тарқалган саралаш алгоритми. Одатда, тезкор саралаш алгоритмида катта

ҳажмдаги қийматларни ҳисоблаш учун мўлжалланган O (n log n) мураккаблик мавжуд. Бу саралаш алгоритми деярли оддий, агар буни англаган бўлсангиз тезкор саралашни ҳам пуфакчали саралаш каби тез ёза оласиз. Бўлиш ва таққослаш стратегияси қўлланилган. Қуйида рекурция қадами тасвирланган.

1. Таянч қиймат танлаш. Биз ўрта элементни таянч қиймат сифатида оламиз, лекин у сараланган қийматларнинг ўртасидаги элементни, ҳаттоки массивда мавжуд бўлмаган сонни ҳам олиши мумкин.

2. Бўлаклаш. Шу йўсинда элементларни қайта жойлаштиради, таянчдан кичик бўлган элементларнинг барчаси таянчдан чапга, катталари ўнгга ўтади. Таянчга тенг бўлган элементлар унинг ҳар қайси қисмига тушиши мумкин. Ёдда тутинг массив тенг бўлмаган қисмларга бўлиниши мумкин.

3. Ҳар икки қисмни саралаш. Тезкор саралаш рекурцив ўнг ва чап қисмларнинг ҳар икки томонига қўлланиши мумкин.

Бўлаклаш алгоритми қисмларда

Иккита индекс бор i ва j ҳудди шу дастлабки бўлаклаш бошланишида i массивнинг биринчи элементини, j эса охирги элементини кўрсатади. Сўнгра алгоритм i ни таянчга тенг ёки ундан катта қийматли элемент топулгунча олдинга суради. j индекс эса таянчга тенг ёки кичик қийматли элемент топулгунча орқага силжийди. Агар $i \leq j$ бўлса кейин улар ўрин алмашишади ва i кейинги позицияга (i + 1) га j эса (j - 1) га ўтади. Алгоритм $i > j$ шарт бажарилганда тўхтайди. Бўлаклашдан кейин i-нчи элементлар таянчдан кичик ёки тенг j-нчидан кейингилари эса таянчдан катта ёки тенг.

Мисол: {1,12,5,26,7,14,3,7,2}ни тезкор сараланг.

Сараланмаган массив

Танланган сон қиймати **7**

$12 \geq 7 \geq 2$, 12 билан 2 нинг ўрни алмаштирамиз.

$26 \geq 7 \geq 7$, 26 билан 7 нинг ўрни алмаштирамиз.

$7 \geq 7 \geq 3$, 7 билан 3 нинг ўрни алмаштирамиз.

i > j, бўлаклаш тўхтатилади.

Бўлакларга ажратилади ва алоҳида бўлаклар рекурцияга жўнатилади

...

Сараланган массив

Ёдда тутинг биз бу ерда фақат битта рекурция қадами кўрсатдик, мисолимиз чўзилиб кетмаслиги учун. Лекин, аслида, {1,5,7,3} ва {14,7,26,12} лар кейин рекурцив сараланган.

Бу нимага ишлайди?

Бўлаклаш қадамида алгоритм массивни икки қисмга бўлади, чап қисмдаги ҳар бир a_i элемент, ўнг қисмдаги ҳар бир b_j элементдан кичик ёки тенг. Шунингдек a_i ва b_j **$a_i \leq$ Танлаб олинган сон $\leq b_j$ тенгсизликни қаноатлантиради. Рекурцияга мурожатлар тугагандан кейин ҳар икки қисм сараланган бўлади ва ҳисоб аргументларини олиб юқоридагидек ифода этилади, тўлиқ массив сараланди.**

Мураккаблик анализи

Тезкор саралаш алгоритми O(n log n) мураккабликдаги алгоритмларга киради. Энг ёмон ҳолатда, тезкор саралаш $O(n^2)$ марта ишлайди, лекин кўп амалий ахборотларда бу яхши ишлайди ва O(n log n) саралаш алгоритмини қўллайди.

Кодлар қисми

Бўлаклаш алгоритми дастурлаш тилларида дастурини ёзса бўладиган алгоритмлар сарасига киради, шунинг учун у алоҳида функция сифатида ишлатилиши мумкин. Бу код C++ тезкор саралаш учун мустаҳкам функцияни ўз ичига олади.

```
void quickSort(int arr[], int left, int right)
{
      int i = left, j = right, tmp;
      int pivot = arr[(left + right) / 2];
      /* partition */
```

```
    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;
        if (i <= j) {
            tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            i++; j--;
        }
    };
    /* recursion */
    if (left < j) quickSort(arr, left, j);
    if (i < right) quickSort(arr, i, right);
}
```

Ўрнига қўйиш саралаш алгоритми

Ўрнига қўйиш алгоритми ҳам $O(n^2)$ саралаш алгоритмларига киради. Бошқа квадратик мураккабликдаги саралаш алгоритмларидан фарқли ўлароқ, бу саралаш амалда кичик маълумотли массивларни саралаш учун қўлланилади. Масалан, тест саралаш йўналишини яхшилаш учун ишлатилади. Айрим манбаларга кўра одамлар бундай алгоритмларни рақамларни сарфлаш учун ишлатади.

Ўрнига қўйиш алгоритми ҳам танлаш саралаш алгоритмига ўхшаш. Массив ҳаёлан иккига: сараланган ва сараланмаган қисмларга бўлинади. Бошланишида, сараланган қисми массивнинг биринчи элементини олади, сараланмаган қисми эса қолган элементларни олади. Ҳар қадамда, алгоритм биринчи сараланмаган қисмнинг биринчи элементини олади ва сараланган қисмнинг керакли жойига қўяди. Сараланмаган қисм бош бўлиб қолганда алгоритм тўхтайди. Формаллаганда, ўрнига қўйиш алгоритми қадамлари бунга ўхшаш:

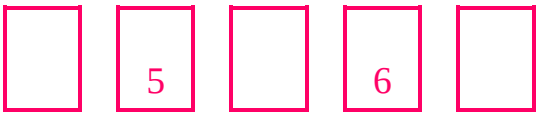
Сараланган қисми		Сараланмаган қисми	
≤X	X<	X	…

Жорий элементни ўрнига қўйиш жараёни формалли:

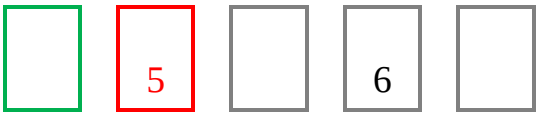
Сараланган қисми			Сараланмаган қисми
≤X	X	X<	...

Ўрнига қўйиш алгоритмини тушунарлироқ қилиш учун учун мисол кўрайлик.

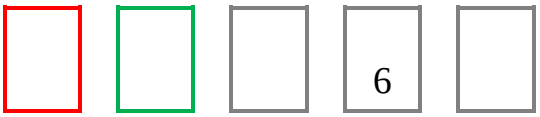
Мисол: {7,-5,2,16,4} ни ўрнига қўйиш усулидан фойдаланиб сараланг.



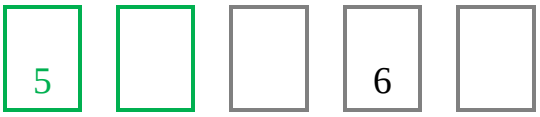
Сараланмаган



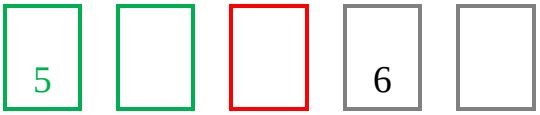
-5 ни жойлаштириш



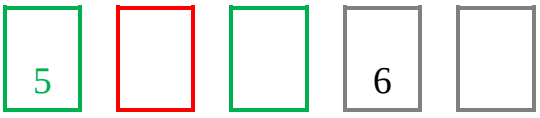
7 > -5, -5 орқага сурилади.



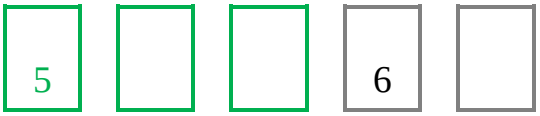
ҳосил бўлган массив



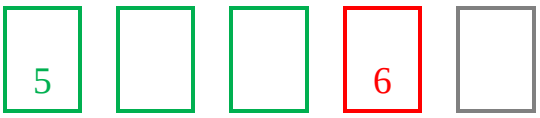
2 ни жойлаштириш



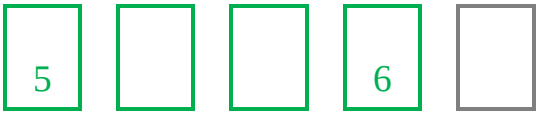
7 > 2, 2 орқага сурилади.



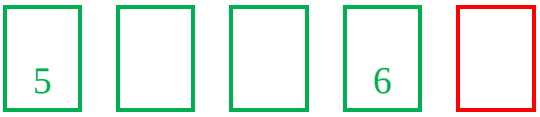
ҳосил бўлган массив



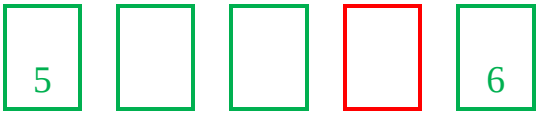
16 ни жойлаштириш



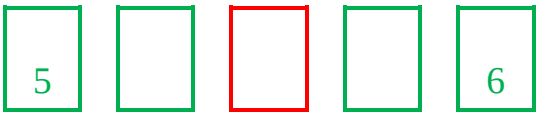
7 < 16, 16 ўрнида



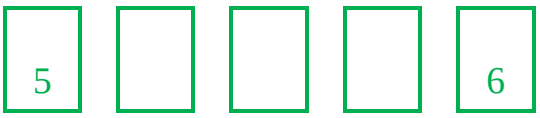
4 ни жойлаштириш



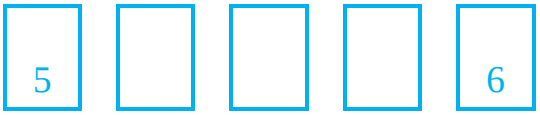
16 > 4, 4 орқага сурилади.



7 > 4, 4 орқага сурилади.



2 < 4, 4 ўрнида



Сараланган массив

Бинар қидирувдан фойдаланиш

Жойлаштиришга тўғри жойни топиш учун бинар қидирувдан фойдаланиш маъқул. Ўринлаштиришнинг бу усули бинар ўринлаштириш дейилади. Жойлаштириш учун позиция топилгандан кейин, алгоритм суради ва элементни жойлаштиради. Бу усулда паст сонли саралаш мавжуд, лекин умуман $O(n^2)$ оддий мураккаблик сақланади. Кўринишнинг амалий қисмидан бу муҳит жуда ҳам муҳим эмас, чунки ўрнига қўйиб саралаш бироз кичик маълумот гуруҳлари билан ишлайди.

Мураккаблиги

Ўрнига қўйишнинг умумий мураккаблиги $O(n^2)$ одатий стандартда, ўрнига қўйиш методига қарамасдан. O(n) гача ўрнига қўйиш қўлланилиб, деярли сараланган массивда ўрнига қўйиш яхшироқ бажарилиш намойиш этади. Ўринлаштиришлар сони $O(n^2)$ одатда, лекин таққослашлар сони, ўринлаштириш алгоритмига боғлиқ фарқ қилиши мумкин. Ўринлаштиришлар сони ўрин алмаштириш (swap) ёки суриш (shift) ишлатилганда $O(n^2)$ та, бинар ўринлаштириш (binary insertion) ишлатилганда O(n log n) та бўлади.

Ўрнига қўйиш саралаш алгоритмининг хусусиятлари

✓ Мослашувчан (дастлабки элементлар сафига мос бажарилади);
✓ Варқарор (бирхил элементларни нисбий жойини сақлаб қолади);
✓ Ички жойда (қўшимча бош жойни ўзгармас қийматини талаб қилади);
✓ Алоқавий боғлиқлик (online) (янги элементлар саралаш давомида ҳам қўшилиши мумкин);

```
void insertionSort(int arr[], int length)
{
     int i, j, tmp;
     for (i = 1; i < length; i++) {
          j = i;
          while (j > 0 && arr[j - 1] > arr[j])
          {
               tmp = arr[j];
               arr[j] = arr[j - 1];
```

```
                arr[j - 1] = tmp;
                j--;
            }
        }
}
```

Танлаб саралаш алгоритми

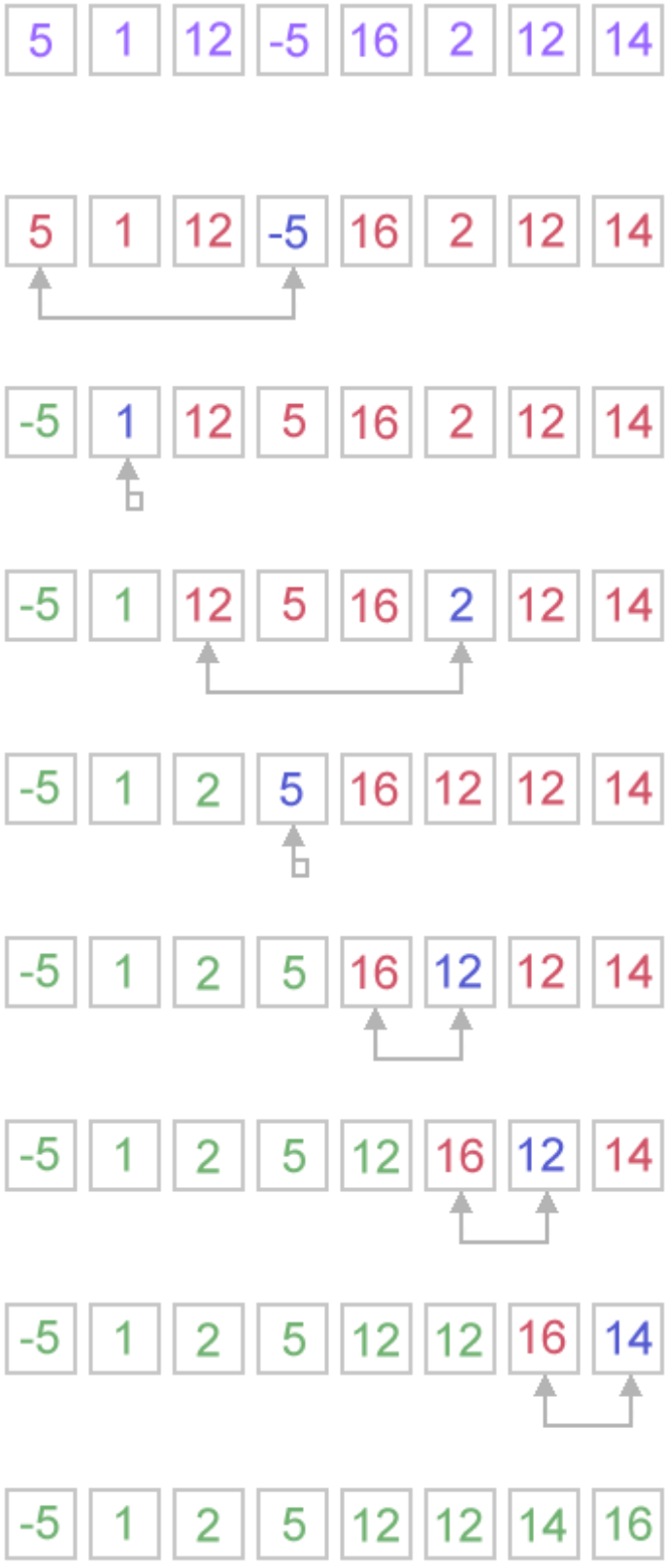
Танлаб саралаш катта ҳажмли қийматларни саралашда кам унум бўлган $O(n^2)$ саралаш алгоритмларининг бири. Танлаб саралаш ўзининг дастурий соддалиги билан аҳамиятли ва у айрим ҳолатдаги бошқа саралашларни ҳам бажара олади.

Алгортимнинг ғояси анча оддий. Массив ҳаёлда иккига бўлинади саралангани ва сараланмагани. Дастлаб, саралганган қисми бўш бўлади, сараланмаган қисми эса тўлиқ массивни ўз ичига олади. Сараланмаган қисми бўш бўлиб қолганда алгоритм тўхтайди.

Алгоритм массивни саралаётганда, сараланмаган қисмнинг биринчи элемти билан минимал элементни ўрнини алмаштиради ва бу сараланган қисмга киритилади. Танлаш алгоритми бу бажаруви барқарор эмас. Боғланган лист сараланган ҳолатда ва ўрин алмаштиришдан кўра, минимал элемент сараланмаган қисмга боғланганда танлаш алгоритми барқарор бўларди.

Мисол:

{5, 1, 12, -5, 16, 2, 12, 14} ни танлаб саралашдан фойдаланиб сараланг.



Агар сараланмаган қисм бўш бўлса, танлаб саралаш тўхтатилади. Билганимиздек, ҳар бир қадамда сараланмаган қисмнинг элементлари 1 га камаяди. Шунинг учун танлаб саралаш ташқи ҳалқадан, тўхташдан олдин n та (n-массив элементлари сони) қадам бажаради. Ташқи ҳалқанинг ҳар – бир қадами сараланмаган қисмдан минимумини топишни талаб қилади. n + (n - 1) + (n - 2) + ... + 1, ни ҳисоблаш $O(n^2)$ натижани, таққослашлар сонини беради.

Ўрин алмашишлар сони нолдан (саралланган массивда) n-1 гача (массив тескари сафда саралланган ҳолатда) ўзгаради. Умумий алгоритмнинг мураккаблиги $O(n^2)$ ни ташкил этади.

Танлаб саралаш кўпи билан n-1 та ўрин алмаштиришларни талаб қилиши шундайки, бу уни ёзиш операцияси ўқиш операциясидан анча қимматлироқ бўлган ҳолларда уни самарали қилади.

```
void selectionSort(int arr[], int n)
{
     int i, j, minIndex, tmp;
     for (i = 0; i < n - 1; i++)
     {
          minIndex = i;
          for (j = i + 1; j < n; j++)
               if (arr[j] < arr[minIndex])
                    minIndex = j;
          if (minIndex != i)
          {
               tmp = arr[i];
               arr[i] = arr[minIndex];
               arr[minIndex] = tmp;
          }
     }
}
```

8–маъруза: Чизиқли вақтда саралаш. Саралаш алгоритмининг қуйи баҳоси.

Режа:

1. Чизиқли вақтда саралаш;
2. Саралаш алгоритмининг қуйи баҳоси;

Саралаш – бу берилган тўплам элементларини бирор бир тартибда жойлаштириш жараёнидир. Саралашни мақсади тартибланган тўпламда керакли элементни топишни осонлаштиришдан иборат. Саралаш дастурларни трансляция қилинаётганда, маълумотлар мажмуасини ташқи хотирада ташкил қилинаётганда, кутубхоналар, каталоглар, маълумотлар базаси яратилаётганда тадбиқ қилинади. Маълумки, саралашнинг турли ҳил алгоритмлари мавжуд. Сабаби, битта масалани саралаш учун жуда кўплаб турли ҳил алгоритмлардан фойдаланиш мумкин. Берилган масалани ҳал қилишда баъзилари мукаммал бўлиши мумкин. Шунинг учун саралаш масаласида алгоритмларни қиёсий таҳлилини ўтказиш зарурати пайдо бўлади.

Саралаш масаласини қўйилишини қуйидагича ёзиш мумкин.

Фараз қилайлик, a_1, a_2 ,..., a_n, элементлар кетма-кетлиги берилган бўлсин. У ҳолда саралаш алгоритми элементларни массивга шундай жойлаштирадики, натижада улар қандайдир муносабатга нисбатан $f(a_{к1}) \leq f(a_{к2}) \leq \ldots \leq f(a_{кп})$ тартибга эга бўлади. Одатда f тартиблаш функцияси қандайдир махсус қоида билан ҳисобланмасдан, балки элементни калит қиймати бўйича массив элементлари тартибланади.

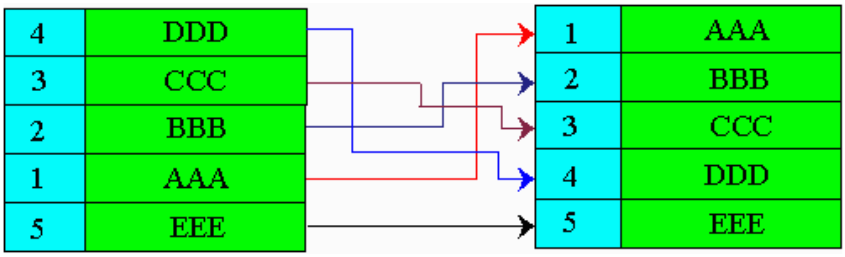
Маълумотларга қайта ишлов берилаётганда маълумотни информацион майдонини ҳамда уни машинда жойлашишини (адресини) билиш зарур.

Саралашни иккита тури мавжуд: ички ва ташқи:

– ички саралаш бу оператив хотирадаги саралаш;

– ташқи саралаш – ташқи хотирада саралаш.

Саралаш бу маълумотларни калитлари бўйича хотирада регуляр кўринишда жойлаштиришдир. Регулярлик деганда маълумотлар калит қийматлари бўйича массивда бошидан охиригача ўсиши ёки камайиши тушинилади.



Агар сараланаётган ёзувлар хотирада катта хажмни эгалласа, у ҳолда уларни алмаштиришлар катта сарф (вақт ва хотира маъносида) талаб қилади. Ушбу сарфни камайтиши мақсадида, саралаш калитлар адреси жадвалида амалга оширилади. Бунда фақатгина маълумот кўрсаткичлари алмаштирилиб, массив ўз жойида қолади. Юқоридаги усул адреслар жадвалини саралаш усули дейилади.

Сараланаётганда бир ҳил калитлар учраши мумкин, бу ҳолда сараланагандан кейин бир ҳил калитлилар бошланғич тартибда қандай жойлашган бўлса, ушбу тартибда қолдирилиши мақсадга мувофиқ бўлади (Бир ҳил калитлилар ўзларига нисбатан). Бундай усулга турғун саралаш дейилади.

Саралаш самарадорлигини бир неча мезонлар бўйича баҳолаш мумкин:

– саралашга кетган вақт;

– саралаш учун талаб қилинган оператив хотира;

– дастурни ишлаб чиқишга кетган вақт.

Биринчи мезонни қараб чиқайлик. Саралаш бажарилганда таққослашлар ёки алмаштиришлар сони ҳисоблаш мумкин.

Фараз қилайлик, $H = 0{,}01n^2 + 10n$ – таққослашлар сони. Агар $n < 1000$ бўлса, у ҳолда иккинчи қўшилувчи катта, акс ҳолда яъни, $n > 1000$ бўлса, биринчи қўшилувчи катта бўлади.

Демак, кичкина n ларда таққослашлар сони n га тенг бўлади, катта n ларда эса n^2 га тенг бўлади.

Саралашда таққослашлар сони қуйидаги оралиқларда бўлади:

$0(n \log n)$ дан $0 (n^2)$ гача; 0 (n) – идеал ҳолатда.

Саралашни қуйидагича усуллари бор:

– қатъий (тўғридан-тўғри) усуллар;

– яхшиланган усуллар.

Қатъий усуллар:

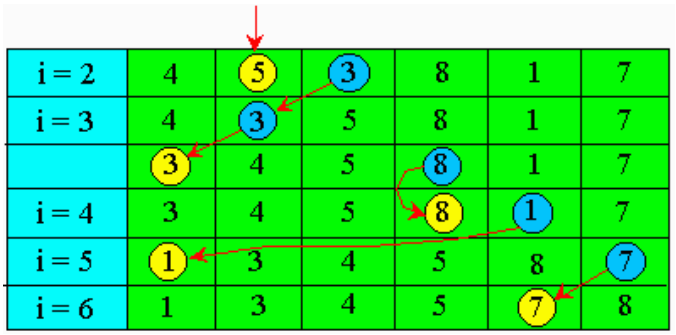
1. тўғридан-тўғри қўшиш усули;
2. тўғридан-тўғри танлаш усули;
3. тўғридан-тўғри алмаштириш усули.

Юқорида келтирилган учала усулда ҳам алмаштиришлар сони деярли бир ҳил бўлади.

Тўғридан-тўғри қўшиш усули билан саралаш

Бундай усул карта ўйинида кенг қўлланилади. Элементлар (картлар) ҳаёлан “тайёр” a(1),...,a(i-1) ва бошланғич кетма-кетликларга бўлинади. Ҳар бир қадамда (i=2 дан бошланиб, ҳар бир қадамда бир бирликка ошириб борилади) бошланғич кетма-кетликдан i-чи элемент ажратиб олиниб тайёр кетма-кетликнинг керакли жойига қўшилади.

Таклиф қилинаётган усулни қуйидаги мисолда кўриб чиқамиз. Фараз қилайлик, калит қиймати 4, 5, 3, 8, 1, 7 бўлган элементлар берилган бўлсин.



Керакли жойни қидириш жараёнини қуйидаги тартибда олиб бориш қулай бўлади. Таққослашлар амалга ошириш мобайнида, навбатдаги a(j) элемент билан солиштирилади, кейин эса x бўш жойга қўйилади ёки a(j) ўнга сурилади ва жараён чапга “кетади”. Шуни эътиборга олиш лозимки, саралаш жараёни қуйидаги шартларни бирортаси бажарилганда якунланади:

1. x элементи калитидан кичик калитли a(j) элемент топилди.
2. тайёр кетма-кетликнинг чап томони охирига етиб борилди.

Таклиф этилаётган усул алгоритми қуйидагича бўлади:

```
Procedure StraightInsertion
 Var
   i,j:index; x:item;
 begin
   for i:=2 to n do
     x:=a[i]; a[0]:=x; j:=1;
     while x<a[j-1] do a[j]:=a[j-1]; j:=j-1; end;
     a[j]:=x
     end
 end StraightInsertion
```

алгоритм самарадорлиги

Фараз қилайлик, таққослашлар сони C, ўринлаштиришлар сони M бўлсин. Агар массив элементлари камайиш тартибида бўлса, у ҳолда тақққослашлар сони энг катта бўлиб, у $C_{\max} = \frac{n(n-1)}{2}$ га тенг бўлади, яъни $O(n^2)$. Ўринлаштиришлар сони эса $M_{\max} = C_{\max} + 3(n-1)$ га тенг бўлади, яъни $O(n^2)$. Агар берилган массив ўсиш тартибида сараланган бўлса, у ҳолда таққослашлар ва ўринлаштиришлар сони энг кичик бўлади, яъни $C_{\min} = n-1$, $M_{\min} = 3(n-1)$.

Тўғридан-тўғри танлаш усули билан саралаш

Фараз қилайлик, $a_1, a_2, \ldots, a_n$ элементлар кетма-кетлиги берилган бўлсин. Мазкур усул қуйидаги тамойилларга асосланган:

1. Берилган элементлар ичидан энг кичик калитга эга элемент танланади.
2. Ушбу элемент бошланғич кетма-кетликдаги биринчи элемент a_1 билан ўрин алмашади.
3. Ундан кейин ушбу жараён қолган n-1 та элемент, n-2 та элемент ва хоказо, токи битта энг “катта” элемент қолгунча давом эттирилади.

Таклиф қилинаётган усул алгоритми қуйидагича бўлади:

Паскал тилидаги дастури:

```
Procedure StraightSelection
 Var
       i,j,k: index; x:item;
 begin
   for i:=1 to n-1 do
        k:=I; x:=a[i];
        for j:=i+1 to n do
       if a[j]<x then k:=j; x:=a[k]
end;
end;
a[k]:=a[i];
a[i]:=x
end;
end StraightSelection
```

Алгоритм самарадорлиги:

Таққослашлар сони $M = \frac{n}{2}(n-1) = \frac{n^2 - n}{2}$.

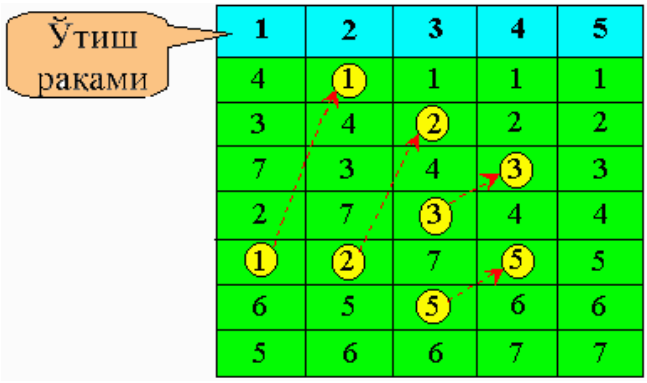
Алмаштиришлар сони $C_{min} = 3(n - 1)$, $C_{max} = 3(n - 1)\frac{n}{2}$

(n^2 тартиб).

Ушбу усул бўйича саралаш бажарилса, энг ёмон холда таққослашлар ва алмаштиришлар сони тартиби n^2 бўлади.

Тўғридан-тўғри алмаштириш усули билан саралаш (пуфаксимон)

Ушбу усулни ғояси қуйидагича: n - 1 марта массивда қуйидан юқорига қараб юриб калитлар жуфти-жуфти билан таққосланади. Агар пастки калит қиймати юқоридаги жуфти калитидан кичик бўлса, у ҳолда улар ўрни алмаштирилади.



Паскал тилидаги дастури:

```
for i := 2 to n do
  for j := n downto i do
    if a[j - 1] > a[j] then
      begin
        x := a[j - 1];
        a[j - 1] := a[j];
        a[j] := x;
      end;
```

Бизнинг холатда битта ўтиш “бекор” бўлди. Элементларни ортиқча ўринлаштирмаслик учун байроқча киритиш мумкин.

Пуфаксимон усулни яхшиланган усули бу шейкер саралаш усули бўлиб, ҳар бир ўтишдан кейин цикл ичида йўналиш ўзгартирилади.

Алгоритм самарадорлиги:

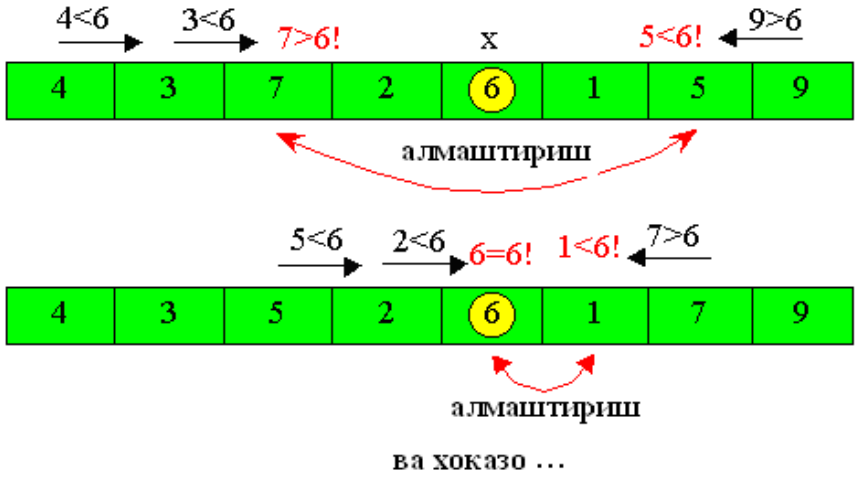
таққослашлар сони $M = \frac{n}{2} \cdot \frac{n}{2} = \frac{n^2}{4}$,

алмаштиришлар сони $C_{max} = 3\frac{n^2}{4}$.

Саралашнинг яхшиланган усуллари

Quiksort – тез саралаш усули

Ғояси: Бу усул алмаштириш усулидаги саралашга тегишли бўлиб унинг асосини калитларни танланган калитга нисбатан ажратиш ташкил қилади.



6 дан чап томонда калитлари кичик, ўнг томонда эса калитлари 6 дан катта бўлган элементлар жойлашади (юқоридаги чизма).

```
procedure Sort (L, R: integer);
begin
  i := L;
  j := r;
  x := a[(L + r) div 2];
  repeat
    while a[i] < x do
      i := i + 1;
    while a[j] > x do
      j := j - 1;
    if i <= j then
      begin
        y := a[i];
        a[i] := a[j];
        a[j] := y;
        i := i + 1;
        j := j - 1
      end;
  until i > j;
  if L < j then sort (L, j);
  if i < r then sort (i, r);
end;
```

```
procedure QuickSort;
begin
  sort (1, n);
end;
```

Алгоритм самарадорлиг:

O(n log n) – энг самарали усул.

Шелл саралаши (қисқариб борувчи қадамлар орқали саралаш)

Тўғри қўшиш усулини 1959 йилда Д. Шелл томонидан мукаммаллаштириш таклиф қилинган. Қуйидаги чизмада ушбу усул тасвирланган:

саралаш	44	55	12	42	94	18	6	67
кейин тўртлик	44	18	6	42	94	55	12	67
иккилик	6	18	12	42	44	55	94	67
яккалик	6	12	18	42	44	55	67	94

Бошида бир биридан 4 қадамда жойлашган элементлар ўзаро гуруҳланиб саралаш амалга оширилади. Бундай жараён тўртлик саралаш деб аталади. Биринчи ўтишдан кейин элементлар қайта гуруҳланиб, энди ҳар икки қадамдаги элементлар таққосланади. Бу эса иккилик саралаш деб номланади. Ва ниҳоят, учинчи ўтишда оддий ёки яккалик саралаши амалга оширилади.

Бир қарашда мазкур усул билан саралаш амалга оширилганда саралаш жараёни камайиш ўрнига ортиб борадигандек туюлсада, элементларни ўрин алмаштиришлар нисбатан кам амалга оширилади.

Кўриниб турибдики, бу усул натижасида тартибланган массив ҳосил бўлиб, ҳар бир ўтишдан кейин саралашлар камайиб боради. Энг ёмон ҳолатда охирги ишни яккалик саралаш амалга оширади.

Барьер усулидан фойдаланилганда ҳар бир саралаш ўзининг барьерига эга бўлиши лозим ҳамда дастур унинг жойини аниқлаши учун уни иложи борича осонлаштириш лозим. Шунинг учун массивни [-h1..N] гача кенгайтириш лозим бўлади.

h[1..t] – қадамлар ўлчами массиви

a[1..n] - сараланаётган массив

k – саралаш қадами

x – қўшилаётган элемент қиймати

Subroutine ShellSort

```
const   t = 3;
          h(1) = 5;
          h(2) = 3;
          h(3) = 1;
var
  h: array [1..t] of integer;
  a: array [1..n] of integer;
   k, x, m, t, i, j: integer;
 begin
   for  m = 1 to t do
     begin
       k:= h(m);
       for i = k + 1 to n do
         begin
          x:= a(i);
          for j = i-k  downto k   do
             begin
              if  x < a(j ) then
                a(j+k):= a(j);
               else
                 goto 1;
              end ;
             end;
            end;
1:          a(j+1) = x ;
         end;
    end .
```

Умуман олганда, қандай қадамлар танланганда энг яхши натижа олиниши исботланмаган бўлсада, лекин бу қадамлар бири иккинчисини кўпайтувчилари бўлмаслиги лозимлиги аниқланган.

Д. Кнут қадамларни қуйидагича кетма-кетлигини таклиф қилган (тескари тартибда): 1,3,7,15,31,…,яъни: $h_{m-1}=2h_m+1$, $h_t=1$, $t = [\log_2 n] - 1$. Агар қадамлар ушбу кўринишда аниқланса, алгоритм самарадорлиги тартиби $\mathbf{O}(n^{1.2})$.

Назорат саволлари

1. Саралаш деганда нимани тушунасиз?

2. Саралашнинг асосий усулларини айтиб беринг.
3. Саралашнинг қайси усулари қатъий усулга тегишли?
4. Саралашнинг яхшиланган усулларини айтиб беринг.
5. Қандай саралаш турғун дейилади?
6. Тўғридан-тўғри қўшиш усули ғояси нимадан иборат?
7. Тўғридан-тўғри танлаш усули ғояси нимадан иборат?
8. Тўғридан-тўғри алмаштириш усули ғояси нимадан иборат?
9. Юқоридаги усулларнинг бир биридан фарқини айтиб беринг.
10. Қайси саралаш усули энг самарали бўлиб ҳисобланади?
11. Шелл усули қайси асосий саралаш усулига тегишли?

9–маъруза: Медианалар ва тартиблаш статистикаси. Минимум ва максимум. Бир вақтда минимума ва максимумни қидириш.

Режа:

1. Медианалар ва тартиблаш статистикаси;
2. Минимум ва максимум;
3. Бир вақтда минимума ва максимумни қидириш;

Жойлаштириш усули (**хешлаштириш**) маълумотлар тузилмасида элемент жойлашган ўринни тез аниқлашга йўналтирилган усулдир. Жойлаштириш усулида маълумотлар оддий массив сифатида ифодаланган бўлади. Ушбу усулда маълумотлар калитлари *H* акслантириш ёрдамида массив индексларига акслантирилади. Шу сабабли мазкур акслантириш *«калитларни акслантириш»* деб номланади.

Элементни жадвалга қўшишдан олдин унинг адреси хеш-функция орқали аниқланади: *A = h(K)*, бу ерда K – калит, A – жадвалдаги элемент адреси бўлиб, $0 \le A \le N-1$, шарт ўринли бўлади.

Ушбу усулдан кўпинча дарахтларда ҳамда уни тадбиқ этиш мувоффақият келтирувчи масалаларда фойдаланилади.

Калитларни акслантириш билан боғлиқ бўлган асосий муаммо бу калитларни қабул қилиши мумкин бўлган қийматлар тўпламини хотира адреси (массив индекслари) қабул қилиши мумкин бўлган тўпламдан анча кенглигидир. Қуйидагича мисол кўриб чиқайлик. Фараз қилайлик, 1000 та одам бор. Ҳар бир одамни аниқлаш (идентификация қилиш) учун калит сифатида исмларини қарайлик. Фараз қилайлик ҳар бир исм 16 тагача ҳарфдан ташкил топган бўлсин. У ҳолда биз бўлиши мумкин бўлган калит қийматлари тўпламини (бу ерда бўлиши мумкин бўлган калитлар тўплами 26^{16} та элментдан иборат бўлади, агар алифбо 26 та харфдан иборат бўлса),

10^3 та индексли массивга акслантириш лозим бўлади. Юқоридаги мисолдан кўриниб турибдики, H функция ичига акслантирувчи акслантириш бўлади, яъни «кўпгина қиймат бир қийматга акслантирилади». Агар бирор бир калит берилган бўлса, у ҳолда қидирув амалининг биринчи қадами - калит билан боғланган индексни ҳисоблаш, яъни h = H(k), иккинчи ва асосийси – текшириш бўлиб ҳисобланади, яъни бунда ҳақиқатдан ҳам h функция T массивда (жадвалда) k калитли элементни идентификация қилаётгани текширилади.

Акслантириш функциясини танлаш

Ўз-ўзидан маълумки, акслантиришнинг ихтиёрий яхши функцияси калитларни берилган индекслар оралиғига иложи борича тенг тақсимлайди. Агарда ушбу талаб ўринли бўлса, у ҳолда қўшимча шарт ва чекланишлар бўлмайди. Агарда акслантириш мутлақо тасодифий бўлса яна ҳам яхшироқ бўлади. Мазкур усулга “майдалаш” (хешлаштириш), яъни аргументни бўлаклаш деб аталади. H функция эса “жойлаштириш функцияси” деб аталади. Равшанки, H функция иложи борича самарали хисобланиши лозим, яъни унча кўп бўлмаган асосий арифметик амаллардан ташкил топган бўлиши лозим.

Фараз қилайлик, k калит тартиб рақамини барча мумкин бўлган калитлар тўпламида ифодаловчи ORD(k) функция берилган бўлсин. Бундан ташқари, i массив индекси 0 **...** N — 1 ораликда жойлашган деб фараз қиламиз, бу ерда N — массив ўлчами. Бундай ҳолда “жойлаштириш функцияси” сифатида қуйидаги функцияни олиш мумкин:

H(k) = ORD(k) MOD N

Агарда “жойлаштириш функцияси” юқоридаги каби аниқланадиган бўлса, у ҳолда калит қийматлари индекслар ўзгариши оралиғига текис тақсимланади. Шу сабабли, калитларни акслантириш амалга оширилаётган вақтда кўпинча уни асос қилиб олишади. Бундан ташқари, агар масив узунлиги N иккининг қандайдир даражасига тенг бўлса, у ҳолда функция самарали ҳисобланади. Лекин, агар калит харфлар кетма-кетлигидан ташкил топган бўлса, у ҳолда айнан бундай функциядан воз кечишга тўғри келади. Сабаби, бундай ҳолатда барча калитлар тенгэхтимолликга эга деб қараш нотўғри бўлади. Натижада, фақатгина бир неча белгилар билан фарқланувчи сўзларни битта индексга акслантириш эхтимоллиги юқори бўлади, бу эса калитларни нотекис тақсимланишига олиб келиши мумкин. Шу сабабли, амалий масаллар ҳал қилинаётганда N сифатида туб сонларни олиш тавсия этилади.

Зиддиятни ҳал қилиш алгоритмлари

Агар берилган калитга мос жадвал қатори керакли (қидирилаётган) элементга эга эмаслиги маълум бўлса, у ҳолда зиддият (“конфликт”) юзага келди дейилади. Бундай ҳолат, агарда бир неча элемент битта индексга акслантириладиган калитларга эга бўлса юзага келади. Бундай ҳолатда мазкур берилган калит орқали тўлиқ аниқланувчи индекс бўйича иккинчи уриниш амалга оширилади (Муқобил индекс шакллантириш орқали). Иккинчи индексни шакллантиришнинг бир қанча усуллари мавжуд. Энг содда йўлларидан бири бу – биринчи H(k) индекслари бир ҳил бўлган барча қаторни бир-бирига боғлаш, яъни боғланган рўйхат каби. Бундай усулга тўғридан-тўғри боғлаш (direct chaining) деб аталади. Ҳосил бўлган рўйхат элементлари асосий жадвалда жойлашиши ҳам жойлашмаслиги ҳам мумкин.

Бундай ҳолатда рўйхат элементлари жойлашган хотира тўлалик (тўлиб-тошиш, переполнение) соҳаси дейилади. Ушбу усулни камчилиги, иккиламчи рўйхатларни кузатиб бориш ҳамда зиддиятга борувчи элементлар рўйхати ҳар бир қаторида мурожаат учун жой ажратиш лозим бўлади.

Зиддиятни ҳал қилишнинг яна бир усулида эса берилган жадвални берилган қаторида керакли элемент мавжуд бўлмаса, токи керакли элемент топилгунча ёки бўш қаторга боргунча бошқа қаторларини кўриб чиқилади. Агар қараб чиқиш бўш қаторгача бориб етса, у ҳолда кўрсатилган калит берилган жадвалда йўқ деб ҳисобланади. Зиддиятни бундай ҳал қилиш усулига очиқ адресли деб аталади. Табиийки, ихтиёрий берилган калит учун индекслар кетма-кетлиги иккинчи уринишда бир ҳил бўлиши лозим. Бундай ҳолатда қараб (кўриб) чиқиш алгоритми қуйидаги схемада ишлайди:

```
h = H(k)
i = 0
repeat
  if T(h)  =  k
        then  элемент топилди
        else if T(h) = free
                        then элемент жадвалда йўқ
                        else {зиддият}
                        i := i + 1
                        h := H(k) + G(i)
                endif
          endif
until  топилди, ёки жадвалда йўқ.
```

Назорат саволлари

1. Калитларни алмаштириш нима?
2. Акслантириш функцияси вазифаси нимадан иборат?
3. Қандай ҳолатларда зиддият юзага келади?