3- Mavzu: Algoritmik tillar.
Algoritmlarning tahlili asoslari.

1.Algoritmik tillar

Ushbu qo'llanmaning asosiy maqsadi algoritmik tafakkurni rivojlantirish bo'lganligi bois, qo'yilgan masalalarni yechishda o'zimiz uchun dasturlash tiliga o'xshash va ularning umumiy jihatlarini o'z ichiga olgan maxsus tilni tashkil etamiz. Har qanday tilda bo'lgani kabi bu tilda ham **alifbo, sintaksisi** va **semantika** bo'ladi. Bu tushunchalarni qisqacha yoritib o'tamiz.

Alifbo — aniq bir til uchun asosiy belgilar ro'yxati, ya'ni shu tildagi matnlarni yozish uchun qo'llaniladigan «alifbo harflari» — boshqa belgini qo'llash mumkin emas.

Sintaksis — bu jumlalarni hosil qilish qoidasi bo'lib, biror jumlani to'g'ri yoki xato yozilganligini aniqlash uchun xizmat qiladi. Aniqroq qilib aytadigan bo'lsak, til sintaksisi shu tilda belgilarni ma'noga ega bo'ladigan biriktirishni aniqlab beruvchi qoidalar ro'yxati.

Semantika — jumla yoki gaplarning mazmunini aniqlaydi. Semantika hosil qilingan jumlalar yoki gaplarni qanday amallar ketma-ketligini aniqlab berishini ta'minlaydi.

Qulay belgilashlarni o'ylab topish san'ati inson madaniyatida juda muhim ahamiyatga ega. Masalan, sonlarni belgilashni olaylik. Avvalgi bobda aytib o'tilganidek, hammangiz sonlarni ustun ko'rinishda qo'shish va ko'paytirishni bilasiz. Al-Xorazmiy to- monidan yoritib berilgan o'nlik belgilash sistemasi bunga imkon beradi. Rim raqamlari orqali yozilgan sonlarni qo'shib ko'ring- chi, qo'shish masalasini hal etishda belgilashlar sizga hech qanday yordam bermayotganini ko'rasiz. Yana, masalan, musiqani olaylik. Musiqa tovushlarini belgilash uchun notani o'ylab topib, musi- qachilar ancha murakkab va qiziqarli musiqalarni yozish hamda tarqatish imkoniyatiga ega bo'ldilar. Shu kabi juda ko'p misollarni keltirish mumkin.

Agar algoritm yordamida joiz boshlang'ich qiymat asosida izlangan natijani olish mumkin bo'lsa u holda **algoritmni joiz** boshlang'ich qiymatga **qo'llash mumkin** deyiladi. Agar boshlang'ich qiymat joiz bo'lsa ham natija olish mumkin bo'lmasa, u holda unga **algoritm qo'llash mumkin emas** deyiladi.

Endi joiz boshlang'ich qiymatlar sinfi qanday ekanligini ko'rib chiqamiz. Boshlang'ich qiymatlar ba'zan narsa yoki buyumlar, sonlar ekanini ko'rdik. Bu fikr olingan natijalar uchun ham o'rinli. Bu narsalar orasidagi umumiylik nimada? Algo- ritm — bu qoidalar va demakki, ular qandaydir tillarda ifoda- langan, degan fikrni e'tiborga olsak, bu umumiylik ko'rinadi. Bir necha marta bu qoidalarni aniq bajarilishi qanchalik muhim ekanligi haqida gapirib o'tdik.

Algoritmikada qulay belgilashlar ham, albatta, yanada muhim ahamiyatga ega. Bunda uning o'ziga xos tomonlari ham bor. Biz ham o'z belgilash usulimizni kiritishdan avval ba'zi tomonlarga e'tibor berishimiz lozim bo'ladi. Yozish uchun lotin harflari va o'nlik sanoq sistemasidagi raqamlardan foydalanamiz. **Ko'rsatmalar**

so'zlar bo'lib, ularni yozish uchun jumlalarga buyruq mazmunini beramiz. Bunda ularni qisqa va ma'noli bo'lishiga e'tibor beramiz

Algoritmni oddiy tilda ifodalash. Algoritmlarni ifodalashning eng keng tarqalgan shakli - oddiy tilda so'zlar bilan bayon qilishdir. Bu nafaqat hisoblash algoritmlarida, balki hayotiy, turmushdagi "algoritm"larga ham tegishlidir. Masalan, biror bir taom yoki qandolat mahsulotini tayyorlashning retsepti ham oddiy tilda tavsiflangan algoritmdir. Shaharlararo telefon - avtomat orqali aloqa o'rnatishning o'ziga xos algoritmidan foydalanasiz. Do'kondan yangi kir yuvish mashinasi yoki magnitofon sotib olinsa, ishni foydalanishning algoritmi bilan tanishishdan boshlaymiz. Masalani kompyuterda echishda ham, ko'pincha matematika tilini ham o'z ichiga olgan tabiiy tildan foydalanish mumkin. Algoritmning bunday tildagi yozuvi izlanayotgan natijaga olib keladigan amallar ketma-ketligi ko'rinishida bo'lib, odam tomonidan bir ma'noli idrok etilishi kerak. So'zlar bilan ifodalangan har bir amal "algoritmning qadami" deb ataladi.

2.Dasturlash tillari va ularni tasniflash.

Hozirgi kunda dasturlash tillarini u yoki bu belgisi bo'yicha tasniflash mumkin. Dasturlash tilining kompyuterga bog'liqlik darajasi bo'yicha tasniflash eng umumiy hisoblanadi.
Yuqorida aytilgan belgiga qarab, dasturlash tillari kompyuterga bog'liq va kompyuterga bog'liq bo'lmagan tillarga bo'linadi.
Kompyuterga bog'lik tillar, o'z navbatida, kompyuter tillari va kompyuterga mo'ljallangan tillarga ajratiladi.
Dasturlash tilining kompyuter tiliga yaqinligi darajasini tariflash uchun til darajasi tushunchasi qo'llaniladi.
Kompyuter tili 0 daraja deb qabul qilingan bo'lib, sanoq boshi hisoblanadi.
Odamning tabiiy tili "eng yuqori darajadagi til" deb qaraladi.
Kompyuterga bog'liq bo'lmagan tillar ham ikkita turga bo'linadi:

Protseduraga mo'ljallangan tillar

Muammoga mo'ljallangan tillar

Protseduraga mo'ljallangan tillar turli masalalarni echish algoritmlarini (protseduralarni) tavsiflashga mo'ljallangan; shuning uchun ular ko'pincha oddiy qilib "algoritmik tillar" deb ataladi.
Ushbu tillar echilayotgan masalalar xususiyatlarini to'la hisobga oladi va kompyuterning turiga deyarli bog'liq emas. Bu xildagi tillar tarkibi kompyuter tiliga qaraganda tabiiy tilga, masalan, ingliz tiliga yaqinroq.
Hozirgi kunda hisoblash, muhandis-texnik, iqtisodiy, matnli va sonli axborotlarni taxlil qilish va boshqa masalalarni echish tillari malum.
Masalan: FORTRAN tili 1954 yili ishlab chiqilgan bo'lib, FORmula TRANslator - formulalar translyatori degan manoni anglatadi va ilmiy va muhandis - texnik masalalarni hisoblashlarda qo'llaniladi.

ALGOL tili 1960 yili yaratilgan bo’lib, ALGORITMIC Langauge -algoritmik til degan ma'noni anglatadi va ilmiy-texnik masalalarni hisoblashlarda qo’llaniladi.
KOBOL tili 1959 yili yaratilgan bo’lib, Common Businees Oriented Langauge - “savdo-sotiq masalalariga mo’ljallangan til” degan ma'noni anglatadi. Korxona va tarmoqning moddiy boyligini, moliyasini, ishlab chiqargan mahsulotini hisobga olish bilan bog’liq iqtisodiy masalalarni echish uchun qo’llaniladi.
PASKAL tili 1971 yilda e`lon qilingan bo’lib, frantsuz olimi Blez Paskal nomiga qo’yilgan. Turli xildagi masalalar echimini olishda tartiblangan (strukturaviy) dasturlar tuzishda qo’llaniladi.
PL/1 tili 1964 yilda yaratilgan bo’lib, Programming Langauge/ 1 - 1-tartib raqamli dasturlash tili ma'nosini anglatadi. Ushbu til universal tillar turkumiga kiradi.
Bu tilda ishlab chiqilgan dasturlar kompyuterni yangisi bilan almashtirilganda qaytadan tuzib chiqilishi zarur emas.
BEYSIK (BASIC - Beginner's All Purpose Sumbolic Instruction Code - boshlovchilar uchun ko’p maqsadli dasturlash tili) hisoblash algoritmlarini yozish uchun qo’llaniladigan algoritmik tildir. Bu til 1965 yilda Dartmut kolleji xodimlari Kemini va Kurtslar tomonidan ishlab chiqilgan.
Protseduraga mo’ljallangan tillardan masalalarning matematik ifodalari, algoritmlar va dasturlash usullari bilan tanish bo’lgan mutaxassislar foydalaniladilar.
Bunda ulardan kompyuterning tuzilishini mukammal bilish talab qilinmaydi.
Muammoga mo’ljallangan tillar kompyuterda masala echish usullari va dasturlash usullari bilan tanish bo’lmagan foydalanuvchilar uchun yaratilgandir. Foydalanuvchi masalani tariflashi, boshlang’ich malumotlarni berishi va natijani chiqarishning talab qilingan ko’rinishini aytishi kifoya.

3.Algoritmning asosiy turlari

Masala echimining algoritmi ishlab chiqilayotgan davrda asosan uch xil turdagi algoritmlardan foydalanib, murakkab ko’rinishdagi algoritmlar yaratiladi.
Algoritmning asosiy turlariga chizig’li (a), tarmoqlanadigan (b) va takrorlanadigan (c) ko’rinishlari kiradi.
Murakkab masalalarning echimini olish algoritmlari yuqoridagi turlarining barchasini o’z ichiga olishi mumkin.
Chiziqli turdagi algoritmlarda bloklar biri ketidan boshqasi joylashgan bo’lib, berilgan tartibda bajariladi. Bunday bajarilish tartibi “tabiiy tartib” deb ham yuritiladi.
Yuqorida ko’rib o’tilgan birinchi misol chiziqli turdagi algoritmga misol bo’ladi. Amalda hamma masalalarni ham chiziqli turdagi algoritmga keltirib echib bo’lmaydi.

O’zaro kesiladigan chiziqlar soni ko’p bo’lganda, chiziqlar soni haddan tashqari ko’p bo’lsa va yo’nalishlari ko’p o’zgaraversa tuzimdagi ko’rgazmalik yo’qoladi. Bunday hollarda axborot oqimi chizig’i uzishga yo’l qo’yiladi, uzilgan chiziq uchlariga "birlashtiruvchi" belgisi qo’yiladi. Agar uzilish bitta sahifa ichida bo’lsa,

O belgisi ishlatilib, ichiga ikki tarafga ham bir xil harf-raqam belgisi qo’yiladi. Agar tuzim bir necha sahifaga joylansa, bir sahifadan boshqasiga o’tish "sahifalararo bog’lanish" belgisi ishlatiladi. Bunda axborot uzatilayotgan blokli sahifaga qaysi sahifa va blokka borishi yoziladi, qabul qilinayotgan sahifada esa qaysi sahifa va blokdan kelishi yoziladi.

Chiziqli xisoblash jarayonining tuzimi quyidagicha ko`rinishda ifodalanadi. yoki boshqa ifodaga ko’ra amalga oshirilishi mumkin yani birorta mantiqiy shartni bajarilishiga bog’lik holda hisoblash jarayoni u yoki bu tarmoq bo’yicha amalga oshirilishi mumkin. Bunday tuzilishdagi hisoblash jarayonining algoritmi **“tarmoqlanuvchi turdagi algoritm”** deb ataladi.

Algoritmning bu konstruktsiyasi tuzimda

yo`q va **ha**

ko`rinishida ifodalanadi.

Ko’pgina hollarda masalalarning echimini olishda bitta matematik bog’lanishga ko’ra unga kiruvchi kattaliklarni turli qiymatlariga mos keladigan qiymatlarini ko’p martalab hisoblash to’g’ri keladi.

Hisoblash jarayonining bunday ko’p martalab takrorlanadigan qismi “takrorlanishlar” deb ataladi. Takrorlanishlarni o’z ichiga olgan algoritmlar “takrorlanuvchi turdagi algoritmlar” deb ataladi. Takrorlanuvchi turdagi algoritmni yozish va chizish o’lchamlarini sezilarli darajada qisqartirish takrorlanadigan qismlarni ixcham ifodalash imkonini beradi.

1. Algoritm tahlili tushunchasi

Algoritm tahlilini, qo’yilgan masalani ushbu algoritm bilan echish qancha vaqt talab qilishi darajasi deb tasavvur qilish mumkin. Har bir qaralayotgan algorimtni N o’lchovli boshlang’ich ma'lumotlar massividagi masalalarning qanchalik tez echilishi bilan baholaymiz. Masalan, saralash algoritmi N ta qiymatdan iborat ro’yxatni o’sish tartibida joylashtirish uchun qancha taqqoslash talab qiladi yoki N*N o’lchamli ikkita matritsani ko’paytirishda qancha arifmetik amallar zarurligini hisoblash. Bitta masalani turli algoritmlar bilan echish mumkin. Algoritmlar tahlili bizga algoritmni tanlash uchun qurol bo’ladi. To’rtta qiymatdan eng kattasini tanlaydigan ikkita algoritmni qaraymiz:

```
largest = a
if b > largest then
largest = b
end if
return a
if s > largest then
largest = s end if
if d > largest then
largest = d end if
 return largest
```

```
if a > b then if a > s then if a > d then
return a
else
 return d end if
else
 if s > d then return s
 else
return d end if end if
else
if b > s then if b > d then
```

```
return b
else
return d end if
else
if s > d then
return s
else
return d
end if end if end if
```

Ko’rinib turibdiki, qaralayotgan algoritmlarning har birida uchta taqqoslash bajariladi. Birinchi algoritmni o’qish va tushunish oson, ammo kompyuterda bajarilish nuqtai nazaridan ularning murakkablik darajalari teng. Bu ikki algoritm vaqt nuqtai nazaridan teng, lekin birinchi algoritm largest nomli qo’shimcha o’zgaruvchi hisobiga ko’proq xotira talab qiladi. Agarda son yoki belgilar taqqoslansa, ushbu qo’shimcha o’zgaruvchi katta ahamiyatga ega bo’lmaydi, lekin boshqa turdagi ma'lumotlar bilan ishlaganda bu muhim ahamiyatga ega. Ko’plab zamonaviy dasturlash tillari katta va murakkab ob'ektlarni yoki yozuvlarni taqqoslash operatorlarini aniqlash imkonini beradi. Bunday hollarda qo’shimcha o’zgaruvchilarni joylashtirish katta joy talab qiladi. Algoritmlarning effektivligini tahlili qilishda bizni birinchi navbatda vaqt masalasi qiziqtiradi, ammo xotira muhim rol o’ynaydigan vaziyatda uni ham muhokama qilamiz. Algoritmlaring turli xossalari bitta masalani echuvchi ikki turdagi algoritmlarning effektivligini taqqoslash uchun xizmat qiladi. Biz shuning uchun hech qachon matritsalarni ko’paytirish algoritmi bilan saralash algoritmini emas, balki ikkita turli saralash algoritmlarini bir-biri bilan taqqoslaymiz.

Algoritm tahlilining natijasi – belgilangan algoritmning kompyuterdan qancha vaqt yoki takrorlash talab qilishini aniq hisoblovchi formula emas. Bunday ma'lumot muhim emas, bu holatda kompyuter turi, u bitta yoki undan ortiq foydalanuvchi tomonidan ishlatilyaptimi, uning protsessori va chastotasi qanaqa, protsessor chipida komandalar to’liqmi va kompilyator bajarilayotgan kodni qay darajada amalga oshirmoqda kabi tomonlarni nazarda tutish kerak. Bu shartlar algoritm bajarilish natijasida dasturning ishlash tezligiga ta'sir qiladi. Yuqoridagi shartlar hisobiga dasturni boshqa tez ishlaydigan kompyuterga o’tkazilganda algoritm yaxshi ishlaganday bajarilishi tezroq amalga oshadi. Aslida esa unday emas, biz shuning uchun tahlilimizda kompyuterning imkoniyatlarini inobatga olmaymiz.

2.Boshlang’ich ma'lumotlarning sinflari

Algoritmni tahlil qilishda kiruvchi ma'lumotlarni tanlash uning bajarilishiga ta'sir qilishi mumkin. Aytaylik, ba'zi saralash algoritmlari, agar kirish ro’yxati saralangan bo’lsa, juda tez ishlashi mumkin, boshqa algoritmlar shunday ro’yxatda uncha yaxshi bo’lmagan natijani ko’rsatadi. Tasodifiy ro’yxatda esa natija buning teskarisi bo’lishi mumkin. Shuning uchun biz ma'lumotlarning bir kirish ro’yxatidagi algoritmlar harakatini tahlil qilish bilan chegaralanmaymiz. Biz

algoritmni ham eng tez, ham eng sekin ishlashini ta'minlovchi ma'lumotlarni qidiramiz. Bundan tashqari, barcha mavjud ma'lumotlar to’plamidagi algoritmlarning o’rtacha samarasini ham baholaymiz.

Nisbatan murakkab masalalarni echishda algoritmdan muayyan kompyuter tilidagi dasturga o’tish juda qiyin. Bunday bevosita o’tishda algoritmning alohida qismlari orasidagi bog’lanish yo’qoladi, algoritm tarkibining asosiy va muhim bo’lmagan qismlarini farqlash qiyin bo’lib qoladi. Bunday sharoitda keyinchalik aniqlash va to’g’rilash ancha vaqt talab qiladigan xatolarga osongina yo’l qo’yish mumkin. Odatda algoritm bir necha marta ishlab chiqiladi, ba'zan xatolarni to’g’rilash, algoritm tarkibini aniqlashtirish va tekshirish uchun bir necha marta orqaga qaytishga to’g’ri keladi. Algoritm ishlab chiqishning birinchi bosqichida algoritmni yozishning eng qulay usuli - algoritmni tuzim ko’rinishda ifodalashdir.

3.Xotira bo’yicha murakkablik

Biz asosan algoritmlarning vaqt bo’yicha murakkabligini muhokama qilamiz, ammo ish bajarish uchun u yoki bu algoritmga qancha xotira kerakligi haqida ham aytish mumkin. Kompyuter xotirasi (ham ichki, ham tashqi) hajmi chegaralangan. Kompyuterlar rivojlanishining dastlabki bosqichlarida bu tahlil uslubiy xarakterga ega edi. Barcha algoritmlar chegaralangan xotira etarli yoki qo’shimcha maydonni talab qiluvchi algoritmlarga bo’linadi. Ko’pincha dasturchilar xotirasiga ega va tashqi qurilmalar talab qilmaydigan sekin ishlovchi algoritmni tanlashar edi.

Kompyuter xotirasiga bo’lgan talab juda katta edi, shuning uchun qaysi ma'lumotlar saqlanib qoladi, bunday saqlashning samarali usullari qanday kabi savollar o’rganilar edi. Faraz qilaylik, masalan, biz -10 dan +10 gacha intervaldagi verguldan keyin bitta o’nli belgiga ega bo’lgan haqiqiy son yozyapmiz. Haqiqiy sonni yozishda ko’pchilik kompyuterlar 4 dan 8 baytgacha xotira sarflaydi, lekin agar bu son avvaldan 10 ga ko’paytirsak, -100 dan +100 gacha intervaldagi butun son hosil qilamiz va uni saqlash uchun bor yo’g’i bir bayt sarflanadi. Birinchi variant bilan solishtirsak, 3-7 bayt tejashga erishildi. 1000 ta shunday son saqlaydigan dastur 3000 dan 7000 baytgacha tejaydi.

Agar o’tgan asrning 80-yillarida kompyuterlarning xotirasi 65536 bayt bo’lganligini e'tiborga olsak, jiddiy tejash ko’zga tashlanadi. Aynan shu kompyuter dasturlarining uzoq yil ishlashi xotirani tejash zaruriyati Bilan bir qatorda 2000 yil muammo tug’dirdi. Agar sizning dasturingiz turli sanalardan foydalansa, yilni yozish uchun 1999 o’rniga 99 ifodasini saqlagan holda joyning yarmini tejasa bo’ladi. 80-yillardagi dastur mualliflari mahsulotlari 2000 yilgacha yashashini taxmin ham qilishmagan edi.

4.Eng yaxshi holat uchun murakkablik

Bo’limning nomlanishidan ham ko’rinib turibdiki, algoritmlar uchun eng yaxshi holat bu qisqa vaqt ichida amalga oshiriladigan algoritmning ma'lumotlar jamlanmasi. Bunday jamlanma algoritm eng amal bajaradigan qiymatlar

kombinatsiyasini ifodalaydi. Agar biz izlash algoritmini tekshirsak, izlangan qiymat birinchi algoritm tekshirayotgan katakka yozilgan bo'lsa (odatda maqsadli qiymat yoki kalit deb ataladi), ma'lumotlar to'plami eng yaxshi hisoblanadi. Bunday algoritmga uning murakkabligidan qatiy nazar, bitta taqqoslash kerak bo'ladi. Shuni eslatish kerakki, ro'yxatdan izlashda, uning qanchalik uzun bo'lishidan qatiy nazar, eng yaxshi holat doimiy vaqtni talab qiladi. Umuman, eng yaxshi holatda algoritmni bajarish vaqti kichik yoki doimiy bo'ladi, shuning uchun biz bunday tahlilni kam o'tkazamiz.

5.Eng yomon holat uchun murakkablik

Algoritmlarni baholash ob'ektivligini oshirish uchun vaqt bo'yicha asimptotik murakkablik tushunchasi algoritm effektivligining asosiy o'lchovi sifatida qabul qilingan. Algoritmlar effktivligi termini ushbu o'lchovning sinonimi hisoblanib, asosan eng yomon holatda algoritmning bajarilish vaqtiga taalluqli[1].
Eng yomon holatni tahlil qilish juda muhim, chunki u algoritm ishining maksimal vaqtini tasavvur qilishga yordam beradi. Eng yomon holatni tahlil qilganda algoritm eng ko'p ish bajaradigan kirish ma'lumotlarini topish zarur. Izlovchi algoritm uchun bu kabi kiruvchi ma'lumotlar – bu shunday ro'yxatki, unda izlangan kalit oxirida keladi yoki umuman bo'lmaydi. Natijada N taqqoslash kerak bo'ladi. Eng yomon holatning tahlili tanlangan algoritmga qarab dasturning ishlash vaqti uchun yuqori bahoni beradi.

6.O'rtacha holat

O'rta holatning tahlili eng murakkab hisoblanadi, chunki u ko'pgina detallarni hisobga olishni talab qiladi. Tahlil asosini mavjud bo'lgan kiruvchi ma'lumotlar to'plamini bo'lib chiqish lozim bo'lgan turli guruhlarni aniqlash tashkil qiladi. Ikkinchi qadamda kiruvchi ma'lumotlar to'plami qaysi guruhga tegishli bo'lish ehtimoli aniqlanadi. Uchinchi qadamda har bir guruhdagi ma'lumotlarga algoritmning ish vaqti hisoblanadi. Algoritmning bir guruhdagi hamma kiruvchi ma'lumotlar uchun ishlash vaqti bir xil bo'lishi kerak, aks holda guruhni bo'lish lozim. O'rtacha ish vaqti

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

formula orqali hisoblanadi. Bu erda n - kiruvchi ma'lumotlar o'lchami, m - guruhlar soni, pi - kiruvchi ma'lumotlarning i sonli guruhga tegishlilik ehtimoli, ti – i sonli guruhdagi ma'lumotni qayta ishlash uchun algoritmga kerak bo'ladigan vaqt deb belgilangan.

Ba'zi hollarda biz kiruvchi ma'lumotlarning har bir guruhga tushish ehtimolini bir Xill deb taxmin qilamiz. Boshqacha aytganda, agar guruh 5 ta bo'lsa, birinchi

guruhga tushish extimoli ikkinchi yoki boshqa guruhga tushish extimolidek, ya'ni har bir guruhga tushish extimoli 0,2 ga teng. Bu holda ishning o’rtacha vaqtini avvalgi formula Bilan yoki unga ekvivalent soddalashtirilgan barcha guruhlarning teng extimolligida haqiqiy bo’lgan formuladan foydalanishimiz mumkin.

$$A(n)\sum_{i=1}^{m} p_i t_i, \quad (2)$$

7.Ehtimolliklar

Biz algoritmlarni kiruvchi ma'lumotlarga ko’ra tahlil qilmoqchimiz, buning uchun esa u yoki bu kiruvchi ma'lumotlar to’plami qanchalik ko’p uchrashini baholashimiz kerak. Shu bilan birga, biz kiruvchi ma'lumotlar u yoki bu sharoitlarga to’gri kelish extimolligi bilan ishlashimizga to’g’ri keladi. U yoki bu hodisaning extimolligi nol va bir oralig’idagi sondan iborat, 0 extimolligi hodisa hech qachon sodir bo’lmasligi,1 extimoli esa bo’lishi mumkinligini bildiradi. Agar bizga turli kiruvchi qiymatlarning soni aniq 10 ga tengligi ma'lum bo’lsa, ishonch Bilan aytishimiz mumkinki, har qanday bunday kirishning extimolligi 0 va 1 oralig’ida bo’ladi, barcha extimolliklarning yig’indisi 1 ga teng, chunki ulardan bittasi amalga oshishi mumkin. Agar har bir kirishning amalga oshish extimolligi bir xil bo’lsa, ulardan har birining extimolligi 0.1 ga teng bo’ladi (10 dan 1 yoki 1/10).

Bizning tahlilimiz, asosan barcha imkoniyatlarni ko’rib chiqishdan iborat bo’ladi, keyin esa biz ularning hammasi teng extimolli deb faraz qilamiz. Agar imkoniyatlarning umumiy soni N ga teng bo’lsa, ulardan har birining amalga oshishi extimolligi 1/N ga teng bo’ladi.

Algoritmlarning axborotlarni qayta ishlash jarayoni

Algoritm va hisoblash jarayoni orasidagi farqni ko’rish qiyin emas. Masalan, algoritmda bir marta uchragan amal bir necha marta bajarilgan bo’lishi mumkin va hisoblash jarayonida bir necha marta ifodalanishi mumkin. Bu amallarga misol sifatida 3- yoki 5- yoki 6-amallarni keltirishimiz mumkin. Bizning misolda qoida bo’yicha bir turdagi amallar turli berilganlar ustida bajariladi. Biroq, bu berilganlar bir xil nomda tasvirlanishi mumkin.

Algoritm bo’yicha har doim ham hisoblash jarayonini oldindan aytib bera olmaymiz. Masalan, hisoblash jarayonida amallar soni qanchalik ko’p bo’lishini oldindan aytish qiyin. Algoritmni bajarish №1 amaldan boshlanadi. Algoritmda har doim keyingi bajariladigan amal aniqlangan bo’ladi. Yashirin holda u keyingi nomer bilan belgilangan amal hisoblanadi. Agar algoritmdagi amallar tartibi nomerga mos tushmasa, u o’tish amali yordamida ko’rsatiladi. To’xtatish amalidan so’ng algoritm bajarilishi to’xtatiladi. Shunday qilib, navbatdagi amal bir qiymatli aniqlangan. Determinallashgan deb nomlanuvchi bu algoritmning xossasiga ko’ra, boshlang’ich berilganlar uchun hisoblash jarayoni har doim aniqlangan bo’ladi. Shunday qilib, bir xil boshlang’ich berilganlar uchun hisoblash jarayoni ham bir xil bo’ladi.

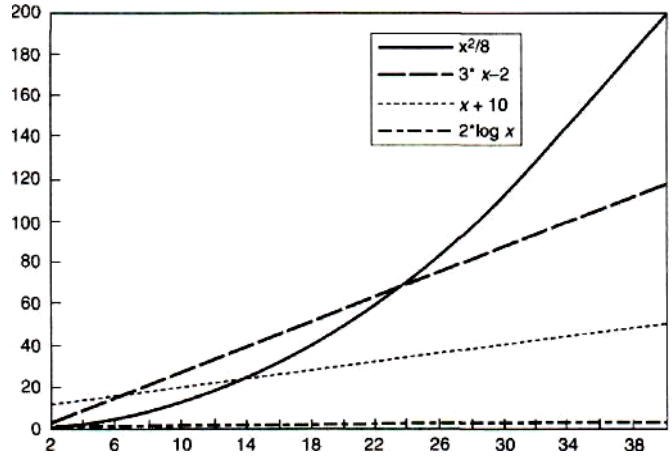
8.Qo’shish formulalari

Algoritmlarni tahlil qilganda biz ba'zi kattaliklar yig’indisini qo’shishimizga to’g’ri keladi. Aytaylik, bizda sikli algoritm bor. Agar sikl o’zgaruvchisi 5 qiymatini olsa, sikl 5 marta bajariladi, agar uning qiymati 20 ga teng bo’lsa 20 bo’ladi. Agar sikl o’zgaruvchisi M ga teng bo’lsa, sikl M marta bajariladi. Agar sikl o’zgaruvchisi 1 dan N gacha hamma qiymatlarga o’tsa, sikl bajarilishining jami soni 1 dan N gacha bo’lgan hamma natural sonlar yig’indisiga teng bo’ladi. Yig’indi belgisining pastki qismida o’zgaruvchi yig’indining boshlang’ich qiymati, yuqori qismida esa – oxirgi qiymati turibdi. Bunday ifodalanish bizni qiziqtirgan yig’indi bilan qanday bog’liqligi tushunarli.

Agar biror qiymat shu kabi yig’indi ko’rinishida yozilsa, natijani boshqa shu kabi ifodalar bilan solishtirish mumkin bo’lishi uchun uni soddalashtirish kerak.

9.O’sish tezliklari

Algoritm bilan bajariladigan jarayonlar sonini aniq bilish algoritmlarni tahlil qilishda muhim rol o’ynamaydi. Kiruvchi ma'lumotlarning hajmi ko’payganida bu sonning o’sish tezligi muhimroq hisoblanadi. U algoritmning o’sish tezligi deb ataladi. Agar 1-rasmga diqqat bilan qarasak, funktsiya grafiklarining quyidagi xususiyatlarini ko’rsatish mumkin. x^2 funktsiya avval sekin o’sadi, lekin x o’sganda uning o’sish tezligi ham oshadi. x funktsiyasining o’sish tezligi o’zgaruvchining hamma qiymatlari oralig’ida doimiydir. 2 log x funktsiyasi umuman o’smaydi, lekin bu yolg’on tasavvur. Haqiqatda esa u o’sadi, faqat juda sekin.



10.O’sish tezliklarini tasniflash

Algoritm murakkabligining o’sish tezligi muhim rol o’ynaydi va biz o’sish tezligi formulasi kata ustunlikka ega hadi bilan aniqlanishini ko’rdik. Shuning uchun biz sekin o’sadigan kichik hadlarga e'tibor qaratmaymiz. Barcha kichik hadlarni olib tashlab, murakkablikning o’sish tezligi hisoblanuvchi algoritm yoki funktsiyaning tartibiga ega bo’lamiz. Algoritmlarni ular murakkabligining o’sish tezligiga qarab guruhlarga ajratish mumkin. Biz 3 toifani kiritamiz: murakkabligi mazkur funktsiya

kabi tez o’suvchi algoritmlar, murakkabliklari o’sha tezlikda o’suvchi algoritmlar va murakkabligi bu funktsiyadan sekin o’suvchi algoritmlar**.**

Intuitiv algoritm tushunchasi

Hisoblash mashinasining ishi algoritmlarni bajarishdan iborat bo’ladi. Shuning uchun hisoblash mashinalarining umumiy imkoniyatlari qaysi muammo-masalalarni algoritm sifatida tasvirlash mumkinu, qaysilarini mumkin emasligiga bog’liq bo’ladi.Matematikaning eng asosiy tushunchalarnidan biri bo’lgan algoritm tushunchasi hisoblash masalalari paydo bo’lganidan ancha oldin vujudga kela boshlagan edi. Asrlar davomida kishilar intuitiv algoritm tushunchalaridan foydalanib kelganlar. Bu tushunchani shunday ta’riflash mumkin:

Algoritm – bu qoidalarning qat’iy va chekli sistemasi bo’lib, ba’zi ob’ektlar ustida bajariladigan amallarni aniqlaydi va chekli qadamdan keyin qo’yilgan maqsadga olib kelishni ta’minlaydi.

Dasturlash tillari va ularni tasniflash

Hozirgi kunda dasturlash tillarini u yoki bu belgisi bo’yicha tasniflash mumkin. Dasturlash tilining kompyuterga bog’liqlik darajasi bo’yicha tasniflash eng umumiy hisoblanadi. Yuqorida aytilgan belgiga qarab, dasturlash tillari kompyutera bog’liq va kompyuterga bog’liq bo’lmagan tillarga bo’linadi. Kompyuterga bog’lik tillar, o’z navbatida, kompyuter tillari va kompyuterga mo’ljallangan tillarga ajratiladi.

Dasturlash tilining kompyuter tiliga yaqinligi darajasini tariflash uchun til darajasi tushunchasi qo’llaniladi. Kompyuter tili 0 daraja deb qabul qilingan bo’lib, sanoq boshi hisoblanadi. Odamning tabiiy tili “eng yuqori darajadagi til” deb qara-ladi. Kompyuterga bog’liq bo’lmagan tillar ham ikkita turga bo’linadi: birinchisi protseduraga mo’ljallangan tillar, ikkinchisiga - muammoga mo’ljallangan tillar. Protseduraga mo’ljallangan tillar turli masalalarni echish algoritmlarini (protseduralarni) tavsiflashga mo’ljallangan; shuning uchun ular ko’pincha oddiy qilib “algoritmik tillar" deb ataladi. Ushbu tillar echilayotgan masalalar xususiyatlarini to’la hisobga oladi va kompyuterning turiga deyarli bog’liq emas. Bu xildagi tillar tarkibi kompyuter tiliga qaraganda tabiiy tilga, masalan, ingliz tiliga yaqinroq.

Foydalanilgan adabiyotlar

1. Informatika va informatsion texnologiyalar, M.Aripov va boshqalar. Oliy o‘quv yurti talabalari uchun darslik. Toshkent-2019 y.
2. Axborot texnologiyalari, M.Aripov va boshqalar. Oliy o‘quv yurti talabalari uchun o‘quv qo‘llanma. Toshkent-2019 y.
3. Delphi tilida dasturlash asoslari, Sh.Nazirov. Toshkent-2018 y.