7- AMALIY MASHG‘ULOT

Mavzu: Izlash algoritmlari.

Ishning maqsadi: Izlash va saralash algoritmlarini tahlil qilish, xususan, ketma-ket izlash, ikkilanma izlash, pufaksimon saralash, Shell saralashi algoritmlarini aniq misollarda tahlil qilish.

I. Nazariy qism

1. Izlash algoritmlari.

1.1. Ketma-ket izlash.

Ketma-ket izlash algoritmida bizni ro‘yxatdan ma’lum bir maqsad elementi deb ataluvchi elementni izlash talab qilinadi. Ketma-ket izlashda biz har doim ro‘yxat tartiblanmagan deb tasavvur qilamiz, bir qancha algoritmar tartiblangan ro‘yxat bilan ishlashda juda yaxshi natijalar ko‘rsatsa ham bunga qat’iy zarurat yo‘q. Izlash asosan kerakli element ro‘yxatda borligini tekshirishdan emas, balki elementga taalluqli ma’lumot, kalit so‘zni olish uchun qo‘llaniladi.

Masalan, kalit so‘z ishchining nomeri, tartib raqami yoki istalgan mukammal identifikator bo‘lishi mumkin. Kerakli kalit topilgandan keyin dastur tez-tez unga bog‘liq ma’lumotlarni o‘zgartiradi yoki barcha ma’lumotlarni ko‘rsata oladi. Izlash algoritmidan oldin kalitning o‘rnini aniqlash kabi muhim masala bor. Shu sababli izlash algoritmlari kerakli kalitni aniqlaydigan element o‘rnini qaytaradi. Agar kalit so‘z topilmasa, izlash algoritmi massivning yuqori chegarasidan oshadigan qiymatni qaytaradi. Bizning maqsad uchun ro‘yxat elementlari 0 dan N gacha nomerlarga ega deb tasavvur qilamiz. Agar bu element ro‘yxatda bo‘lmasa, 0 qaytarish imkonini beradi. Oddiylik uchun kalit so‘z qaytarilmaydi deb tasavvur qilamiz.

Ketma-ket izlash algoritmalari ro‘yxat bo‘ylab birinchi elementdan boshlab qidirilayotgan element topilmagunga qadar bittadan elementni tekshirib chiqadi. Aniqki, kalit so‘z qancha uzoqda joylashgan bo‘lsa, uni izlashga shuncha ko‘p vaqt ketadi. Ketma-ket izlash algoritmlarining analizi uchun buni eslab qolish talab qilinadi.

Ketma-ket izlashning to‘liq algoritmi quyidagicha ko‘rinishda:

```
SequentialSearch(list.target,N)
list spisok \\dlya prosmotra
Targe \\selevoye znacheniye
N   \\chislo elementov v spiske
for i=l to N do
if  (target=list[i]) return i
end if end for return
```

1.2. Ikkilanma izlash.

Qidirilayotgan qiymatni tartiblangan ro‘yxatning o‘rta elementi bilan solishtirganda uchta holat bo‘lishi mumkin: qiymatlar teng, qidirilayotgan qiymat kichik yoyinki qidirilayotgan qiymat katta. Birinchisida juda yaxshi holatda qidiruv tugadi. Qolgan ikki holat uchun ro‘yxat yarmini tashlab yuborishimiz mumkin.

Qidirilayotgan qiymat o‘rta qiymatdan kichkina bo‘lganda, agar u ro‘yxatda mavjud bo‘lsa, u ro‘yxatning birinchi yarmida bo‘ladi. Agar u o‘rta elementdan katta va ro‘yxatda mavjud bo‘lsa, u ro‘yxatning ikkinchi yarmida bo‘ladi. YAgona tekshirishda ro‘yxat yarmini tashlab yuborishimiz uchun shuning o‘zi yetarli.Bu jarayonni qaytarish natijasida qolgan yarim ro‘yxatning yarmini tashlab yuborishimiz mumkin. Natijada biz quyidagi algoritmga ega bo‘lamiz:

```
Binary
Search(list.target,N)
list \\ ko‘rib chiqish uchun ro‘yxat
target      \\ maqsadli qiymat
N   \\ ro‘yxatdagi elementlar soni
start=1
end=N
```

while start<=end do
middle=(start+end)/2
select(Compare(list[middle],target)) from
case -1: start=middle+1
case 0: return middle
case 1: end=middle-l
end select
end while
return 0

Agar qidirilayotgan qiymat topilgan o‘rta qiymatdan oshib ketsa, start o‘zgaruvchisiga middle o‘zgaruvchisidan 1 ta ko‘p bo‘lgan qiymat beriladi.Agar qidirilayotgan qiymat topilgan o‘rta qiymatdan kichik bo‘lsa, end o‘zgaruvchisiga middle dan 1 ta kam qiymat beriladi. Bittaga surish shuni bildiradiki, solishtirish natijasida biz o‘rta qiymat qidirilayotgan qiymatligini aniqlaymiz, shuning uchun uni tahlil qilishdan chiqarishimiz mumkin.

Takrorlanish (sikl) har doim ham to‘xtaydimi? Agar qidirilayotgan qiymat topilsa, return operatori ishlab turganligi uchun ham javob albatta maqbul bo‘ladi. Kerakli qiymat topilmagan taqdirda, har bir takrorlanish iteratsiyasida yoki start qiymati oshadi yoki end qiymati kamayadi. Bu shuni ko‘rsatadiki, qiymatlar doimiy yaqinlashadi. Ma’lum bir vaqtga yetganda bu ikki qiymat tenglashadi va takrorlanish yana bir marta start=end=middle shartni tekshirish bilan qaytariladi. Bu o‘tishdan keyin (agar element bu index bilan qidirilayotgan element bo‘lmasa) yoki start qiymati middle va end dan bittaga katta, yoki teskarisi, end o‘zgaruvchisi qiymati middle va start dan bittaga kam bo‘ladi. Ikkala holda ham while takrorlanishga qo‘yilgan shart yolg‘on va takrorlanish boshqa bajarilmaydi. Shuning uchun takrorlanish har doim tugaydi.

Bu algoritm to‘g‘ri qiymatni beradimi? Agar qidirilayotgan qiymat topilsa, javob shartsiz maqbuldir, chunki return operatori bajarildi. Agar ro‘yhatning qolgan yarim qismining o‘rta qiymati to‘g‘ri kelmaydigan bo‘lsa, har bir takrorlanishda qolgan yarim elementlar tashlab yuboriladi, chunki ular juda katta yoki juda kichik.

Yuqorida aytilganidek, natijada biz solishtirishimiz kerak bo‘lgan yagona elementga kelamiz. Agar bu bizga kerak kalit bo‘lsa, u holda middle o‘zgaruvchisi uni qaytaradi.

Agar kalit qiymati qidirilayotgan qiymatdan farqli bo‘lsa, start o‘zgaruvchisi qiymati end qiymatidan oshadi yoki teskarisi - end qiymati start nikidan kamayadi.

Agar qidirilayotgan qiymat ro‘yxatda bo‘lganda, u middle qiymatidan kichik yoki katta bo‘lardi. Ammo start va end ko‘rsatadiki, oldingi solishtirish qolgan barcha imkoniyatlarni cheklaydi va shuning uchun qidirilayotgan qiymat ro‘yhatda yo‘q.

Demak takrorlanish yakunlanadi, qaytariladigan qiymat esa nolga teng bo‘ladi, bundan ko‘rinadiki bo‘lishlarni ketma-ket tarzda bajarish lozim. Ro‘yxat hajmidan qat’iy nazar ketma-ket bo‘lish (va kerak vaqtda ko‘rilmaydigan bo‘lakni tashlab yuborish) da biz ro‘yxatdan bitta elementga kelamiz. SHunday qilib, algoritm to‘g‘ri javob qaytaradi.

Algoritm bir necha marotaba ro‘yxatni teng ikkiga bo‘lishi uchun tasavvur qilamizki, ma’lum k lar uchun $N = 2^k - 1$. Agar bu to‘g‘ri bo‘lsa, ikkinchi bo‘linishda qancha element qoladi? Uchinchida-chi? Umuman aytganda, aniqki, bir qancha bo‘linishdan keyin takrorlanish 2^j-1 elementli ro‘yxat bilan davom etadi, $2^{j-1}-1$ta element ro‘yhatning birinchi yarmida, $2^{j-1}-1$qismi esa ikkinchi yarmida. SHuning uchun keyingi o‘tishda $2^{j-1}-1$ta element $1 \le j \le k$ qoladi. Bu holat so‘nggi tahlilni osonlashtiradi, lekin keyingi misollardan bunga hojat yo‘qligini ko‘rasiz.

II. Amaliy qism

Ketma-ket izlash algoritmlarining yomon holat tahlili.

Ketma-ket izlash algoritmlarida ikkita yomon shartlashgan holat mavjud. Birinchi holat, qidirilayotgan element ro‘yxat oxirida. Ikkinchisi, u umuman ro‘yxatda bo‘lmagan holat hisoblanadi. Bu holatlarga qancha solishtirishlar bajarilishini ko‘ramiz. Biz ro‘yxatdagi barcha kalit so‘zlar unikal deb tasavvur qilgandek, shu sababli agar so‘nggi elementda bir xillik uchrasa oldin qilingan barcha

solishtirishlar natijasiz bo‘lgan. Ammo algoritm oxirgi elementga bormaguncha bu solishtirishlarni davom ettiradi. Natijada N ta solishtirish bajarilgan bo‘ladi, bunda N-ro‘yxat elementlari soni.

Qidirilayotgan element ro‘yxatda mavjud emasligini tekshirish uchun uni barcha element bilan solishtirib chiqishga to‘g‘ri keladi.Agar biz biror-bir elementni tashlab ketsak, qidirilayotgan elementni ro‘yxatda umuman yo‘qligini yoki uni tashlab ketganligimizni aniqlay olmaymiz. Bu shuni ko‘rsatadiki, biror-bir element qidirilayotgan element emasligini aniqlash uchun N ta solishtirish talab qilinadi.

Shuning uchun qidirilayotgan element ro‘yxat oxirida yoki ro‘yxatda umuman yo‘qligini bilish uchun N ta solishtirish bajarish kerak. Ma’lumki N istalgan izlash algoritmining yuqori murakkablik chegarasini beradi, agar solishtirishlar N tadan ortiq bo‘lsa, u holda bu holat qaysidir element bilan solishtirishlar kamida ikki marotaba bo‘lganini, demak ortiqcha ish qilingani va algoritmni yaxshilash mumkinligini bildiradi.

Yuqori murakkablik chegarasi bilan murakkablik tushunchalari o‘rtasida yomon shartlashgan holatlarda farq bor.

Yuqori chegara masala uchun kerakli, yomon holat tushunchasi uchun esa aniq algoritmda hal qiluvchi ahamiyatga ega. Keyingi bo‘limda biz yomon shartlashgan holat yuqori chegara N dan kam bo‘lgan izlash algoritmlari bilan tanishamiz.

Ketma-ket izlash algoritmlarining o‘rta holat tahlili.

Izlash algoritmlarida ikkita o‘rta holat bo‘lishi mumkin. Birinchisida qidiruv har doim muvaffaqiyatli yakunlanadi, ikkinchisida qidirilayotgan qiymat ro‘yxatda mavjud bo‘lmaydi. Agar qidirilayotgan qiymat ro‘yxatda bo‘lsa, u holda N solishtiruvli vaziyatni o‘z ichiga oladi: u birinchi, ikkinchi, uchinchi, to‘rtinchi va hokozo bo‘lishi mumkin. Biz bunday barcha holatlar mumkin deb tasavvur qilamiz, bunda har bir holat 1/N ga teng.

Quyidagi savollarga javob beramiz.

1) *Agar qidirilayotgan qiymat birinchi o‘rinda bo‘lsa, nechta solishtirish bajarish talab qilinadi?*
2) *Ikkinchi o‘rinda bo‘lsachi?*
3) *Agar uchinchida?*
4) *Agar N ta elementdan so‘nggisi bo‘lsachi?*

Agar siz algoritmga diqqat bilan qaragan bo‘lsangiz, unda javoblar 1,2,3 va N ko‘rinishda bo‘lishini aniqlashingiz qiyin emas. Bundan ko‘rinadiki, har bir N holat uchun solishtirishlar soni qidirilayotgan qiymat o‘rniga teng. Natijada murakkabligi o‘rtacha hollar uchun

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

$$= \frac{1}{N} \cdot \frac{N(N+1)}{2}$$ tenglik hosil qilamiz

$$\frac{N+1}{2}.$$

Tasavvur qilamiz, qidirilayotgan qiymat ro‘yxatda uchramasligi mumkin, bunda imkoniyatlar soni N+1 ga o‘sadi. Ko‘rgandikki, qiymat ro‘yxatda bo‘lmaganda uni izlash N ta solishtirishni talab qiladi. Agar barcha N+1 imkoniyatlar bo‘lishi mumkin deb qaralsa, quyidagi kelib chiqadi:

$$A(N) = \frac{1}{N+1}\left(\sum_{i=1}^{N} i + N \right)$$

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

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

$$\approx \frac{N+2}{2}$$

(Qachon N juda katta bo‘lganda, 1/(N+1) 0 ga intiladi.)

Ko‘rinib turibdiki, elementni ro‘yxatda mavjud emasligini hisobga olganda o‘rtacha murakkablik bor yo‘g‘i ½ ga oshadi. Ro‘yxat uzunligi ulkan bo‘lgan holatda ro‘yxat uzunligi bilan bu qiymat solishtirilganda u sezilarli kichikdir.

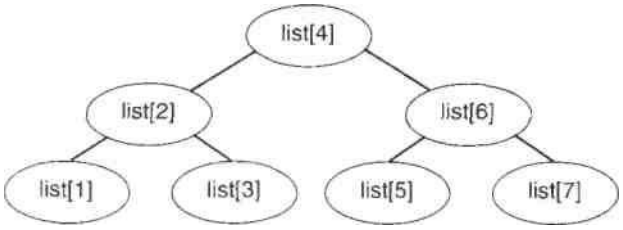
Ikkilanma izlash algoritmlarining yomon holat tahlili.

Oldingi abzatsda biz ro‘yxatni ikkiga bo‘lish ko‘rsatgichi bir qancha takrorlanishlardan keyin birga kamayishini ko‘rsatgan edik. Oxirgi takrorlanish iteratsiyasi bo‘laklar kattaligi 1 ga teng bo‘lganda bajarilishini ham ko‘rgan edik, bu esa j=1 (2^1-1=1 bo‘lganligi uchun) bo‘lganda bajariladi. Bu shuni bildiradiki, N=2^k-1 dan (o‘tishlar soni k) oshmaydi. Oxirgi tenglik k ga bog‘liqligini bilgan holda, yomon shartlashgan vaziyat o‘tishlar soni $k= log_2(N+l)$ ga teng bo‘lganda bajariladi.

Izlash jarayoni tahlilida daraxt yordam berishi mumkin. Daraxt shoxlarida ma’lum o‘tishda tekshiriladigan elementlar turadi. Agar qidirilayotgan qiymat ayni qiymatdan kichkina bo‘lsa chap shoxda, katta bo‘lsa o‘ng shoxda bo‘ladi.

7 ta elementli ro‘yxatning daraxt yechimi 1-rasmda ko‘rsatilgan. Umumiy holatda biz ro‘yxat har xil qismlarining o‘rtasini tanlashga harakat qilamiz, daraxt bu holatda shartli muvozanatlidir.

Agar biz $N = 2^k$—1 deb olsak, bu holatdagi daraxt yechimi har doim to‘liq bo‘ladi. Unda k ta qism bo‘ladi, bunda $k = \log_2(N+ 1)$. Biz har bir qismda bittadan solishtirish bajaramiz, shuning uchun barcha solishtirishlar soni $\log_2(N+1)$ dan oshmaydi.



1-rasm. *Ro‘yxatdagi 7 ta elementdan izlashning daraxt yechimi*

Ikkilanma izlash algoritmining o‘rta holat tahlili.

Ketma-ket qidiruv holati kabi o‘rta holat tahlilida ham ikkita vaziyatga qaraymiz. Birinchi holatda qidirilayotgan qiymat ro‘yxatda aniq mavjud, ikkinchi holatda esa qidirilayotgan qiymat ro‘yxatda mavjud bo‘lmasligi mumkin. Birinchi holatda qidirilayotgan qiymat N ta imkoniyatli vaziyatda. Biz bu barcha holatlarni bir xil 1/N imkoniyatli deb hisoblaymiz. Agar izlashning binar daraxt usulini qaraydigan bo‘lsak, 1 elementli qismida faqat 1 ta solishtirish, 2 elementli qismida 2 ta solishtirish, 3 elementli qismida 3 ta solishtirish bo‘ladi. Umuman, I elementli qismda i ta solishtirish talab qilinadi. 7.3.2 da i qism ta 2^{i-1} shox va N=2^k-1 da daraxt k qismliligi ko‘rsatilgan. Bu shuni ma’lum qiladiki, barcha solishtirishlar sonini quyidagi formula bilan topish mumkin.

$$A(N) = \frac{1}{N}\sum_{i=1}^{k} i2^{i-1} \qquad (N = 2^k - 1 \text{ учун})$$

$$= \frac{1}{N} \cdot \frac{1}{2}\sum_{i=1}^{k} i2^{i}$$

(7.19) dan foydalanib quyidagini hosil qilamiz:

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

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

$$= \frac{1}{N}(k2^k - 2^k + 1)$$

$$= \frac{k2^k - (2^k - 1)}{N}$$

$$= \frac{k2^k}{N} - 1$$

$N = 2^k$-1 bo‘lganda, $2^k = N$+1 o‘rinli. Shuning uchun

$$A(N) = \frac{k(N+1)}{N} - 1$$

$$= \frac{kN + k}{N} - 1$$

N ning oshishi bilan k/N nolga yaqinlashadi va

$A(N) \sim$ k-1($N = 2^k$ -1 uchun);

$A(N) \sim \log_2(N+1)-1$.

Endi ikkinchi holatga qaraymiz, bunda qidirilayotgan qiymat ro‘yxatda yo‘q. Elementning vaziyatlari soni oldingidek N ga teng, bundan tashqari N+1 – vaziyat qidirilayotgan elementni ro‘yhatdan tashqari solishtirish ham bor.

Vaziyatlar soni N+1 teng, chunki qidirilayotgan element ro‘yhatning birinchi elementidan kichkina bo‘lishi, irinchidan katta ikkinchidan kichkina va hokazo, oxiri N dan katta bo‘lishi mumkin. Har bir bu hollar uchun qidirilayotgan element ro‘yxatda mavjud bo‘lmagan holatda k tadan solishtirish mavjud. Barcha imkoniyatlar 2 N+1 ta.

Bundan

$$A(N) = \frac{1}{2N+1}\left(\sum_{i=1}^{k} i2^{i-1} + (N+1)k\right) \quad \text{Nq} 2^k\text{-1 uchun}$$

Yuqoridagi tenglikdan:

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

$$= \frac{((k-1)2^k + 1) + (2^k - 1 + 1)k}{2(2^k - 1) + 1}$$

$$= \frac{(k2^k - 2^k + 1) + 2^k k}{2^{k+1} - 1}$$

$$\approx \frac{k2^{k+1} - 2^k + 1}{2^{k+1}}$$

$$\approx k - \frac{1}{2} = \log_2(N+1) - \frac{1}{2} \qquad N=2^k\text{-1 uchun}$$

Oxirgi qiymat holat natijasini sezilarsiz darajada oshiradi. Misol uchun 2^{20}—1= 1 048 575 ta elementdan tashkil topgan ro‘yxatdan birinchi holatda 19, ikkinchi holatda esa 19,5 natijani oldik.

MAVZU BO‘YICHA TOPSHIRIQLAR

1. Ketma-ket izlashning o‘rta holat bo‘yicha murakkabligi maqsadli qiymat 0.25 ga teng bo‘lganda qanday bo‘ladi. Agar uning qiymati 0.75 ga teng bo‘lsa, ro‘yxatning birinchi yarmida bo‘lish ehtimoli qanday bo‘ladi?
2. Ikkilanma izlash algoritmi uchun 12 elementli ro‘yxatning yechim daraxtini quring. Daraxtning ichki tarmoqlari tekshirilayotgan elementlar bilan belgilangan bo‘lishi, daraxtning chap tarmog‘i maqsadli qiymat tekshirilayotgan elementdan kichik bo‘lgan holni, o‘ng tomoni esa katta bo‘lgan holni ko‘rsatishi kerak.