**№4-ma’ruza. Qidiruv va heshlash algoritmlar.** Chiziqli va binary qidiruv. Hesh jadval va hesh funksiyalar

**REJA**

1. Qidiruv tushunchasi
2. Ketma-ket qidiruv
3. Indeksli ketma-ket qidiruv
4. Binary (ikkilik) qidiruv
5. Hesh jadval va hesh funksiyalar

***Kalitli so‘zlar:*** *qidiruv, jadval, fayl, kalit, noyob kalit, ikkilamchi kalit, tashqi kalit, ichki kalit, ketma-ket qidiruv, indeksli ketma-ket qidiruv, qidiruv samaradorligi, jadvalni qayta tartiblash, transpozisiya, binar qidiruv, mukammal daraxt.*

**1. Qidiruv tushunchasi**

Kompyuterda ma’lumotlarni qayta ishlashda qidiruv asosiy amallardan biri bo‘lib hisoblanadi. Uning vazifasi berilgan argument bo‘yicha massiv ma’lumotlari ichidan mazkur argumentga mos ma’lumotlarni topishdan iborat. Ixtiyoriy ma’lumotlar majmuasi jadval yoki fayl deb ataladi. Ixtiyoriy ma’lumot (yoki tuzilma elementi) boshqa ma’lumotdan biror bir belgisi orqali farq qiladi. Mazkur belgi kalit deb ataladi. Kalit noyob bo‘lishi, ya’ni mazkur kalitga ega ma’lumot jadvalda yagona bo‘lishi mumkin.

Bunday noyob kalitga boshlang‘ich (birinchi) kalit deyiladi. Ikkinchi kalit bir jadvalda takrorlansada u orqali xam qidiruvni amalga oshirish mumkin. Ma’lumotlar kalitini bir joyga yig‘ish (boshqa jadvalga) yoki yozuv sifatida ifodalab bitta maydonga kalitlarni yozish mumkin. Agar kalitlar ma’lumotlar jadvalidan ajratib olinib alohida fayl sifatida saqlansa, u holda bunday kalitlar tashqi kalitlar deyiladi. Aks holda, ya’ni yozuvning bir maydoni sifatida jadvalda saqlansa ichki kalit deyiladi.

Kalitni berilgan argument bilan mosligini aniqlovchi algoritmga berilgan argument bo‘yicha qidiruv deb ataladi. Qidiruv algoritmi vazifasi kerakli ma’lumotni jadvalda topish yoki yo‘qligi aniqlashdan iboratdir. Agar kerakli ma’lumot yo‘q bo‘lsa, u holda ikkita ishni amalga oshirish mumkin:

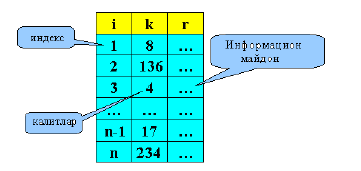
* ma’lumot yo‘qligini indikasiya (belgilash) qilish;
* jadvalga ma’lumotni qo‘yish.

Faraz qilaylik, k - kalitlar massivi. Har bir k(i) uchun r(i) - ma’lumot mavjud. ***Key*** - qidiruv argumenti. Unga ***rec*** - informasion yozuv mos qo‘yiladi. Jadvaldagi ma’lumotlarning tuzilmasiga qarab qidiruvni bir necha turlari mavjud.

**2. Ketma-ket qidiruv**

Mazkur ko‘rinishdagi qidiruv agar ma’lumotlar tartibsiz yoki ular tuzilishi noaniq bo‘lganda qo‘llaniladi. Bunda ma’lumotlar butun jadval bo‘yicha operativ xotirada kichik adresdan boshlab, to katta adresgacha ketma-ket qarab chiqiladi.

Massivda ketma-ket qidiruv (search o‘zgaruvchi topilgan element raqamini saqlaydi).



*1-Chizma. Ketma-ket qidiruv funksiyasi va dasturi.(C++da)*

C++ tilida dastur quyidagicha bo‘ladi:

*#include <iostream.h>*

*#include <conio.h>*

*int search(int a[], int N, int key)*

*{*

*int i = 0;*

*while (i!=N) if (a[i]==key) return i;*

*else i++; return -1;*

*}*

*main (){*

*int i, N, mas[1000], key, P; cout<<"Masiv uzunligini kiriting!"<<endl; cin>>N;*

*cout<<"Massiv elementlarini kiriting!"<<endl;*

*for (i=0; i<N; i++)*

*cin>>mas[i];*

*cout<<"Qidiruv elementini kiriting!"<<endl; cin>>key;*

*P=search(mas,N,key);*

*if (P==-1) cout<<"Bunday element massivda yoq"; else cout<<"Bunday element bor"<<P+1; getch(); return 0;*

*}*

Massivda ketma-ket qidiruv algoritmi samaradorligini bajarilgan taqqoslashlar soni M bilan aniqlash mumkin. Mmin = 1, Mmax = n. Agar ma’lumotlar massiv yacheykasida bir hil extimollik bilan taqsimlangan bo‘lsa, u holda Msr » (n + 1)/2 bo‘ladi.

Agar kerakli element jadvalda bo‘lmasa va mazkur elementni jadvalga qo‘shish lozim bo‘lsa, u holda yuqoridagi dasturdagi oxirgi ikkita operator quyidagiga almashtiriladi.

*n:=n+1;*

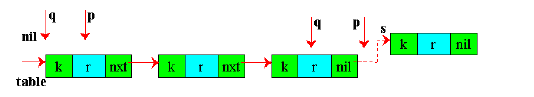
*k[n]:=key;*

*r[n]:=rec;*

*search:=n;*

*exit;*

Agar ma’lumotlar jadvali bir bog‘lamli ro‘yxat ko‘rinishida berilgan bo‘lsa, u holda ketma-ket qidiruv ro‘yxatda amalga oshiriladi.



Algoritmlar varianti:

*#include <iostream.h>*

*#include <conio.h>*

*#include <alloc.h>*

*#include <stdlib.h> struct TNode { int value;*

*TNode\* pnext;*

*TNode(int val): pnext(0), value(val) {}*

*};*

*//Ro‘yhatga element qo‘shish void add2list(TNode \*\*pphead, int val) { TNode \*\*pp = pphead, \*pnew;*

*//while(\*pp) {*

*//if(val < (\*pp)->value)*

*// break;*

*//else*

*// pp = &((\*pp)->pnext);*

*//}*

*pnew = new TNode(val); pnew->pnext = \*pp;*

*\*pp = pnew;*

*}*

*//Ro‘yhat elementlarini ekranga chiqarish void print(TNode \*phead) {*

*TNode\* p = phead; while(p) {*

*cout <<""<< p->value<<"|" <<"*

*"<<"|"<<p<<"|"<< "—> ";*

*p = p->pnext;*

*}*

*cout << endl;*

*}*

*// Ro‘yhatda element qidirish, C++*

*TNode\* Find(TNode \*phead, int x)*

*{*

*TNode \*p=phead; while(p)*

*if (p->value==x) return p; else p = p->pnext; return 0;*

*}*

*//Ro‘hat elementini o‘chirish void deleteList(TNode \*phead) { if(phead) {*

*deleteList(phead->pnext); if(phead) delete phead;*

*}*

*}*

*//Asosiy qism*

*int main() {int mas[1000], N, key; TNode\* T; clrscr();*

*TNode \*phead = 0;*

*//srand(time(0));*

*cout<<"Royhat uzunligini kirit"<<endl; cin>>N;*

*cout<<"Elementlarni kirit!"<<endl; for(int j=0; j<N; j++) cin>>mas[j];*

*for(int i = 0; i < N; ++i) add2list(&phead,mas[i]);*

*cout<<"Qidiruv elementni kiriting!"<<endl; cin>>key;*

*// rand() % 100); cout << "Elements of the list:" << endl; print(phead);*

*T=Find(phead,key);*

*if (T==0) cout <<"Bunday element yoq"<<endl; else cout <<"Bunday element bor"<<" "<<T- >value<<" "<<T<<endl;*

*//deleteList(phead);*

*getch();*

Ro‘yxatli tuzilmaning afzalligi shundan iboratki, ro‘yxatga elementni qo‘shish yoki o‘chirish tez amalga oshadi, bunda qo‘shish yoki o‘chirish element soniga bog‘liq bo‘lmaydi, massivda esa elementni qo‘shish yoki o‘chirish taxminan barcha elementlarni yarimini siljitishni talab qiladi. Ro‘yxatda qidiruvni samaradorligi taxminan massivniki bilan bir hil bo‘ladi.

Umuman olganda ketma-ket qidiruv samaradorligini oshirish mumkin.

Faraz qilaylik, kun davomida ma’lumotlar yig‘ilib, kechqurun ular qayta ishlansin. Ma’lumotlar to‘plangandan keyin ular saralanadi.

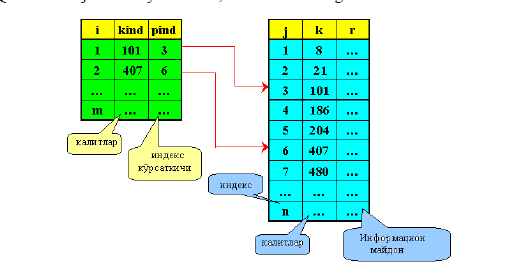
**3. Indeksli ketma-ket qidiruv**

Mazkur ko‘rinishdagi qidiruv amalga oshirilayotganda ikkita jadval tashkil qilinadi: o‘z kalitiga ega ma’lumotlar jadvali (o‘sish tartibida tartiblangan) va indekslar jadvali, bu xam ma’lumotlar kalitidan iborat-u, lekin bu kalitlar asosiy jadvaldan aniq bir interval orqali olingan. (2-chizma).

Boshida berilgan argument bo‘yicha ketma-ket qidiruv indekslar jadvalida amalga oshiriladi. Qachonki, biz berilgan kalitdan kichik kalitni aniqlaganimizda, asosiy jadvalda qidiruvni quyi chegarasini o‘rnatamiz - ***low***, keyin esa yuqori chegarani - ***hi***, ya’ni *(kind > key ).*

Masalan, *key = 101.*

Qidiruv to‘la jadval bo‘yicha emas, balki low dan hi gacha davom etadi.



*2-Chizma. Indeksli ketma-ket qidiruv funksiyasi va dasturi.(C++da)*

**Misol:**

*#include <iostream.h>*

*#include <conio.h>*

*int InSeqsearch(int realArray[], int kind[2][1000],int m,int key, int \*t)*

*{*

*int i=0,*

*low = 0, hi = 0;*

*while ((i<m) && (kind[0][i]<=key)) {*

*i + +;*

*(\*t)++;*

*}*

*(\*t)++;*

*if (i==0) low=0; else*

*low=kind[1][i-1]; if (i==m) hi=N; else*

*hi=kind[1][i]-1; for (int j=low; j<=hi; j++)*

*{*

*(\*t)++;*

*if( key==realArray[j] )*

*{*

*return j;*

*}*

*}*

*return -1;*

*}*

*main ()*

*{*

*int i = 0 ,*

*N = 0,*

*mas[1000] = {0}, kind[2][1000] = {0}, key = 0,*

*P = 0, index = 0, kindIndex = 0, t = 0;*

*cout<<endl<<"Masiv uzunligini kiriting!"<<endl; cin>>N;*

*cout<<"Massiv elementlarini kiriting!"<<endl; for (i=0; i<N; i++) cin>>mas[i]; cout<<"Qidiruv elementini kiriting!"<<endl; cin>>key;*

*cout<<"Boshlangich qadamni kiriting! "<<endl; cin>>P;*

*i = P-1;*

*while(i<N)*

*{*

*kind[0][kindIndex] = mas[i]; kind[1][kindIndex++] = i;*

*i += P;*

*}*

*index = InSeqsearch(mas,N,kind,kindIndex,key,*

*&t) ;*

*if (index == -1)*

*cout<<"Bunday element massivda yuq "<< index <<" "<<t<<endl; else*

*cout<<"Bunday element bor"<<" "<<index+1<<" "<<t<<endl;*

*getch(); return 0;*

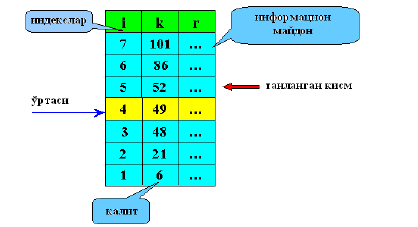
*}*

**4. Binary (ikkilik) qidiruv**

Faraz qilaylik, o‘sish tartibida tartiblangan sonlar massivi berilgan bo‘lsin. Ushbu usulni asosiy g‘oyasi shundan iboratki, tasodifiy qandaydir AM element olinadi va u X qidiruv argumenti bilan taqqoslanadi. AgarAM=X bo‘lsa, u holda qidiruv yakunlanadi; agar AM <X bo‘lsa, u holda indekslari M dan kichik yoki teng bo‘lgan barcha elementlarni kelgusi qidiruvdan chiqarib yuboriladi. Xuddi shuningdek, agar AM >X bo‘lsa.

M ixtiyoriy tanlanganda ham taklif qilinayotgan algoritm korrekt ishlaydi. Shu sababali M ni shunday tanlash lozimki, tadqiq qilinayotgan algoritm samaraliroq natija bersin, ya’ni uni shunday tanlaylikki, iloji boricha kelgusi jarayonlarda ishtirok etuvchi elementlar soni kam bo‘lsin. Agar biz o‘rtacha elementni, ya’ni massiv o‘rtasini tanlasak yechim mukammal bo‘ladi.

Quyidagi chizma ko‘rinishida berilgan massivni qarab chiqaylik. Faraz qilaylik, bizdan kaliti 52 ga teng bo‘lgan elementni topish talab qilinsin. Dastlabki taqqoslanadigan element 49 bo‘ladi. 49<52 bo‘lgani uchun sababli, keying qidiruv 49 dan yuqorida turgan elementlar orasida amalga oshiriladi. Yangi hosil bo‘lgan massiv o‘rtasi 86. Agar berilgan kalit bilan ushbu kalitni taqqoslasak 86>52 bo‘ladi. Demak, navbatdagi qidiruvlar 86 bilan 49 orasidagi elementlar ichida amalga oshirilishi lozim. Keyingi qadamda ma’lum bo‘ldiki, qaralayotgan elementlar o‘rtasidagi element kaliti 52 ga teng. Shunday qilib, massivda berilgan kalitga teng bo‘lgan elementni topdik.



*3-chizma. Binary (ikkilik) qidiruv*

*Yuqoridagi algoritmni amalga oshirish dasturi S++ tilida quyidagicha bo‘ladi:*

*#include <iostream.h>*

*#include <conio.h>*

*int Binsearch(int a[], int N, int key, int \*t)*

*{*

*int l=0, r=N-1, mid=(l+r)/2;*

*while (l<=r)*

*{ \*t+=1;*

*if (a[mid]==key) return mid;*

*if (a[mid]>key) r=mid-1;*

*else l=mid+1;*

*mid=(l+r)/2;*

*}*

*a[N]=key;*

*return N;*

*}*

*main ()*

*{*

*int i, N, mas[1000], key, P, t=0;*

*cout<<endl<<"Masiv uzunligini kiriting!"<<endl;*

*cin>>N;*

*cout<<"Massiv elementlarini kiriting!"<<endl;*

*for (i=0; i<N; i++)*

*cin>>mas[i];*

*cout<<"Qidiruv elementini kiriting!"<<endl;*

*cin>>key;*

*P=Binsearch(mas,N,key, &t);*

*if (P==N) cout<<"Bunday elementni massivga*

*qo‘shis lozim"<<" "<<P+1<<" "<<t<<endl;*

*else cout<<"Bunday element bor"<<" "<<P+1<<"*

*"<<t<<endl;*

*getch();*

*return 0;*

*}*

Agar key = 101 bo‘lsa, kerakli yozuv 3 marta taqqoslashlarda aniqlanadi (ketma-ket qidiruvda taqqoslashlar soni 7 ta bo‘lar edi). Agar S – taqqoslashlar soni va n – jadvaldagi elementlar soni bo‘lsa, u holda

S = log2n.

Masalan, n = 1024.

Ketma-ket qidiruvda S = 512, binar qidiruvda S = 10.

Agar katta xajmdagi ma’lumotlar ichida qidiruv amalga oshirilayotgan bo‘lsa, u holda binar va indeksli ketma-ket qidiruvni umumlashtirib olib borish mumkin. Sababi, har ikkala qidiruv ham tartiblangan massivda amalga oshiriladi.

**5. Hesh jadval va hesh funksiyalar**

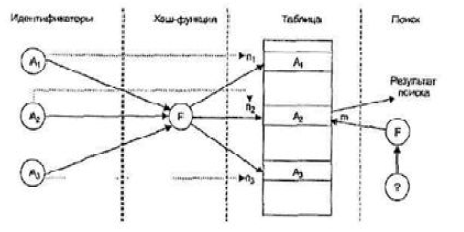
*Joylashtirish usuli (heshlashtirish)* ma’lumotlar tuzilmasida element joylashgan o‘rinni tez aniqlashga yo‘naltirilgan usuldir. Joylashtirish usulida ma’lumotlar oddiy massiv sifatida ifodalangan bo‘ladi.

Elementni jadvalga qo‘shishdan oldin uning adresi hesh-funksiya orqali aniqlanadi: A = h(K), bu yerda K - kalit, A - jadvaldagi element adresi bo‘lib, 0 £ A £ N-1, shart o‘rinli bo‘ladi.

***F hesh-funksiya deb*** R kiruvchi elementlar to‘plamini manfiy bo‘lmagan butun sonlar to‘plami Z ga o‘girishga aytiladi. **Z:F(r)=n, reR, neZ.**

Hesh-adreslash bu hesh-funksiya qiymatlar soxasini qandaydir bir ma’lumotlar massivining yacheykasi adresi sifatida foydalanishdan iborat. U holda ma’lumotlar massivi o‘lchami foydalanilayotgan hesh-funksiyaning qiymatlar soxasiga mos kelishi kerak.

Turli A1, A2, A3 identifikatorlar uchun mos ravishda n1, n2, n3 hesh- funksiya qiymatlari to‘g‘ri kelsin. n1, n2, n3 adreslarga mos yacheykalarda A1, A2, A3 identifikatorlar haqida ma’lumot joylanadi. A3 identifikatorni qidirishda n3 adres qiymati hisoblanadi va tegishli jadval yacheykasidan ma’lumotlar tanlanadi.



*4-chizma. Hesh funksiya*

***Akslantirish funksiyasini tanlash.*** Bu metod juda effektiv, elementlari jadvalga joylash vaqti xam, qidiruv vaqti ham faqat hesh-funksiyani hisoblashga ketadi.

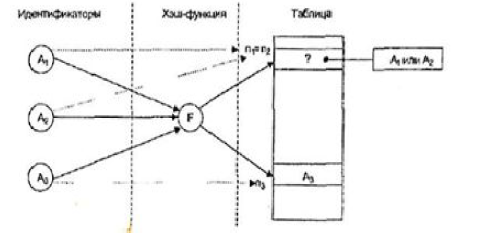
Bu usulning 2 ta yaqqol kamchiligi bor. Ulardan biri: identifikatorlar jadvalining xotira xajmidan unumsiz foydalanilishi. Massiv o‘lchami hesh- funksiya qiymatlar soxasiga mos kelishi kerak, ayni vaqtda real holatda jadvalda saqlanayotgan identifikatorlar ancha kam bo‘lishi mumkin. Ikkinchi kamchiligi mos keluvchi hesh-funksiyani tanlay bilish.

Hesh-funksiyadan natija olish - “heshlash” simvollar zanjiri ustida oddiy arifmetik va mantiqiy amallarni bajarish hisobiga erishiladi.

Hesh-adreslashda identifikatorlar jadvalining bir yacheykasiga 2 ta turli xil bo‘lgan identifikatorlar joylashishi mumkin emas. Bu vaziyat, ya’ni 2 yoki undan ortiq identifikatorlar hesh funksiyaning bir xil qiymatiga ega bo‘lish xodisasi kolliziya deb nomlanadi.

Kolliziyaning yuzaga kelishi 2 ta har xil identifikator A1 va A2larning hesh-funksiya qiymatlari nl va n2 bir xil (n1=n2) bo‘lishi hisoblanadi.

***Ziddiyatni hal qilish algoritmlari.*** Kolliziya ro‘y berishini butunlay oldini oladigan, yaxshi hesh-funksiyani qurish mumkinmi?



*5-chizma. Hesh jadval tuzilishi*

Aniqki, butunlay kolliziyaga uchramasligi uchun hesh-funksiyaning har bir natijaviy qiymati unikal bo‘lishi kerak.

Kolliziya muammosini yechish uchun turli usullarni qo‘llash mumkin. Ulardan biri “reheshlash” metodi hisoblanadi.

Bu metodga ko‘ra, A element uchun hesh-funksiya orqali hisoblangan h(A) adresi band bo‘lgan yacheykani ko‘rsatsa, unda n1=h1(A) funksiya qiymatini hisoblash zarur va n1 adresga tegishli yacheykani bandligini tekshirish kerak.

Hi(A) funksiyani hisoblashni eng oddiy metodi, uni hi(A)=(h(A)+pi)modNm asosida qurishdir, bu yerda ***pi*** qandaydir bir hisoblangan butun son, Nm -identifikatorlar jadvalidagi elementlarning maksimal soni. O‘z o‘rnida eng oddiy usul pi ni o‘rniga i ni qo‘yish bo‘ladi. Unda quyidagi formulani olamiz hi(A)=(h(A)+i)modNm. Bu holda hesh- funksiyaning bir xil qiymatlariga mos kelgan identifikatorlarni joylash uchun bo‘sh yacheykani qidirish mantiqan hesh-funksiya h(A) ko‘rsatgan joydan boshlanadi.

**Nazorat savollari**

1. Qidiruv vazifasi nimadan iborat?
2. Noyob kalit deganda nimani tushunasiz?
3. Ro‘yxatda berilgan kalitli element yo‘q bo‘lganda qaysi amal bajariladi?
4. Ketma-ket qidiruv va indeksli ketma-ket qidiruvlarning farqi nimadan iborat?
5. Ulardan qaysi biri samaraliroq va nima sababdan?
6. Jadvalni qayta tartiblashning qanday usullarini bilasiz?