

Binarno iskanje

Binarno iskanje je algoritem za iskanje indeksa/ključa v urejeni polju podatkov. Iskanje poteka tako, da se pri vsakem noven povpraševanju izloči polovico podatkov.

Primer binarnega iskanja: zamislimo si neko naključno število in prijatelju rečemo, naj jo ugiba. Povemo mu, da je to število med 1 in 50. Prijatelj bo nato spraševal: "Ali je večje od 20?" in mi mu rečemo da. Prijatelj bo sedaj vedel da mora ugibati število, ki je nad 20, zato bo vse tiste, ki so manjše izločil. Če rečemo, da je število manjše od 35 bo vedel, da mora izločiti vsa števila nad 34. To se nadaljuje tako dolgo, dokler končno ne najde števila.

Zakaj uporabimo binarno iskanje: Z binarnim iskanjem hitreje najdemo željeni podatek kot s sekvenčnim iskanjem (iščemo od prvega vse dokler ne najdemo našega podatka). Predpogoj je le, da je tabela urejena.

Primer:

Imamo tabelo z naslednjimi vrednostmi

2	5	7	9	10	12	17	29
---	---	---	---	----	----	----	----

Iščemo podatek 12 znotraj tabele.

Postopek ponavljamo doker leva in desna stran se ne ujemata. Na začetku je leva stran na indeksu 0 in desna stran n-1.

Najdemo sredino – $(\text{levo} + \text{desno}) / 2 \Rightarrow$ v našem primeru je $0 + 7 / 2 = 3$

Preverimo, če je na sredini naš podatek – 7 ni enako 12

Preverimo, če je naš podatek večji od podatka na sredini, premaknemo meje:

če je podatek večji od podatka na sredini \Rightarrow imamo novo levo stran \Rightarrow sredina + 1

če je podatek manjši od podatka na sredini \Rightarrow imamo novo desno stran \Rightarrow sredina - 1

Vse skupaj ponavljamo dokler ne najdemo podatka, oz ga ne najdemo in vrnemo podatek o napaki!

2	5	7	9	10	12	17	29
---	---	---	---	----	----	----	----

^ (ni enako 12 in je manjše.. nova leva stran

9	10	12	17	29
---	----	----	----	----

^ (enako našemu iskanemu

podatku) – podatek smo našli na poziciji 5 in to v drugem koraku.

Koda:

```
// funkcija vrne lokacijo ključa, v tabeli
// če vrednosti ne najdemo nam funkcija vrne -1
int binarno_iskanje(int *t, int n, int pod)
{
    int levo, desno, sredina;
    levo = 0;
    desno = n - 1;
    while (levo <= desno)
    {
        sredina = (levo + desno) / 2;
        if (pod == t[sredina])
        {
            return sredina;
        }
        else if (pod > t[sredina])
            levo = sredina + 1;
        else
            desno = sredina - 1;
    }
    return -1;
}
```

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void vnesi(int *t,int n)
{
    int i;
    for (i=0;i<n;i++)
    {
        t[i] = rand() % 15 + 35;
    }
}

void izpis(int *t, int n)
{
    int i;
    for (i=0;i<n;i++)
        printf("%d ",t[i]);
    printf("\n");
}

void uredi(int *x, int n)
{
    int i,j,pom;
    for (i=0;i<n-1;i++)
        for (j=n-1;j>0;j--)
        {
            if (x[j-1]>x[j])
            {
                pom=x[j-1];
                x[j-1]=x[j];
                x[j]=pom;
            }
        }
}

int binarno_iskanje(int *t, int n, int pod)
{
    int levo, desno, sredina;
    levo = 0;
    desno = n - 1;
    while (levo <= desno)
    {
        sredina = (levo + desno) / 2;
        if (pod == t[sredina])
        {
            return sredina;
        }
        else if (pod > t[sredina])
            levo = sredina + 1;
        else
            desno = sredina - 1;
    }
    return -1;
}

int main(void)
{
    int t[20];
    int p;
    srand(time(NULL));
    vnesi(t,20);
    izpis(t,20);
    uredi(t,20);
    izpis(t,20);
    printf("Podatek 40 se nahaja na indeksu %d\n",binarno_iskanje(t,20,40));
    return 0;
}

```