Pretraživanje

Pretraživanje je proces koji za cilj ima pronalaženje elementa niza, liste, stabla ili neke druge strukture podataka, koji zadovoljava unapred definisani kriterijum. Najčešće je to traženje elementa koji sadrži određeni podatak, koji nazivamo **ključ**. U zavisnosti od tipa strukture podataka u kojoj se vrši pretraživanje, algoritmi mogu biti manje ili više kompleksni.

U nastavku ćemo dati neke osnovne algoritme za pretraživanje nizova, lista i binarnih stabala.

Sekvencijalno pretraživanje

Sekvencijalno pretraživanje je sigurno najjednostavniji algoritam za pretraživanje nizova i lista. Ovaj algoritam se naziva još i **linearno pretraživanje** iz razloga što se traženje elementa vrši tako što se ispituje jedan po jedan element strukture podataka sve dok se ne nađe odgovarajući element ili se ne dođe do kraja strukture.

Sekvencijalno pretraživanje niza

Posmatrajmo niz celih brojeva *a* dužine *n*. U nastavku je data funkcija koja pronalazi indeks traženog elementa *t* unutar niza *a*. Ukoliko niz ne sadrži traženi element, funkcija će vratiti -1.

```
int trazi(int a[],int n,int t)
{
   int i;

   for(i=0;i<n;i++)
      if(a[i]==t) return(i);

   return(-1);
}
```

ili korišćenjem **while** petlje:

```
int trazi(int a[],int n,int t)
{
   int i;

   i=0;

   while(i<n)
   {
      if(a[i]==t) return(i);
      i++;
   }

   return(-1);
}
```

Iako u implementaciji sa **for** petljom to možda i nije tako očigledno, u primeru sa **while** petljom se jasno vidi da se u svakoj iteraciji obavljaju dve provere: 1) da li je *i* manje od *n* i 2) da li je *a[i]==t*. Proveravenje prvog uslova je moguće izbeći dodavanjem traženog elementa iza poslednjeg elementa u nizu, tako da algoritam izgleda ovako:

```
int trazi(int a[],int n,int t)
{
   int i;

   a[n]=t;
   i=0;

   while(a[i]!=t) i++;

   if(i<n) return(i);

   return(-1);
}
```

Dodavanje traženog elementa iza poslednjeg elementa u nizu obezbeđuje izlazak iz **while** petlje ukoliko algoritam ne pronađe traženi element među prvih *n* elemenata niza. Ukoliko je nakon izlaska iz petlje indeks *i* manji od dužine niza *n*, to je znak da je traženi element nađen unutar prvobitnog niza na poziciji *i*. U suprotnom je jasno da je algoritam pronašao veštački dodat element, što je pokazatelj da u prvobitnom nizu nije bilo traženog elementa.

Binarno pretraživanje

Ukoliko sa sigurnošću znamo da je niz koji pretražujemo uređen u rastućem ili opadajućem poretku, onda za pretraživanje možemo iskoristiti neki od algoritama koji koristi ovu činjenicu kako bi skratio vreme traženja. Jedan od takvih algoritama je **binarno pretraživanje**. Ovaj algoritam vrši pretraživanje niza tako što prvo upoređuje traženi element sa središnjim elementom niza. Ukoliko središnji element nije jednak traženom elementu, zahvaljujući činjenici da je niz rastući ili opadajući, algoritam određuje u kojoj polovini (podnizu) se može naći traženi element, a drugu polovinu odbacuje. Isti proces se dalje ponavlja na dobijenoj polovini, sve dok se ne pronađe traženi element ili dok u podnizu nakon deljenja ne ostane ni jedan element, što bi značilo da niz ne sadrži traženi element.

Naredna funkcija vrši binarno pretraživanje u rastućem nizu:

```
int trazi(int a[],int n,int t)
{
   int donji,gornji,srednji;

   donji=0;
   gornji=n-1;

   while(donji<=gornji)
   {
      srednji=(donji+gornji)/2;

      if(a[srednji]==t) return(srednji);

      if(t<a[srednji])  gornji=srednji-1;
      else              donji=srednji+1;
   }

   return(-1);
}
```

Na početku se posmatra ceo niz, tako da su početne vrednosti donje i gornje granice jednake nultom i poslednjem indeksu niza. Zatim se određuje središnji element i ispituje da li je on jednak traženom elementu. Ukoliko je to traženi element, funkcija vraća njegov indeks. U suprotnom, proverava se da li je traženi element manji ili veći od središnjeg elementa. Ako je traženi element manji, odbacuje se gornji podniz tako što se gornja granica pomera na prvu poziciju ispod središnjeg elementa. U suprotnom, odbacuje se donji podniz tako što se donja granica pomera na prvu poziciju iznad središnjeg elementa.

Nakon toga čitav postupak se ponavlja na odabranom podnizu sve do pronalaženja traženog elementa ili do trenutka kada podniz više ne sadrži ni jedan element. Ukoliko je niz koji se pretražuje opadajući, izbor polovine niza koja se dalje pretražuje se obavlja na suprotan način nego što je to u slučaju rastućeg niza.