Pretraživanje

Sortiranje je proces u kome se elementi niza ili liste uređuju tako da čine rastući ili opadajući niz podataka. Rastući niz je takav da je svaki element veći od svog prethodnika, a manji od svog sledbenika. Nasuprot tome, svaki element opadajućeg niza je manji od svog prethodnika, a veći od svog sledbenika.

Sortiranje izborom uzastopnih minimuma – Selection sort

Jedan od najjednostavnijih algoritama za sortiranje niza u rastući poredak je svakako **izbor uzastopnih minimuma**, poznat i kao **selection sort**. Ovaj algoritam uređuje niz u rastući tako što traži najmanji element i smešta ga na početak niza. Zatim se za ostatak niza ponavlja isti postupak, sve dok se ne dođe do podniza koji ima samo jedan element.

```
void sort(int a[],int n)
{
   int i,j,pom;

   for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
         if(a[j]<a[i])
         {
            pom=a[i];
            a[i]=a[j];
            a[j]=pom;
         }
}
```

Sortiranje deljenjem niza - Quick sort

Quick sort je algoritam za sortiranje niza tehnikom "podeli i osvoji". Kao što i sam naziv tehnike "podeli i osvoji" nagoveštava, ovaj algoritam sortiranje obavlja iz dva dela. U prvom delu cilj je rasporediti članove niza tako da svi elementi manji od unapred izabranog elementa budu sa njegove leve strane, a svi elementi veći od tog elementa budu sa njegove desne strane. Element u odnosu na koji vršimo raspoređivanje može biti bilo koji element niza (najčešće prvi ili poslednji zbog jednostavnosti) i njega nazivamo **pivot**.

Postupak u kome se elementi niza razvrstavaju na manje od pivota i veće od pivota se može obaviti na sledeći način:

1. Neka je prvi element niza izabran za pivot
2. Posmatramo redom sve elemente niza i proveravamo da li su veći ili manji od prvog elementa
3. Ukoliko je posmatrani element veći od prvog elementa, nema potrebe za njegovim premeštanjem jer se on već nalazi desno od pivota. Međutim, ukoliko je posmatrani element manji od prvog elementa on treba da bude prebačen u grupu elemenata manjih od pivota. To znači da će se pozicija pivota pomeriti jedno mesto u desno, a posmatrani element će zameniti mesto sa elementom koji je bio na toj poziciji, s obzirom da je on veći od pivota.
4. Nakon prolaska kroz sve elemente niza određena je konačna pozicija pivota i ostaje samo da se prvi element niza postavi na to mesto tako što će zameniti poziciju sa elementom koji je bio na mestu pivota.

Da bismo bolje razumeli prethodno opisani postupak, posmatrajmo stanja kroz koja prolazi sledeći niz:

1. element niza je izabran za pivot	10	7	12	13	4	5	19	11	6	17
Analiziranje 2. elementa (7) i zamena sa 2. (7)	10	7	12	13	4	5	19	11	6	17
Analiziranje 3. elementa (12)	10	7	12	13	4	5	19	11	6	17

Analiziranje 4. elementa (13)	10	7	12	13	4	5	19	11	6	17
Analiziranje 5. elementa (4) i zamena sa 3. (12)	10	7	4	13	12	5	19	11	6	17
Analiziranje 6. elementa (5) i zamena sa 4. (13)	10	7	4	5	12	13	19	11	6	17
Analiziranje 7. elementa (19)	10	7	4	5	12	13	19	11	6	17
Analiziranje 8. elementa (11)	10	7	4	5	12	13	19	11	6	17
Analiziranje 9. elementa (6) i zamena sa 5. (12)	10	7	4	5	6	13	19	11	12	17
Analiziranje 10. elementa (17)	10	7	4	5	6	13	19	11	12	17
Zamena zbog postavljanja pivota na novu poziciju	6	7	4	5	10	13	19	11	12	17

Primećujemo da su po završetku čitavog postupka svi elementi manji od pivota pozicionirani sa njegove leve strane, a svi elementi veći od pivota sa njegove desne strane. U nastavku je data funkcija *podeli* koja obavlja prethodno navedeni postupak:

```
int podeli(int a[],int donji,int gornji)
{
   int i,pivot;

   pivot=donji;

   for(i=donji+1;i<=gornji;i++)
      if(a[i]<a[donji])
      {
         pivot++;
         zameni(a,i,pivot);
      }

   zameni(a,donji,pivot);

   return(pivot);
}
```

pri čemu funkcija *zameni* zamenjuje mesta navedenim članovima niza i definisana je kao:

```
void zameni(int a[],int i,int j)
{
   int pom;

   pom=a[i];
   a[i]=a[j];
   a[j]=pom;
}
```

Sada kada smo niz organizovali na takav način da su svi elementi manji od pivota sa njegove leve strane, a svi veći od pivota sa njegove desne strane, čitav postupak možemo ponoviti posebno na levom i desnom podnizu. Ponavljanje ovog postupka vršimo sve dok ne dobijemo podniz koji se sastoji od jednog ili dva elementa. U slučaju da podniz ima najviše jedan element, sortiranje nije potrebno. Ukoliko, pak, podniz sadrži dva elementa, sortiranje se svodi na eventualnu zamenu mesta ta dva elementa. U nastavku je data rekurzivna funkcija *quick* koja obavlja navedeni postupak:

```
void quick(int a[],int donji,int gornji)
{
   int pivot;

   // podniz ima jedan ili ni jedan element
   if(donji>=gornji) return;

   // podniz ima dva elementa
```

```
   if(donji+1==gornji)
   {
      if(a[donji]>a[gornji]) zameni(a,donji,gornji);
      return;
   }

   // podniz ima vise od dva elementa
   pivot=podeli(a,donji,gornji);
   quick(a,donji,pivot-1);
   quick(a,pivot+1,gornji);
}
```

S obzirom da krajnji korisnik funkcije za sortiranje očekuje da parametri funkcije budu samo niz i dužina niza, možemo napisati funkciju koja ima ovakve parametre i koja na odgovarajući način poziva rekurzivnu funkciju *quick*:

```
void sort(int a[],int n)
{
   quick(a,0,n-1);
}
```