ALGORITMI I STRUKTURE PODATAKA:

QUICK-SORT ALGORITAM

Nenad Katanić

Nešto o QuickSort-u...

- Jedan od najbržih danas poznatih algoritama za sortiranje
- Relativno jednostavan u teoriji, ali jako kompliciran za prebacivanje u kôd
- Najčešće se koristi za uzastopno sortiranje, sortiranje srednje velikih i velikih listi (polja) i u slučaju kad nismo sigurni koji algoritam upotrijebiti
- Postupak sortiranja se svodi na dijeljenje niza u dva dijela:
  - 1.) svi elementi koji su manji ili jednaki nekoj unaprijed odabranoj vrijednosti (nekom elementu niza) → *STOŽER*
  - 2.) svi elementi veći od te iste vrijednosti
- Dijeljenje se izvodi tako što uzimamo jedan po jedan element niza i dodajemo ga odgovarajućem dijelu
- Tako u svakom trenutku možemo razlikovati 3 dijela:
  - 1.) elemente koje još nismo obradili (od *(i+1)*-og do posljednjeg
  - 2.) grupa manjih ili jednakih stožeru (od nultog do *j*-tog)
  - 3.) grupa elemenata većih od stožera (od *(j+1)*-og do *i*-tog člana)
- Nakon obrade svih elemenata niz se sastoji iz samo 2 dijela: grupe manjih ili jednakih i grupe većih (grupa neobrađenih je postala prazna)

Opis rada algoritma

- Ako slijedeći elemet *(i+1)*-vi pripada grupi manjih ili jednakih, prvi element iz grupe većih *(j+1)*-vi zamjenjuje mjesto sa analiziranim elementom → tako je *(i+1)*-vi element dodan grupi manji ili jednakih i ona postaje veća za jedan element
- Ako slijedeći element pripada grupi većih, tada se samo grupa većih uvećava za jedan, dok grupa manjih ostaje ista (ne mijenja se vrijednost varijable j)

Rješenje u C-u...

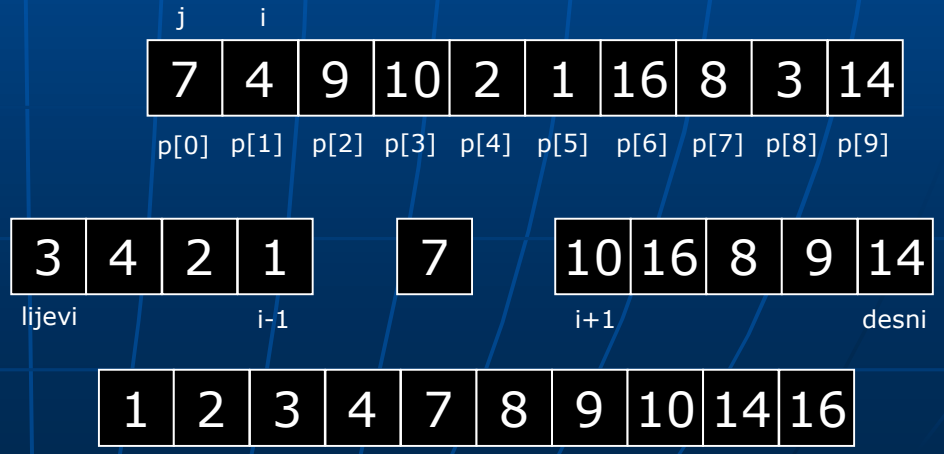
```
int podijeli (int lijevi, int desni, int polje[]) {
        int i,j;
        for (j = lijevi, i = lijevi+1; i <= desni; i++)
                if (polje[i] <= polje[lijevi])
                        zamijeni (&polje[++j], &polje[i]);
        zamijeni (&polje[lijevi], &polje[j]);
        return(j);
}
```

```
void zamijeni (int *a, int *b){
        int pom;
        pom = *a;
        *a = *b;
        *b = pom;
}
```

```
void quick_podniz (int lijevi, int desni, int polje[]) {
        int i,j;
        /* sortiranje jednoelementnog niza */
        if (lijevi >= desni) return;
        /* sortiranje dvoelementnog niza */
        if (lijevi + 1 == desni) {
                if (polje[lijevi] > polje[desni])

        zamijeni(&polje[lijevi], &polje[desni]);
                return;
        }
        i = podijeli (lijevi, desni, polje);
        quick_podniz (lijevi, i-1, polje);
        quick_podniz (i+1, desni, polje);
 }
```

```
void quicksort (int polje[], int duzina) {
        quick_podniz (0, duzina-1, polje);
}
```



Složenost algoritma

- QuickSort algoritam u prosjeku (average-case) ima složenost O(n*logn), a u najgorem slučaju složenost $O(n^2)$
- Funkcija *podijeli* uspoređuje svaki element podniza sa stožernim elementom i stoga ima funkciju složenosti f(n) = n
- U najgorem slučaju nakon svake podjele niza dobivamo dva podniza od kojih jedan sadrži samo jedan element
  - Ako se ovo dogodi prilikom svakog poziva funkcije *podijeli* tada se uspoređivanje vrši na poljima dužine n, zatim (n-1), (n-2) ... Itd.. Stoga imamo:

$$n + (n-1) + (n-2) + ... + 1 = \frac{n*(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} \Rightarrow O(n^2)$$

  - Ovo se događa kada za stožerni element izaberemo prvi element polja, a samo polje je već sortirano
- Kada algoritam zadani problem uza
složenost O(logn) (jer je vrijeme iz
broju podjela broja n sa 2)
- QuickSort algoritam ovakvu podjelu
složenost O (n*logn)