Dodatna nastava iz programiranja 2008/2009
Prirodno Matematički Fakultet, Niš
datum: 01. novembar 2008. godine
predavač: Nikola Milosavljević
e-mail: nikola5000@gmail.com

Osnovni algoritmi za rad sa nizovima - teorija

1 Sortiranje nizova

Selection Sort

Članovi niza mogu biti proizvoljno veliki brojevi (i celi i realni), Složenost: $O(n^2)$, primenljiv za $n \leq 5000$.

```
for i := 1 to n - 1 do
    for j := i + 1 to n do
        if (a[i] > a[j]) then razmeni (a[i],a[j]);
```

Bubble Sort

Članovi niza mogu biti proizvoljno veliki brojevi (i celi i realni), Složenost: $O(n^2)$, primenljiv za $n \leq 5000$.

```
for i := n - 1 downto 1 do
    for j := 1 to i do
        if (a[j] > a[j + 1]) then razmeni(a[j], a[j+1]);
```

Counting Sort

Razlikuje se od prethodnih sortova. Članovi niza moraju biti celi, ne previše veliki brojevi (ili da im vrednosti budu iz nekog konačnog opsega). Ako su npr. u pitanju celi brojevi iz segmenta $[1, k]$, onda je Složenost: $O(n + k)$. Ako je $k \leq n$ ovaj sort je primenljiv čak za $n \leq 10^6$.

```
fillchar(c, sizeof(c), 0);
for i := 1 to n do                  (* Sortiramo niz a i sortirani niz smeštamo u niz b *)
    c[a[i]] := c[a[i]] + 1;         (* U početku je c[i] = broj elemenata niza a koji su jednaki i *)
for i := 2 to k do                  (* a zatim je c[i] = broj elemenata niza a koji su ≤ i *)
    c[i] := c[i] + c[i - 1];
for i := n downto 1 do begin
    b[c[a[i]]] := a[i];
    c[a[i]] := c[a[i]] - 1;
end;
```

2 Binarna pretraga

Posmatrajmo sledeći problem: imamo sortirani realni niz a koji ima n elemenata i potrebno je proveriti da li se vrednost **val** nalazi u tom nizu. Najočigledniji algoritam je linearna pretraga, tj. krenemo od početka niza i za svaki element proverimo da li je jednak traženoj vrednosti. Ovo nam daje složenost $O(n)$ jer u najgorem slučaju (ako se **val** ne nalazi u nizu) mi napravimo n koraka.

Međutim, činjenica da je niz sortiran omogućava nam da ovo znatno ubrzamo pomoću **binarne pretrage**. Ideja je sledeća: prvo proverimo srednji element niza. Ako je on jednak **val** pretraga je završena. Ukoliko je srednji element veći od **val**, onda nastavljamo pretragu u levom podnizu (od prvog do srednjeg elementa) inače nastavljamo pretragu u desnom podnizu (jer je niz sortiran). Ovo nam omogućuje da u svakom koraku "prepolovimo" niz u kome tražimo datu vrednost što vodi logaritamskoj složenosti $O(log(n))$. Npr. za niz od milion elemenata, nama će biti potrebno svega 20 uporedjivanja.

U sledećim kodovima se u nizu $a[1..n]$ traži vrednost **val** i vraća se **ind** - indeks elementa niza a koji je jednak **val** inače **ind := 0** ukoliko takav element ne postoji.

Kod 1

```
ind := 0;
l := 1; r := n;

while (l <= r) and (ind <> 0) do begin
   m := (l + r) div 2;
   if (a[m] = val) then ind := m
   else if (val < a[m]) then r := m - 1
        else l := m + 1;
end;
```

Kod 2

```
ind := 0;
l := 1; r := n;

while (r - l > 1) do begin
   m := (l + r) div 2;
   if (val < a[m]) then r := m
   else l := m;
end;

if (a[l] = val) then ind := l;
if (a[r] = val) then ind := r;
```