Counting sort

Videli smo da merge sort radi u složenosti $O(n \log n)$, što je ujedno i očekivano vreme za quicksort. Da li možemo da sortiramo brže od $O(n \log n)$?

Odgovor je i da i ne. Naime, ovi algoritmi određuju poredak brojeva u konačnom nizu upoređivanjem brojeva. I, kao što je dokazano u lekciji Složenost algoritama, minimalan broj upoređivanja da bi se svaka permutacija brojeva sortirala je $O(n \log n)$, pa dakle sledi da ukoliko algoritam za sortiranje kao 'osnovni alat' koristi upoređivanje, složenost takvog algoritma ne može da bude manja od $O(n \log n)$, pa samim tim dobijamo da je mergesort optimalan algoritam za sortiranje.

Međutim, da li moramo da koristimo upoređivanje brojeva? Ne. Jedan od takvih algoritama je **counting sort**. Pretpostavimo da su svi brojevi u datom nizu celi brojevi čije su vrednosti između 1 i T. Ukoliko za svaki broj x, odredimo koliko postoji brojeva manjih ili jednakih od x, onda znamo na kom mestu se nalazi broj x u konačnom nizu. Najlaksi način da ovo odradimo je da napravimo pomoćni niz, recimo $br[1..T]$, i na početku stavimo sve elemente ovog niza na 0. Potom prođemo kroz početni niz i za svaki element x u ovom nizu povećamo za jedan $br[x]$. Posle ovog koraka $br[x]$ predstavlja broj elemenata u početnom nizu koji imaju vrednost x. Nakon ovoga bi nam odgovaralo kada bismo za svako $i, 1 \leq i \leq T$, izračunali:

$$P[i] = br[i] + br[i-1] + br[i-2] + \cdots + br[1]$$

Pošto bismo onda imali da se broj x u konačnom nizu nalazi na mestima $P[x-1]+1, P[x-1]+2, \ldots, P[x]$. Naivno otkucano, izračunavanje niza P zahteva $O(T^2)$ vremena, međutim, možemo da primetimo da važi

$$P[i] = br[i] + P[i-1]$$

Što nam pomaže da izračunamo niz P u $O(T)$.

Implementacija ove ideje se nalazi u *Algoritam 5*.

```
=====================================================================================
             funkcija: countingSort
             ulaz:  a      - niz brojeva
                    T      - vrednost maksimalnog elementa

             Pretpostavka je da su svi brojevi niza a u intervalu [1,T]

             Nakon izvršavanja funkcije countingSort(a) niz a će biti sortiran
-------------------------------------------------------------------------------------
      Function countingSort(a : int array, T int)
                br : int array[1..T], br[i] = 0, za i = 1..T
                P  : int array[0..T]
                n = length(a)
                For i = 1 to n do
                    br[a[i]] = br[a[i]] + 1
                P[0] = 0
                For i = 1 to T do
                    P[i] = P[i - 1] + br[i]
                For x = 1 to T do
                    For j = P[x-1] + 1 to P[x] do
                        a[j] = x
=====================================================================================
```

Algoritam 5. Pseudo kod za counting sort

Primetimo da nam niz P nije potreban. Implementacija bez niza P se ostavlja za vežbu.

Složenost ovog algoritma za sortiranje je $O(n+T)$, što ukoliko su brojevi relativno mali, npr. $T = O(n)$, dobijamo linearan algoritam za sortiranje.

Ukoliko pored brojeva imamo jos neke podatke, pored niza $br[]$ nam treba još i lista gde bismo čuvali sve te podatke.