Konstrukcija i analiza algoritama, ispit, januar 2014.

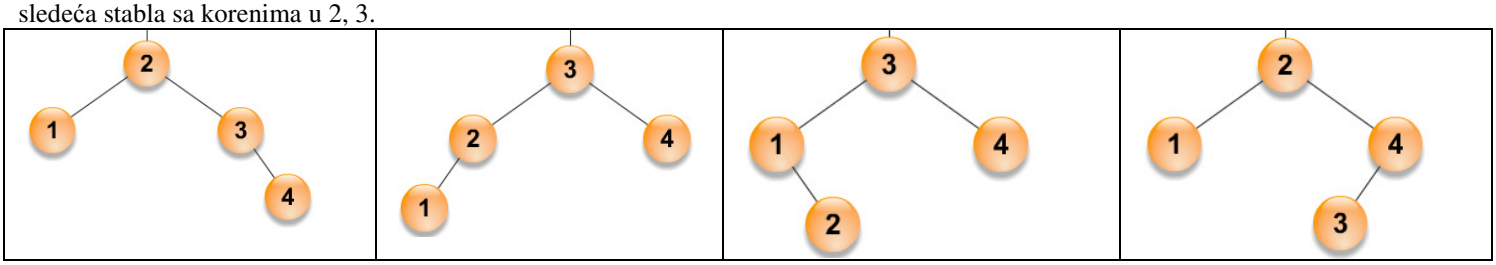
1. a) Nacrtajte sva AVL stabla koja se dobiju nakon ubacivanja elemenata iz permutacija skupa {1,2,3, 4}.
   b) Odrediti najmanji broj čvorova koje može imati AVL stablo visine h.

2. Zadat je težinski bipartitni graf sa *k* čvorova i *l* grana. Kritična težina uparivanja u grafu *G* je težina najteže grane u uparivanju. Konstruisati algoritam složenosti $O(\sqrt{k}\,(k + l)\log l\,)$ za nalaženje uparivanja sa najmanjom mogućom kritičnom težinom.

3. Konstruisati algoritam za određivanje maksimalnog od datih n brojeva (ne nužno različitih) na modelu CRCW, tako da vreme izvršavanja algoritma bude O(1) i da koristi najvise n^2 procesora .

4. Koji od algoritama (Insertion Sort, Counting Sort, Radix Sort, Bucket Sort, Merge Sort) je najefikasniji (u smislu vremenske složenosti) za upotrebu u sledećim slučajevima? Obrazlozite odgovor.
a) Sortiranje 24000000 ravnomerno raspoređenih realnih brojeva između 1 i 6000.
b) Sortiranje niza od 32000000 celih brojeva između 0 i 32000000.
c) Sortiranje niza od 4000000 brojeva.

Rešenja:
a) S obzirom da čvorovi 1, 4 ne mogu biti koren ovog AVL stabla (zbog svojstva visinske balansiranosti), onda je moguće dobiti sledeća stabla sa korenima u 2, 3.



b)

Neka je B_h oznaka za AVL stablo visine h sa najmanjim brojem čvorova sa i neka je $\mathbf{N_h}$ oznaka za broj čvorova stabla B_h .

B_h ima BAR JEDNO podstablo visine h-1 (stablo B_{h-1}), a zbog svojstva da B_h je AVL stablo sa najmanjim brojem čvorova, njegovo drugo podstablo je B_{h-2}, te važi:

$N_h=N_{h-1}+N_{h-2}+1$

$N_{h+1}=N_h+N_{h-1}+1$

$N_{h+1}- N_h = N_h +N_{h-1}+1- N_{h-1}- N_{h-2}-1$

$N_{h+1} - 2N_h + N_{h-2}=0$

Karakteristična jednačina ove rekurentne veze je

$t^3-2t^2+1=0$

odnosno $(t-1)(t^2-t-1)=0$, odnosno njeni koreni su

$$t_1=1,\ t_2=\frac{1+\sqrt{5}}{2},\ t_3=\frac{1-\sqrt{5}}{2}$$

Zato je opsti član niza N_h jednak

$$N_h = C_1 + C_2 * \left(\frac{1+\sqrt{5}}{2}\right)^h + C_3 * \left(\frac{1-\sqrt{5}}{2}\right)^h$$

Zbog $N_0=1$, $N_1=2$, $N_2=4$ dobija se sistem:

$$1=C_1 + C_2 + C_3$$

$$2= C_1 + C_2 * \frac{1+\sqrt{5}}{2} + C_3 * \frac{1-\sqrt{5}}{2}$$

$$4= C_1 + C_2 * \left(\frac{1+\sqrt{5}}{2}\right)^2 + C_3 * \left(\left(\frac{1-\sqrt{5}}{2}\right)^2\right.$$

Rešenja sistema su:

$C_1 = -1$

$C_2 = \frac{2+\sqrt{5}}{\sqrt{5}}$

$C_3 = \frac{\sqrt{5}-2}{\sqrt{5}}$

2.

Koristićemo kombinaciju binarne pretrage i algoritma za nalaženje optimalnog uparivanja.
Rešavamo najpre malo drugačiji problem: pitamo se da li za dato *t* postoji optimalno uparivanje takvo da su težine svih njegovih grana manje ili jednake od *t*
Ovaj problem možemo rešiti tako što iz grafa uklonimo sve grane čije su težine veće od *t* i zatim proveravamo da li optimalno uparivanje u novodobijenom smanjnom grafu ima jednak broj grana kao optimalno uparivanje u polaznom grafu.
Složenost ove provere jednaka je složenosti algoritma za nalaženje optimalnog uparivanja u grafu koje je $O(\sqrt{k}(l+k))$.
U grafu G ima *l* grana, te ima najviše *l* različitih težina. Binarnom pretragom tražimo najmanje *t* tako da je *t* težina neke grane i da postoji optimalno uparivanje u kome sve grane imaju težinu manju ili jednaku *t*. Zato je ukupna složenost $O(\sqrt{k}(l+k)\log l)$.

3.

Koristi se n(n-1)/2 procesora.
Svakom od datih brojeva x[i] pridruzujemo se po jedna (deljena) memorijska lokacija v[i]. Svakom paru {i,j} zadatih brojeva pridružuje se procesor P[i][j].
Najpre se u sve lokacije v[i] upisuje vrednost 1.
U sledećem koraku algoritma, procesor P[i][j] poredi elemente x[i] i x[j];
Ako je jedan od njih manji onda u njegovu odgovarajuću lokaciju upisuje se vrednost 0;
ako su x[i] i x[j] jednaki, onda ako je i<j onda se 0 upisuje u v[i], a inače se 0 upisuje u v[j].
Nakon toga samo u jednoj od lokacija v[i] ostaje upisana vrednost 1.
Otkrije se kom zadatom elementu je pridružena ta lokacija i taj broj je maksimum zadatih vrednosti.

Vreme izvršavanja algoritma je O(1) i on zahteva $O(n^2)$ procesora

4.

a) Bucket Sort, jer je ulazni niz ravnomerno raspoređe, te vremenska složenost Bucket Sort je linearna zbog korpi približno jednakih veličina.

b) Counting sort ili Radix Sort, jer je vremenska složenost oba algoritma linearna po dimenziji ulaznog niza za ograničen ulazni niz iz određenog intervala čija veličina linearna po dimenziji ulaznog niza.

c) Merge Sort zbog složenosti O(n log n), jer nemamo nikakav podatak o tipu podataka ulaznog niza niti o raspodeli niza.