Računalništvo (MAFI): 1. kolokvij

31. januar 2008

Čas reševanja: 120 minut

Reši vse štiri naloge. Pridobljenih 100 točk šteje za maksimalno oceno. Programe oddaj preko spletne učilnice, rešitve teoretičnih nalog pa na papirju.

Naloga 1 [25 točk]

V seznamu števil $\{a_1, a_2, \ldots, a_n\}$ je a_i *lokalni ekstrem*, če je bodisi strogo večji bodisi strogo manjši od svojih sosedov. Na primer, v seznamu $\{1, 2, 5, 3, 6, 8, 8\}$ so lokalni ekstremi 1, 5 in 3.

V Mathematici sestavi funkcijo `lokalniEkstremi`, ki vrne seznam lokalnih ekstremov danega seznama. Primer uporabe:

```
In[1]:= lokalniEkstremi[{1, 2, 5, 3, 6, 8, 8}]
Out[1]= {1, 5, 3}
```

Dodatna naloga [+10 točk]: Za dani $\epsilon > 0$ je število a_i *lokalni ϵ-ekstrem*, če je bodisi $a_i + \epsilon$ večji od sosednjih členov bodisi $a_i - \epsilon$ manjši od sosednjih členov v seznamu. Sestavi funkcijo `lokalniEpsilonEkstrem`, ki sprejme ϵ in seznam ter vrne seznam ϵ-lokalnih ekstremov v seznamu.

Naloga 2 [10 + 15 točk]

V Javi je dana metoda `kajDela`:

```
1  public static int kajDela(int a, int b) {
2      if (b == 0) { return 1; }
3      else {
4          int c = kajDela(a, b/2);
5          c = c * c;
6          if (b % 2 == 1) { c = a * c; }
7          return c;
8      }
9  }
```

(a) Ugotovi, kaj metoda računa, če sta a in b nenegativni celi števili.

(b) Oceni časovno zahtevnost metode. Uporabi notacijo "veliki O" in izrazi časovno zahtevnost kot funkcijo števil a in b.

Naloga 3 [10 + 15 točk]

Mediana tabele *različnih* števil $\{a_1, a_2, \ldots, a_n\}$ je tako število m, da je polovica elementov tabele manjših in polovica večjih od m. Na primer, 4 je mediana tabele $\{3, 1, 5, 4, 7\}$ in $3\frac{1}{2}$ je mediana tabele $\{4, 1, 3, 5\}$.

Pri reševanju naslednjih nalog smeš predpostaviti, da so elementi tabele dobro premešani:

(a) Opiši algoritem, ki poišče mediano v $O(n \cdot \log n)$ korakih.

(b) Opiši algoritem, ki poišče mediano hitreje kot v $O(n \cdot \log n)$ korakih in oceni njegovo časovno zahtevnost.

(c) **Dodatna naloga [+10 točk]:** algoritem iz (b) implementiraj v Javi ali Mathematici.

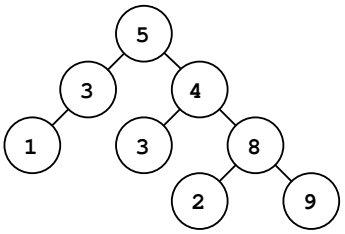
Opomba: Seveda je vsak algoritem, ki je hitrejši od $O(n \cdot \log n)$ hkrati rešitev za (a) in (b). Če torej rešiš (b), je to tudi rešitev za (a).

Naloga 4 [25 točk]

Dan je razred dvojiških dreves:

```
1  public class Drevo {
2      boolean prazen;
3      Drevo levi, desni;
4      int vsebina;
5
6      public Drevo() {
7          prazen = true; levi = null; desni = null;
8      }
9
10     public Drevo(int v, Drevo l, Drevo d) {
11         prazen = false; vsebina = v; levi = l; desni = d;
12     }
13 }
```

Razredu `Drevo` dodaj objektno metodo `public void izpisiPoti()`, ki na zaslon izpiše poti, ki vodijo od korena do listov drevesa, vsako v svojo vrsto. Na primer, če je `d` drevo



klic metode `d.izpisiPoti()` izpiše

```
5 3 1
5 4 3
5 4 8 2
5 4 8 9
```

Vrstni red izpisanih vrstic ni pomemben, pomembno je le, da se nobena pot ne ponovi večkrat in da nobena ne manjka.