



Polsemestrálny test teoretická časť



Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

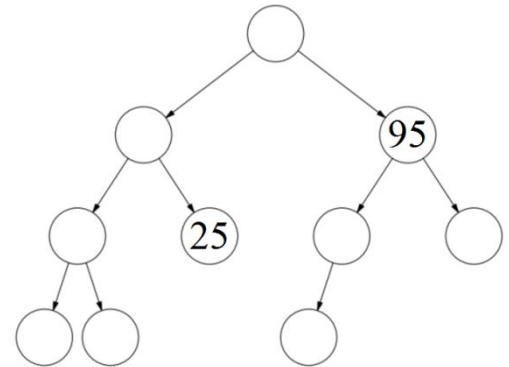
Píšte prosím čitateľne!

Hodnotenie, vyplní opravujúci:

Meno a priezvisko:	Skupina PAZ:	
--------------------	--------------	--

1/1	2/1	3/1.5	4/1.5	5/1	6/2	7/1	8/1.5	9/1.5	10/2	11/1	12/8	Σ/23

1. (1b) V binárnom vyhľadávacom strome vpravo označte všetky uzly, v ktorých sa môže nachádzať hodnota 15.
2. (1b) Zakrúžkovaním vyberte tie hodnoty, ktoré môžu byť maximálnymi hodnotami uloženými v binárnom vyhľadávacom strome vpravo?



15 30 45 80 90 91 95 96 97 127

3. (1.5b) Predpokladajme, že rad q obsahuje hodnoty:

19, 16, 31, 15, 20, 24, 13, 14, 16, 23, 12, 14, 16, 24

Aký obsah bude mať rad q po volaní spracuj(q)?

```
public void spracuj(Queue<Integer> q) {
    int i1, i2;
    q.offer(-1);
    i1 = q.poll();
    while (i1 > 0) {
        i2 = q.poll();
        if (i2 > i1)
            q.offer(i2);
        else {
            q.offer(i1);
            i1 = i2;
        }
    }
}
```

4. (1.5b) Ak sa halda skladá len z koreňa, ktorý je zároveň aj listom, povieme, že táto halda má výšku 0. Predpokladajme, že všetky hodnoty v halde sú rovnaké. Koľko rôznych hald s výškou k existuje?

5. (1b) Aký je maximálny počet uzlov binárneho stromu s výškou k? Ak sa strom skladá len z koreňa, ktorý je zároveň listom, povieme, že má výšku 0.

6. (2b) Doplňte na vyznačené miesto v metóde vhodné príkazy tak, aby výsledná metóda realizovala upravenie podhaldy, ktorej koreň je na indexe *idxKorena* a posledný uzol na indexe *idxPosledneho*, v prípade, že v koreni podhaldy (*idxKorena*) sa nachádza hodnota narušajúca vlastnosť byť haldou. Táto metóda robí presne to, čo metóda *halding* z prednášky o HeapSorte, len je napísaná iným spôsobom.

```
public static void uhalduj(int[] p, int idxKorena, int idxPosledneho) {
    int aktualny = idxKorena;

    while (true) {
        int najIdx = aktualny;
        if ((2*aktualny+1<=idxPosledneho) && (p[najIdx]<p[2*aktualny+1]))
            najIdx = 2*aktualny + 1;

        if ((2*aktualny+2<=idxPosledneho) && (p[najIdx] < p[2*aktualny+2]))
            najIdx = 2*aktualny + 2;

        if (_____ )
            break;

        int tmp = p[aktualny];
        p[aktualny] = p[najIdx];
        p[najIdx] = tmp;

        aktualny = najIdx;
    }
}
```

7. (1b) Nájdite a opravte chybu v metóde *minimum* triedy *SpajanyZoznam*, ktorá vráti minimálnu hodnotu uloženú v neprázdnom spájanom zozname.

```
public int minimum() {
    int vysledok = Integer.MAX_VALUE;
    Uzol aktualny = prvvy;
    while (aktualny != null) {
        aktualny = aktualny.dalsi;
        vysledok = Math.min(vysledok, aktualny.hodnota);
    }

    return vysledok;
}
```

8. (1.5b) Uvažujme nasledovný kód, ktorý spracúva pole *p* veľkosti *n*. O metóde *spracujZaciatok* vieme, že nejako v lineárnom čase (vzhľadom na *k*) spracúva prvých *k* prvkov poľa *p*, t.j., volanie *spracujZaciatok(p, k)* má časovú zložitosť $\Theta(k)$. Aká bude časová zložitosť celého fragmentu kódu vzhľadom na dĺžku poľa *p* (označenú ako *n*)? Odhadnite čo najtesnejšie a svoj odhad zdôvodnite.

```
int k = p.length;
boolean jeDruhy = true;
while (k > 0) {
    if (jeDruhy)
        spracujZaciatok(p, k);

    jeDruhy = !jeDruhy;
    k = k / 2;
}
```

9. (1.5b) Navrhните spôsob, ako možno v čase $O(n \cdot \log n)$ zistiť, či n -prvkové pole obsahuje nejaké 2 rovnaké hodnoty.

10. (2b) Naprogramujte metódu, ktorá zrealizuje spojenie dvoch utriedených polí p_1 a p_2 do utriedeného výsledného poľa v čase $O(n)$.

```
public static int[] zluc(int[] p1, int[] p2){
```

```
}
```

11. (1b) Formálne dokážte $12 \log_6(n) = O(\log_2 n)$.

12. (1b za správnu a -0.5b za nesprávnu odpoveď) **Rozhodnite** o pravdivosti tvrdení:

A: Triedenia výberom (MinSort, SelectionSort) spraví na každom n -prvkovom poli $O(n^2)$ krokov. Zároveň však platí, že počet vykonaných krokov je vždy $\Theta(n^2)$.

Áno Nie

B: Ak $f(n) = O(g(n))$, potom $2 * f(n) + f(n) = O(g(n))$.

Áno Nie

C: Ak má n -prvkový binárny vyhľadávací strom aspoň $n/2$ listov, potom jeho výška je $O(\log n)$.

Áno Nie

D: Najväčší prvok v binárnom vyhľadávacom strome je aj v pre-order, aj v in-order, aj v post-order zápise posledný.

Áno Nie

E: Uvažujme 2 polia dĺžky n . V čase $O(n \cdot \log n)$ môžeme zistiť, či jedno pole je permutáciou druhého.

Áno Nie

F: Ak máme usporiadané n -prvkové pole, potom v čase $O(\log n)$ môžeme vypočítať súčet hodnôt v tomto poli.

Áno Nie

Poznámka: krok = elementárna operácia

Zdôvodnite svoju odpoveď k ľubovoľnej otázke okrem otázky F (2b):