Tietorakenteet, laskuharjoitus 2, 24.-28.1

Huom: Viikon 1 tirapajatehtävien deadline on maanantaina 24.1. klo 23.59

1. Oletetaan, että tietokone suorittaa sekunnissa miljoona operaatiota. Tarkastellaan seuraavia algoritmeja, kun syötteen koko on n:

- Algoritmi A suorittaa $\log_2 n$ operaatiota.
- Algoritmi B suorittaa n operaatiota.
- Algoritmi C suorittaa n^2 operaatiota.
- Algoritmi D suorittaa n^3 operaatiota.
- Algoritmi E suorittaa 2^n operaatiota.

Esimerkiksi jos syötteen koko on 100, algoritmi C suorittaa $100^2 = 10000$ operaatiota, joten sen suoritusaika on 10 millisekuntia.

(a) Kuinka paljon aikaa algoritmit käyttävät, kun syötteen koko on miljoona?

(b) Kuinka paljon aikaa algoritmit käyttävät, kun syötteen koko on miljardi?

(c) Kuinka suuren syötteen algoritmit ehtivät käsitellä sekunnissa?

(d) Kuinka suuren syötteen algoritmit ehtivät käsitellä minuutissa?

(e) Kuinka suuren syötteen algoritmit ehtivät käsitellä vuodessa?

2. Seuraavissa tehtävissä analysoidaan muutamien yksinkertaisten algoritmien aika- ja tilavaativuus. Käytä kalvoilla 44–58 mainittuja ”aikavaativuusanalyysin nyrkkisääntöjä”. Mallia kannattaa ottaa monisteen sivujen 44–58 esimerkeistä.

Seuraava algoritmi tarkistaa, onko taulukossa kahta samaa lukua. Analysoi algoritmin aika- ja tilavaativuus, kun taulukossa on n lukua.

```
private static boolean onkoSamaa(int [] taulu) {
    for (int i = 0; i < taulu.length; i++) {
        for (int j = i+1; j < taulu.length; j++) {
           if (taulu[i] == taulu[j]) {
               return true;
           }
        }
    }
    return false;
}
```

3. Seuraava algoritmi muuttaa taulukon lukujen järjestyksen käänteiseksi. Analysoi algoritmin aika- ja tilavaativuus, kun taulukossa on n lukua.

```
private static void kaannaTaulukko(int [] taulu) {
    int pituus = taulu.length;
    int [] apu = new int[pituus];
    for (int i = 0; i < pituus; i++) {
        apu[i] = taulu[i];
    }
    for (int i = 0; i < pituus; i++) {
        taulu[i] = apu[pituus-i-1];
    }
}
```

4. Analysoi seuraavan rekursiivisen algoritmin aika- ja tilavaativuus, kun sille annetaan syötteeksi kokonaisluku n. Ensin kannattaa ehkä kokeilla algoritmin toimintaa muutamalla syötteellä.

```
private static void rek1(int n) {
    System.out.println(n);
    if (n != 0) {
        rek1(n-1);
    }
}
```

5. Analysoi seuraavan rekursiivisen algoritmin aika- ja tilavaativuus, kun sille annetaan syötteeksi kokonaisluku n. Ensin kannattaa ehkä kokeilla algoritmin toimintaa muutamalla syötteellä.

```
private static void rek2(int n) {
    System.out.println(n);
    if (n != 0) {
        rek2(n-1);
        rek2(n-1);
    }
}
```

6. Luentomateriaalin sivulla 39–43 annetaan matemaattinen määritelmä O-merkinnälle:

Väittämä $f(n) = O(g(n))$ pitää paikkansa, jos on olemassa positiiviset vakiot d ja n_0 siten, että $0 \leq f(n) \leq dg(n)$, kunhan $n > n_0$.

Esimerkkejä:

Väittämä $3n + 5 = O(n)$ pitää paikkansa, koska $3n + 5 \leq 3n + 5n = 8n$ eli sopivat vakiot ovat esimerkiksi $d = 8$ ja $n_0 = 1$.

Väittämä $n^2 = O(n)$ ei pidä paikkaansa, koska silloin olisi olemassa vakio d, jolle pätisi $n^2 \leq dn$ eli $n \leq d$, kunhan $n > n_0$. Kuitenkin jos d on vakio ja n voi kasvaa rajattoman suureksi, n ylittää varmasti jossain vaiheessa arvon d.

Mitkä seuraavista väittämistä ovat tosia ja mitkä ovat epätosia? Esitä perustelut yllä olevien esimerkkien tapaan.

(a) $2n^2 + 7n + 3 = O(n^3)$

(b) $2n^3 + 7n + 3 = O(n^2)$

(c) $\log n = O(10^6)$

(d) $10^6 = O(1)$

(e) $\log(n^2) = O(\log n)$