Tietorakenteet, laskuharjoitus 3, 1.-5.2.

1. Jatkoa viime viikon tehtävään 4. Summan 1+2+...+n voi laskea myös rekursiivisesti seuraavasti:

```
Summa(k)
    if k==0
        return 0
    else
        return summa( k-1 ) + k
```

Määritä algoritmin aikavaativuus ja tilavaativuus syötteellä n.

Voiko summan laskea niin, että aikavaativuus on vain $\mathcal{O}(1)$?

2. Kirjoita pseudokoodina monisteen sivulla 69 luonnosteltu algoritmi.

Hyvän pseudokoodin kirjoittaminen ei ole välttämättä aluksi helppoa. Esityksen on oltava riittävän selkeä mutta ei liian detaljoitu. Joitakin operaatioita voidaan tarvittaessa esittää esim. sanallisisti tyyliin *lisätään X joukkoon A*. Joskus taas sopivan matemaattisen notaation käyttö on eduksi, esim. $sum_A = \Sigma_{x \in A} x$ tarkoittaa, että kaikkien joukon A alkioiden summa lasketaan muuttujaan sum_A. Ihan mitä tahansa operaatioita ei pseudokoodiinkaan saa kirjoittaa, operaatioiden on oltava sellaisia että ne voidaan toteuttaa jotenkin järkevästi oikealla ohjelmointikielellä joko yhden tai useamman käskyn avulla. Pseudokoodia analysoitaessa monimutkaisimmista operaatiosta on muistettava, että niiden aikavaativuus ei ole välttämättä $\mathcal{O}(1)$, vaan operaatiot ovat ikäänkuin metodikutsuja, jotka piilottavat alleen syötteen pituudesta riippuvan laskennan. Tälläinen monimutkainen operaatio voisi olla esim. *Järjestä taulukon A sisältö.*

3. Ilmaise edellisen tehtävän algoritmin aikavaativuus rekursioyhtälönä.

Analysoi tarkemmin mikä on algoritmin aikavaativuus syötteen pituuden n suhteen. Rekursioyhtälön ratkaiseminen ei kurssin tietojen avulla ole mahdollista eikä edes tarpeen. Analyysi kannattaakin tehdä "laskemalla" kuinka monta kertaa algoritmin runko suoritetaan syötteen pituuden suhteen ja huomioimalla itse runko-osan aikavaativuus.

Monisteen sivun 70 kuva auttaa analyysissä. Ensimmäisen viikon tehtävän 2 tulos lienee myös hyödyllinen.

4. $\mathcal{O}$-filosofiaa

(a) Tilavaativuus tarkoittaa, kuinka paljon muistia algoritmi käyttää syötteen lisäksi. Ei ole harvinaista, että algoritmin aikavaativuus on $\mathcal{O}(n)$, mutta tilavaativuus on vain $\mathcal{O}(1)$. Tällöin algoritmi tulee toimeen kiinteällä määrällä apumuuttujia. Onko sen sijaan mahdollista, että algoritmin aikavaativuus on $\mathcal{O}(1)$, mutta tilavaativuus on $\mathcal{O}(n)$?

(b) Tarkastellaan taulukkoa, jossa on kokonaislukuja:

5	2	1	3	1

Koko taulukon sisällön voi kutistaa yhteen kokonaislukuun kertomalla peräkkäisiä alkulukuja, joiden potenssit ovat taulukon luvut: $2^5 \cdot 3^2 \cdot 5^1 \cdot 7^3 \cdot 11^1 = 5433120$. Tästä luvusta saa selville alkuperäiset luvut etsimällä luvun alkutekijät ja lukemalla niiden potenssit.

Luvuille tuntuu siis olevan tuhlaavaista varata viisi kohtaa taulukossa, kun yksikin riittäisi:

5433120	-	-	-	-

Sama menetelmä tepsii mille tahansa taulukolle, jossa on n kokonaislukua. Näyttää siltä, että alkuperäisen taulukon tilavaativuus on $\mathcal{O}(n)$, kun taas uuden taulukon tilavaativuus on vain $\mathcal{O}(1)$. Onko johtopäätös oikea?

5. Näytä miten jonoa on mahdollista simuloida kahden pinon avulla. Oleta tässä jono ja pinot rajoittamattoman kokoisiksi. Mikä on simuloidun jonon operaatioiden aikavaativuus?

6. Näytä miten jono toteutetaan linkitettynä rakenteena siten että operaatiot ovat vakioaikaisia. Toteutuksella tarkoitetaan tässä samantyyppistä pseudokoodiesitystä kuin monisteen sivuilla 95-97 on tehty pinolle tai jollakin kielellä "mahdollisimman selkeästi" toteutettua versiota. Oikeiden kielten valmiiden kirjastojen käyttö ei ole sallittua.

Muista havainnollistaa jonon operaatioiden toimintaa kuvien tms. avulla.