Tehtävä 2: Algoritmien suoritusaika

Tutustu tarkoin seuraavaan tekstiin ja vastaa sitä hyväksi käyttäen tehtävän loppuosassa esitettyihin viiteen kysymykseen.

Algoritmit ovat tietoteknisten järjestelmien ytimiä. Algoritmilla tarkoitetaan täsmällistä käskyjonoa eli toimintaohjetta, joka suorittaa halutun tehtävän. Tehtävällä on parametreina *syötteitä*, joita algoritmi käsittelee. Syötteet merkitään sulkuihin algoritmin nimen jälkeen. Algoritmi muodostaa *tuloksen*. Esimerkiksi summausalgoritmin syötteenä on kokonaislukutaulukko A ja tuloksena on taulukon alkioiden summa.

Algoritmi esitetään täsmällisesti *algoritminotaatiolla* jossa yhdistetään yksinkertaisia toimintoja *peräkkäisellä suorittamisella*, *ehdollisella suorittamisella* ja *toistolla*. Yksinkertaiset toiminnot käsittelevät, vertailevat ja laskevat yksittäisiä lukuja, totuusarvoja tai merkkejä. Tiedon väliaikaiseen tallettamiseen käytetään *muuttujia*. Kukin muuttuja sisältää kerrallaan yksinkertaisen tiedon, esimerkiksi kokonaisluvun tai totuusarvon. Muuttujan sisältö asetetaan tai muutetaan algoritmissa *aseta*-lauseella. Muuttujia voidaan myös koota *taulukoiksi*, joiden alkioita käsitellään yksi kerrallaan. Tässä tehtävässä käytetään vain yksiulotteisia taulukoita.

Esimerkiksi yksinkertainen algoritmi, joka selvittää esiintyykö jokin alkio kokonaislukutaulukossa useammin kuin kerran, voidaan esittää seuraavasti:

OnkoSamoja(taulukko A)

1 **aseta** $n \leftarrow$ taulukon A alkioiden lukumäärä
2 **aseta** *samoja* $\leftarrow$ **epätosi**
3 **aseta** $i \leftarrow 1$
4 **niin kauan kun** $i \leq n$, **suorita rivejä 5-9**
5 **aseta** $j \leftarrow 1$
6 **niin kauan kun** $j \leq n$, **suorita rivejä 7-8**
7 **jos** $A[i] = A[j]$ **ja** $i \neq j$, **niin aseta** *samoja* $\leftarrow$ **tosi**
8 **aseta** $j \leftarrow j + 1$
9 **aseta** $i \leftarrow i + 1$
10 **tulos on** *samoja*

Merkintä $A[i]$ tarkoittaa taulukon A i:nnettä alkiota (tässä kokonaisluku) alusta lukien. Tässä algoritmissa käytetään muuttujaa n ylläpitämään tietoa taulukon A alkioiden lukumäärästä, muuttujia i ja j indeksoimaan (käymään läpi) taulukkoa alkio kerrallaan ja totuusarvoista muuttujaa *samoja* ylläpitämään tietoa siitä, onko samaa alkiota jo löytynyt vai ei. Algoritmi palauttaa tuloksenaan totuusarvon rivillä 10.

Algoritmin suoritusaika

Algoritmille on ensisijaista toimia oikein, mutta myös algoritmin *tehokkuus* on tärkeää. Algoritmin tehokkuutta arvioidaan yleisimmin sen käyttämän suoritusajan suhteen – nopea algoritmi on luonnollisesti parempi kuin hidas. Algoritmin ajankäytön arviointi on erityisen tärkeää, kun algoritmin syöte on (hyvin) suuri. Koska syötteen koko usein vaihtelee tilanteen mukaan, emmekä yleensä tiedä sitä algoritmia suunnitellessamme, suhteutamme suoritusajan syötteen kokoon. Suoritusaika siis ilmaistaan *syötteen koon funktiona.*

Jos tarkastelisimme kokonaista valmista tietokoneohjelmaa, voisimme mitata suoritusaikaa esimerkiksi (nano)sekunteina tai prosessorin kellojaksoina. Tällaisissa mittareissa on kuitenkin useita huonoja puolia. Voisimme mitata vain jo toteutettua algoritmia ja suoritusaikaan vaikuttaisivat huomattavasti sekä käytetty laitteisto, esimerkiksi prosessorin ja muistin rakenne ja nopeus, että mittavälineet. Näin ollen tarvitsemme hieman abstraktimman ja yksinkertaisemman suoritusajan yksikön ja mittaustavan. Käytetään yksikköä *askel.* Ajatellaan aluksi, että kukin algoritmin yksinkertainen operaatio (esimerkiksi yksinkertainen sijoitus, laskutoimitus tai vertailu) vaatii yhden askeleen. Kun usea operaatio suoritetaan peräkkäin, kokonaisuuden suoritusaika on niiden suoritusaikojen summa. Siten esimerkiksi algoritmin

1 **aseta** $a \leftarrow 1$
2 **aseta** $b \leftarrow 1$

suoritusaika on kaksi askelta. Jos algoritmi suorittaa toistuvasti samoja rivejä, kerrotaan toistettavan osan suoritusaika toistojen määrällä. Esimerkiksi algoritmin

1 **aseta** $i \leftarrow 1$
2 **niin kauan kun** $i \leq 1000$, **suorita riviä 3**
3 **aseta** $i \leftarrow i + 1$

suoritusaika on vähintään 1000 askelta. Yksinkertaisten operaatioiden määrä on jopa suurempi kuin 1000. Voitaisiin laskea algoritmissa olevan vaikkapa 1000 yhteenlaskuoperaatiota, 1001 aseta-operaatiota, 1001 vertailuoperaatiota ja ehkä vielä 1000 "näkymätöntä" operaatiota toiston toteuttamiseksi. Toisaalta voimme ajatella rivin 1 suoritettavaksi yhden kerran, rivin 3 1000 kertaa ja rivillä 2 tehtävän vertailu 1001 kertaa. Askeleiden kokonaismäärä on joka tapauksessa suunnilleen jokin luvun 1000 monikerta. Palaamme tähän yksinkertaistamiseen alempana.

Algoritmin pitäisi toimia millä syötteen koolla tahansa. Näin ollen tällaista 1000 kierroksen toistoa ei algoritmeissa yleensä esiinny, vaan toistojen määrä

riippuu syötteestä. Esimerkiksi seuraava algoritmi SUMMA saa parametrinaan kokonaislukutaulukon ja laskee taulukon alkioiden summan.

SUMMA (taulukko A)
1 **aseta** $n \leftarrow$ taulukon A alkioiden lukumäärä
2 **aseta** $i \leftarrow 1$
3 **aseta** $sum \leftarrow 0$
4 **niin kauan kun** $i \leq n$, **suorita rivejä 5-6**
5 **aseta** $sum \leftarrow sum + A[i]$
6 **aseta** $i \leftarrow i + 1$
7 **tulos on** sum

Merkitään taulukon A alkioiden lukumäärää n:llä. Sovitaan, että taulukon alkioiden lukumäärän selvittäminen (rivi 1) on yksi askel. Algoritmin SUMMA suoritusaika on vähintään n askelta. Toki voisimme olla tarkempia ja todeta askelia olevan esimerkiksi $3n + 4$ kappaletta (toistettavan osan käskyjen (vertailu ja 2 asetusta) määrän ja toistojen määrän n tulon ja rivien 1–3 ja 7 asetusten määrän summa).

Kuten huomaamme, suoritusajasta tulee syötteen koon funktio, jota merkitään esimerkiksi $f(n)$. Aina tulee kuitenkin selventää mitä funktion argumentti n tarkoittaa.

Algoritmin suoritusaika voi riippua paitsi syötteen koosta, myös sen laadusta. Yleensä tarkastellaan aina pahinta (hitainta) tapausta, sillä haluamme tietää, kauanko suoritus enintään kestää. Jos ollaan kiinnostuneita vain algoritmin suoritusajan karkeasta ylärajasta, pahinta tapausta ei tarvitse muodostaa, vaan voidaan tyytyä selvittämään algoritmin osien suoritusaikojen ylärajoja ja laskea niiden perusteella yläraja koko algoritmin suoritusajalle.

Kertaluokkatarkastelu

Pienellä syötteellä useimmat algoritmit ovat riittävän nopeita. Niinpä olemme yleensä kiinnostuneita suoritusajasta, kun syötteen koko on suuri. Silloin esimerkiksi algoritmin SUMMA alkuosan kolmen askeleen suoritusaika on vähäinen kokonaisuuden kannalta, ja jätämme sen huomiotta. Ei kannata myöskään yrittää laskea tarkasti montako operaatiota algoritmin toisto-osa sisältää, sillä yhden toistokerran suoritusaika ei riipu taulukon alkioiden lukumäärästä. Näin ollen yksinkertaistamme muotoa $c \cdot f(n)$ olevat suoritusaikafunktiot, jossa kerroin c on syötteestä riippumaton vakio, muotoon $f(n)$. Jos algoritmi tai sen osa koostuu kahdesta peräkkäisestä osasta, niiden yhteenlaskettu suoritusaika on korkeintaan kaksi kertaa niin suuri kuin hitaamman osa suoritusaika. Koska

vakiokerroin voidaan jättää pois, riittää ottaa huomioon kahdesta peräkkäisestä osasta hitaampi. Esimerkiksi n^2+3n on kertaluokkaa n^2 kun muistetaan, että tarkastellaan suoritusaikaa syötteen koon (n) kasvaessa suureksi. Kun n on tuhansia, miljoonia tai enemmän, on polynomin aste huomattavasti merkittävämpi kuin termien vakiokertoimet. Syötteen koon kymmenkertaistuessa kertaluokkaa n olevien funktioiden arvo samoin kymmenkertaistuu, mutta kertaluokkaa n^2 olevien funktioiden arvo satakertaistuu.

Kun algoritmin suoritusaikafunktiota yksinkertaistetaan tällä tavalla, saadaan lopulta yleensä muodoltaan varsin yksinkertainen funktio. Näin yksinkertaistettua funktiota kutsutaan alkuperäisen funktion *kertaluokaksi*. Polynomifunktion kertaluokka on sen korkeimman asteen termi ilman vakiokerrointa. Algoritmin suoritusajan kertaluokka voi olla esimerkiksi n, n^2, n^3, $\log n$, $\sqrt{n}$, $n \cdot \sqrt{n}$ tai 2^n.

Kun aiemmin pääteltiin algoritmin SUMMA suoritusajan olevan $3n + 4$, niin kertaluokkatarkastelussa siis vakio 4 ja vakiokerroin 3 voidaan jättää huomiotta. Näin ollen algoritmin SUMMA suoritusajan kertaluokka on n. Samalla nähdään, ettei yksittäisten operaatioiden tarkan lukumäärän selvittäminen ole oleellista, sillä myös esimerkiksi funktio $8n + 7$ yksinkertaistuu kertaluokaksi n. Vastaavasti esimerkiksi funktiot $3n^2 + 4n$ ja $n^2 - n + 12$ yksinkertaistuvat kertaluokaksi n^2.

Algoritmin ONKOSAMOJA suoritusaikaa analysoitaessa aloitamme sisimmästä toistosta: Rivien 7-8 operaatioiden suoritusaika kullakin kerralla on syötteen koosta riippumaton (vakioaikainen). Rivin 6 toisto suorittaa rivejä 7–8 n kertaa (n on taulukon A alkioiden lukumäärä). Rivien 6–8 suoritusaika on siis yhteensä kertaluokkaa n. Lisäksi riveillä 5 ja 9 on vakioaikaisia operaatioita, mutta yhteensä rivit 5–9 ovat suoritusajaltaan kertaluokkaa n. Kun niitä suoritetaan rivin 4 toistossa n kertaa, niin koko algoritmin suoritusaika on muotoa $vakio \cdot n \cdot n$. Funktion kertaluokka on n^2.

Kysymykset

Kussakin seuraavista kysymyksistä pyydetään algoritmin *suoritusaikaa taulukon alkioiden lukumäärän funktiona.* Voit vastata joko laskemalla tarkan suoritusaikafunktion askeleina (suoritettujen ohjelmarivien määrä, esimerkiksi $2n^2 + n + 3$) tai käyttämällä yksinkertaistavaa kertaluokkatarkastelua ja vastata pelkän kertaluokan (esimerkiksi n^2). Molemmilla vastaustavoilla voi saada täydet pisteet. Ellei muuta mainita, kysytään algoritmin suoritusajan pahinta tapausta.

Kysymys 1. Mikä on seuraavan algoritmin MAKSIMI suoritusaika tai sen kertaluokka? (maksimipistemäärä 4)

MAKSIMI(taulukko A)

1 **aseta** $n \leftarrow$ taulukon A alkioiden lukumäärä
2 **jos** $n = 0$, **tulos on** määrittelemätön, **lopeta**
3 **aseta** $max \leftarrow A[1]$
4 **aseta** $i \leftarrow 2$
5 **niin kauan kun** $i \leq n$, **suorita rivejä 6-7**
6 **jos** $max < A[i]$, **niin aseta** $max \leftarrow A[i]$
7 **aseta** $i \leftarrow i + 1$
8 **tulos on** max

Kysymys 2. Mikä on seuraavan algoritmin JÄRJESTÄ suoritusaika tai sen kertaluokka? (maksimipistemäärä 5)

JÄRJESTÄ(taulukko A)

1 **aseta** $n \leftarrow$ taulukon A alkioiden lukumäärä
2 **aseta** $j \leftarrow 2$
3 **niin kauan kun** $j \leq n$, **suorita rivejä 4-10**
4 **aseta** $apu \leftarrow A[j]$
5 **aseta** $i \leftarrow j - 1$
6 **niin kauan kun** $i > 0$ **ja** $A[i] > apu$, **suorita rivejä 7-8**
7 **aseta** $A[i + 1] \leftarrow A[i]$
8 **aseta** $i \leftarrow i - 1$
9 **aseta** $A[i + 1] \leftarrow apu$
10 **aseta** $j \leftarrow j + 1$
11 **tulos on** taulukko A

Kysymys 3. Kuvaile sellaisen n alkiota sisältävän taulukon sisältö, että kysymyksen 2 algoritmi JÄRJESTÄ toimii mahdollisimman nopeasti. Anna algoritmin vaatima aika kyseisessä tapauksessa luvun n funktiona tai funktion kertaluokka. Tässä on siis kyseessä *parhaan tapauksen vaativuus* eikä pahimman tapauksen vaativuus. (maksimipistemäärä 4)

Kysymys 4. Mikä on seuraavan algoritmin JAOTTELE suoritusaika tai sen kertaluokka? (maksimipistemäärä 6)

JAOTTELE(taulukko A, kokonaisluku x)

```
 1  aseta n ← taulukon A alkioiden lukumäärä
 2  aseta v ← 1
 3  aseta o ← n
 4  aseta apu ← A[v]
 5  niin kauan kun v < o, suorita rivejä 6-11
 6          niin kauan kun v < o ja x < A[o], suorita riviä 7
 7                  aseta o ← o − 1
 8          aseta A[v] ← A[o]
 9          niin kauan kun v < o ja x ≥ A[v], suorita riviä 10
10                  aseta v ← v + 1
11          aseta A[o] ← A[v]
12  aseta A[v] ← apu
13  tulos on taulukko A
```

Kysymys 5. Mikä on seuraavan algoritmin HAE suoritusaika tai sen kertaluokka, kun taulukko A on *valmiiksi järjestetty kasvavaan järjestykseen*? Rivin 5 merkintä $\lfloor\ \rfloor$ tarkoittaa pyöristystä alaspäin lähimpään kokonaislukuun. (maksimipistemäärä 6)

HAE(taulukko A, kokonaisluku x)

```
1  aseta n ← taulukon A alkioiden lukumäärä
2  aseta v ← 1
3  aseta o ← n
4  niin kauan kun v ≤ o, suorita rivejä 5-8
5          aseta k ← ⌊(v + o)/2⌋
6          jos x < A[k], niin aseta o ← k − 1
7          jos x > A[k], niin aseta v ← k + 1
8          jos x = A[k], niin tulos on "löytyy indeksistä k", lopeta
9  tulos on "ei löydy"
```