

1 Hajautus (2p + 2p + 2p)

Hajauta annetut alkiot taulukkoon (koko $m = 19$) käyttäen määriteltyä hajautusmenetelmää.

	menetelmä	hajautusfunktio
a)	lineaarinen kokeilu	$h(k) = (k + 5) \bmod m$
b)	neliöllinen kokeilu	$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m; h'(k) = k + 2; c_1 = 0; c_2 = 1$
c)	kaksoishajautus	$h_1(k) = (k + 3) \bmod m; h_2(k) = 7 - (k \bmod 7)$

Hajautettavat alkiot: 678, 214, 608, 793, 215, 602, 176, 627 ja 545.

Pohdittavaa.

- Millaisella syötejonolla lineaarinen kokeilu toimii huonosti?
- Liian täysi hajautustaulu johtaa sen suorituskyvyn romahtamiseen. Rakenteen tehokkuus edellyttää siis ylimääräistä tilaa. Riittääkö $O(\log N)$ lisätila tähän neliöllisessä kokeilussa?
- Miten hakupuun toimintaa voidaan tehostaa yhdistämällä siihen jokin hajautusmenetelmä?

2 Leveys- ja syvyysuuntainen haku (2p + 2p)

Seuraavassa on annettu erään verkon vieruslistaesitys.

A: B F
B: A C D E H
C: B E G
D: B G
E: C B
F: A H
G: C I D
H: F B
I: G

Piirrä verkko. Missä järjestyksessä verkon solmuissa vierailaan, kun sovelletaan alla olevaa a) BFS-algoritmia b) DFS-algoritmia?

```
Algorithm BFS(G, s)
1. Initialize empty queue Q
2. for each u in V[G] do
3.     visited[u] = false
4.     finished[u] = false
5. visited[s] = true
6. ENQUEUE(Q, s)
7. while Q not empty do
8.     u = DEQUEUE(Q)
9.     for each v in Adj[u] do
10.        if (visited[v] = false)
11.            visited[v] = true
12.            ENQUEUE(Q, v)
13.        finished[u] = true
```

```
Algorithm DFS(G)
1. for each u in V[G] do
2.     visited[u] = false
3.     finished[u] = false
4. for each u in V[G] do
5.     if (visited[u] = false)
6.         DFS-VISIT(G, u)
```

```
Algorithm DFS-VISIT(G, u)
1. visited[u] = true
2. for each v in Adj[u] do
3.     if (visited[v] = false)
4.         DFS-VISIT(G, v)
5. finished[u] = true
```

Pohdittavaa.

i) Suunnittele BFS-algoritmin toteutus Javalla. Mitä kielen omia kirjastoja voisit hyödyntää apurakenteiden toteuttamiseen? Millaisia luokkia toteutuksessasi esiintyy? Mitä metodeja näissä luokissa tarvitaan?

ii) Taulukkolaskimessa solun arvon evaluoiminen edellyttää toisinaan muiden solujen arvojen evaluoimista ensin. Miten voisit hyödyntää DFS-algoritmia sen järjestyksen määrittämiseksi, jossa solujen arvot tulee evaluoida?

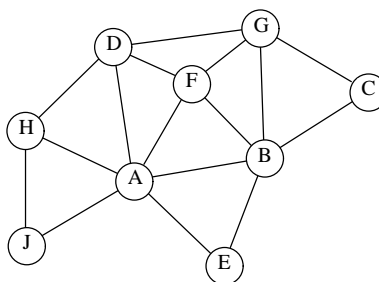
3 Dijkstran algoritmi (4p)

Seuraavassa on esitetty Dijkstran algoritmi pseudokoodina

```
SSSP-Dijkstra(G,root) // G = (V,E,W)
1  for each u in V
2      do u.priority = MAX_VALUE;
3      u.unvisited = TRUE
4      u.father = NULL
5  root.priority = 0 // root in V
6  Q.Insert(root); // Priority Queue Q
7  while Q not empty
8      do u = DeleteMin(Q)
9      u.unvisited = FALSE
10     add edge (u.father, u) into the spanning tree
11     for each (u,v) in E
12         do if (v.unvisited and u.priority + W(u,v) < v.priority)
13             then v.father = u
14                 v.priority = u.priority + W(u,v)
15             Q.InsertOrUpdate(v)
```

Tehtävänä on ratkaista lyhimmin poluin virittävän puun ongelma (*single-source shortest-paths problem*) annetulla algoritmilla. Algoritmin saa syötteenään oheisen painotetun verkon vieruslistana ja lähtösolmuksi root = A. Kaarien painot on annettu kohdesolmujen perässä olevina kokonaislukuina.

A: B3 D5 E6 F10 H14 J15
B: A3 C4 E15 F11 G13
C: B4 G12
D: A5 F7 G2 H9
E: A6 B15
F: A10 B11 D7 G8
G: B13 C12 D2 F8
H: A14 D9 J1
J: A15 H1



Pohdittavaa.

i) Mitä rajoitteita Dijkstran algoritmin saamille parametreille tulee antaa, jotta algoritmi toimisi oikein (Vihje: tarkastele algoritmin toimintaa myös negatiivisilla kaarilla)?

ii) Analysoi tehtävänannossa esitetty algoritmi. Mikä on sen aikavaatimus iso O -notaatioissa? Kuinka paljon algoritmi vaatii lisätilaa iso O -notaatioissa?

4 Primin algoritmi (4 p)

Seuraavassa on esitetty Primin algoritmi pseudokoodina

```

MST-Prim(G,root) // G = (V,E,W)
1  for each u in V
2      do u.priority = MAX_VALUE
3          u.unvisited = TRUE
4          u.father = NULL
5  root.priority = 0 // root in V
6  Q.Insert(root); // Priority Queue Q
7  while Q not empty
8      do u = DeleteMin(Q)
9          u.unvisited = FALSE
10         add edge (u.father, u) into the spanning tree
11         for each (u,v) in E
12             do if (v.unvisited and W(u,v) < v.priority)
13                 then v.father = u
14                     v.priority = W(u,v)
15                     Q.InsertOrUpdate(v)

```

Tehtävänä on muodostaa verkon minimaalinen virityspuu (*minimum spanning tree*) annetulla algoritmilla. Algoritmin saa syötteenään edellisen tehtävän painotetun verkon vieruslistana ja lähtösolmuksi $root = A$.

Pohdittavaa.

Suunnittele Primin algoritmin toteutus Javalla. Mitä kielen omia kirjastoja voisit hyödyntää apurakenteiden toteuttamiseen? Millaisia luokkia toteutuksessasi esiintyy? Mitä metodeja näissä luokissa tarvitaan?

5 Demotehtävä: Hajautus ja verkkoalgoritmit

a) Mitä hyviä ja huonoja puolia tasapainotetuissa hakupuissa on verrattuna hajautukseen?

b) Tarkastellaan yhtenäistä suuntaamatonta painotettua verkkoa. Verkon painot ovat kokonaislukuja. Esitä algoritmi, joka laskee verkon kaikkien särmien painojen summan. Pidä huolta, että kukin särmä lasketaan summaan vain kerran. Esitä algoritmin yleisperiaate sanallisesti. Selitä myös, mitä tietorakenteita tarvitset algoritmin apuna. Esitä algoritmin toiminta myös pseudokoodina tai jollain ohjelmointikielellä, (C, C++, Java tai Pascal). Mikä on algoritmisi tehokkuus, jos verkossa on V solmua ja E särmää? Perustele tuloksesi.