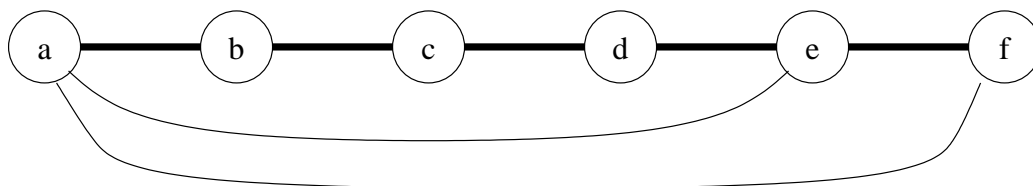


Kevät 2007 2. välikoe, tehtävä 4

(a) Ehdotettu ratkaisumenetelmä on virheellinen.

Tarkastellaan esim. allaolevaa verkkoa, jossa on suoritettu syvyysuuntainen läpikäynti solmusta a alkaen ja valitut puukaaret on paksumattu:



Nyt verkon lyhin sykli on (a, e, f, a) , mutta ehdotettu algoritmi ei löydä sitä.

(b) Ehdotettu ratkaisumenetelmä toimii oikein.

Olkoon (V, T) jokin pienin virittävä puu ja p maksimipainoltaan pienin polku s :stä t :hen puun kaaria pitkin. Tehdään vasta oletus, että on olemassa maksimipainoltaan pienempi polku p' solmusta s solmuun t . Valitaan polun p painavin kaari; olkoon se e . Tarkastellaan verkkoa $(V, T - \{e\})$. Verkossa on kaksi komponenttia, joista toinen sisältää solmun s ja toinen solmun t . Polku p' sisältää ainakin yhden kaaren e' , joka yhdistää näitä komponentteja. Siis $(V, T \cup \{e'\} - \{e\})$ on virittävä puu. Oletuksen mukaan kaikki polun p' kaaret ovat kevyempiä kuin e . Erityisesti tämä koskee kaarta e' , joten

$$w(T \cup \{e'\} - \{e\}) = w(T) + w(e') - w(e) < w(T).$$

Siis verkolla on pienempikin virittävä puu kuin (V, T) ; ristiriita.

Pisteytys: Maksimipistemäärä kohdasta (a) oli 3 pistettä. Yhden pisteen sai, jos ehdotettu menetelmä oli todettu toimimattomaksi, mutta perustelut olivat pahasti puutteelliset tai virheelliset, tai jos menetelmää luultiin toimivaksi, mutta perusteluosuudessa oli sinänsä oikeita havaintoja syklien etsimisestä.

Maksimipistemäärä kohdasta (b) oli 5 pistettä. Yhden pisteen sai, jos ehdotettu menetelmä oli todettu toimivaksi tai jos menetelmää luultiin toimimattomaksi, mutta perusteluissa oli jotain oikeita havaintoja pienimmästä virittävästä puusta. Kolme pistettä sai toteamalla menetelmän toimivaksi ja esittämällä väitteen tueksi hyviä huomioita virittävästä puusta. Täydet viisi pistettä edellyttivät, että perustelut oli muotoiltu kunnolliseksi loogiseksi argumentiksi.

Kevät 2006, 2. välikoe, tehtävä 4

(a) Yksi mahdollisuus on muuttaa kurssilla esitettyä Dijkstran algoritmia välttämään tietyöt, ja tämä toteutetaan hyväksymällä löydetty lyhyempi reitti vain jos sillä ei ole tietyötä (*). Lisäksi verkon käsittely voidaan lopettaa, kun määränpää b siirrettäisiin käsiteltyjen joukkoon. Lopuksi tulostetaan määränpäähän b osoittavat p -linkit käänteisessä järjestyksessä.

Dijkstran algoritmin toiminta on seuraava: alustetaan rakenne asettamalla kaikkien solmujen parhaaksi tunnetuksi etäisyydeksi lähtösolmusta ääretön, paitsi lähtösolmulle itselleen nolla, sekä kaikille solmuille parhaan tunnetun reitin kertovaksi linkiksi NIL, sekä muodostetaan solmuista minimikeko, jotta vielä käsittelemättömistä voidaan varsinaisessa työvaiheessa käsitellä aina se, johon tunnetaan lyhin reitti, ja joka ei siis voi enää muiden käsittelemättömien kautta parantua (kun oletetaan, että negatiivisia kaaripainoja ei esiinny). Keosta luetaan silmukassa solmuja, ja mikäli käsiteltävän solmun kautta löytyy johonkin solmuun v vanhastaan tunnettua parempi reitti (paras tunnettu etäisyys käsiteltävään solmuun $+ kaaren (u,v) pituus$ on pienempi kuin paras aikaisemmin tunnettu etäisyys solmuun v), päivitetään solmun v tietoja, sekä huolehditaan se oikealle paikalle keossa. Keon tyhjentynyt kaikkien saavutettaviin solmuihin tunnetaan lyhin mahdollinen reitti.

Arvostelu: Dijkstraa huonompi aikavaativuus, toiminnalliset virheet, kuten esim. tietöiden jättäminen huomiotta, sekä epäselvät ja liian moniselitteiset selitykset vähensivät pisteitä.

Kaikki täysien pisteiden arvoiset ratkaisut tekivät oleellisesti samaa kuin mainittu muunnettu Dijkstran algoritmi. Esimerkiksi keon sijasta jonoa käyttäneet ratkaisut, joissa solmu viedään lyhyemmän reitin löydyttyä uudelleen jonon hännille, ovat selvästi hitaampia.

(b) Koodi:

```
TietyötönDijkstra(V,Adj,w,t,a,b)
  forall v in V do
    d[v] <- +INF
    p[v] <- NIL
  d[s] <- 0
  S <- {}
  H <- new MinHeap
  forall v in V do
    HeapInsert(H,v,d[v])
  while not Empty(H) do
    u <- HeapDelMin(H)
    if u = b then
      TulostaReitti(b)
      return TRUE
    S <- S U {u}
    forall v in Adj[u] do
      if d[v] > d[u] + w(u,v) and not t(u,v) then // (*)
        d[v] <- d[u] + w(u,v)
        p[v] <- u
        HeapDecreaseKey(H,v,d[v])
  return FALSE

TulostaReitti(u)
  T <- new Stack
  while u <> NIL do
    Push(T,u)
    u <- p[u]
  while not Empty(T) do
    print(pop(T))
```

Arvostelu: jo (a)-kohdassakin mainituista ynnä kaikista muistakin pienistä toiminnallisista virheistä vähennettiin -1 p / virhe