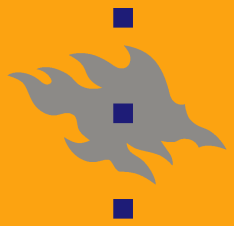


HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

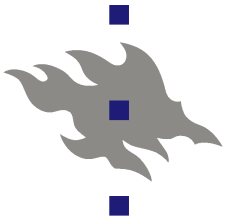
Lyhimmän polun etsiminen muistihierarkiatehokkaasti

Teppo Niinimäki

‹teppo.niinimaki@helsinki.fi›

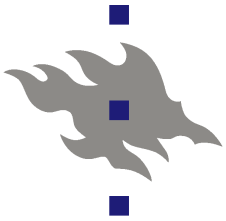
6. huhtikuuta 2009

Tietojenkäsittelytieteen laitos



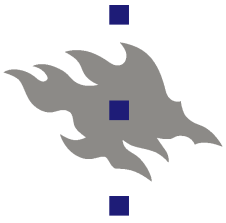
Sisältö

1. Johdanto
2. Prioriteettijonoja
3. Ämpärikeko
4. Lyhimmän polun etsintä ämpärikeolla
   - esimerkki
5. Testituloksia
6. Lopuksi



Johdanto

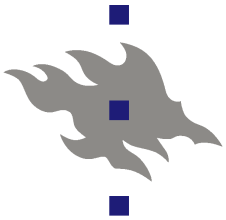
- Lyhimmän polun etsintä on keskeinen verkko-ongelma
- SSSR (Single-Source Shortest Path)
  - etsi lyhimmät polut lähtösolmusta kaikkiin muihin solmuihin
- Dijkstran algoritmi
  - käy verkon läpi leveyssuunnassa
  - jäljellä olevat solmut prioriteettijonossa
  - prioriteettijonon operaatiot: DELETE-MIN, DECREASE-KEY (ja INSERT)



Johdanto (2)

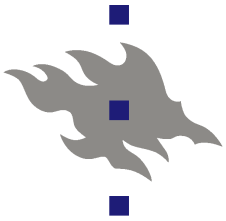
Ongelmat muistihierarkiassa

- Vieruslistojen läpikäynti
  - väh. yksi lohkonsiirto listaa kohden
  - ongelma harvassa verkossa
  - ei yleispätevää ratkaisua
- DECREASE-KEY-operaatio
  - puuttuu useista muistihierarkiatehokkaista prioriteettijonoista
  - useita ratkaisuja
- Oletuksia
  - verkko suuntaamaton
  - kaarten painot ei-negatiivisia



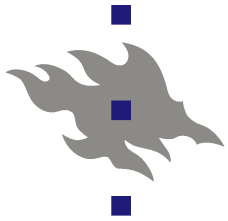
Prioriteettijonoja

- Muistitasotietoisia (cache-aware)
  - muistitasotehokas turnauspuu (I/O-efficient tournament tree)
  - sekvenssikeko (sequence heap)
- Muistitasoriippumattomia (cache-oblivious)
  - ämpärikeko (bucket heap), puskurikeko (buffer heap)
  - suppilokeko (funnel heap)
  - Arge-keko (Arge heap)
- Yleisesti DECREASE-KEY puuttuu, korvattu
  - joko INSERT-operaatiolla
  - tai UPDATE-operaatiolla



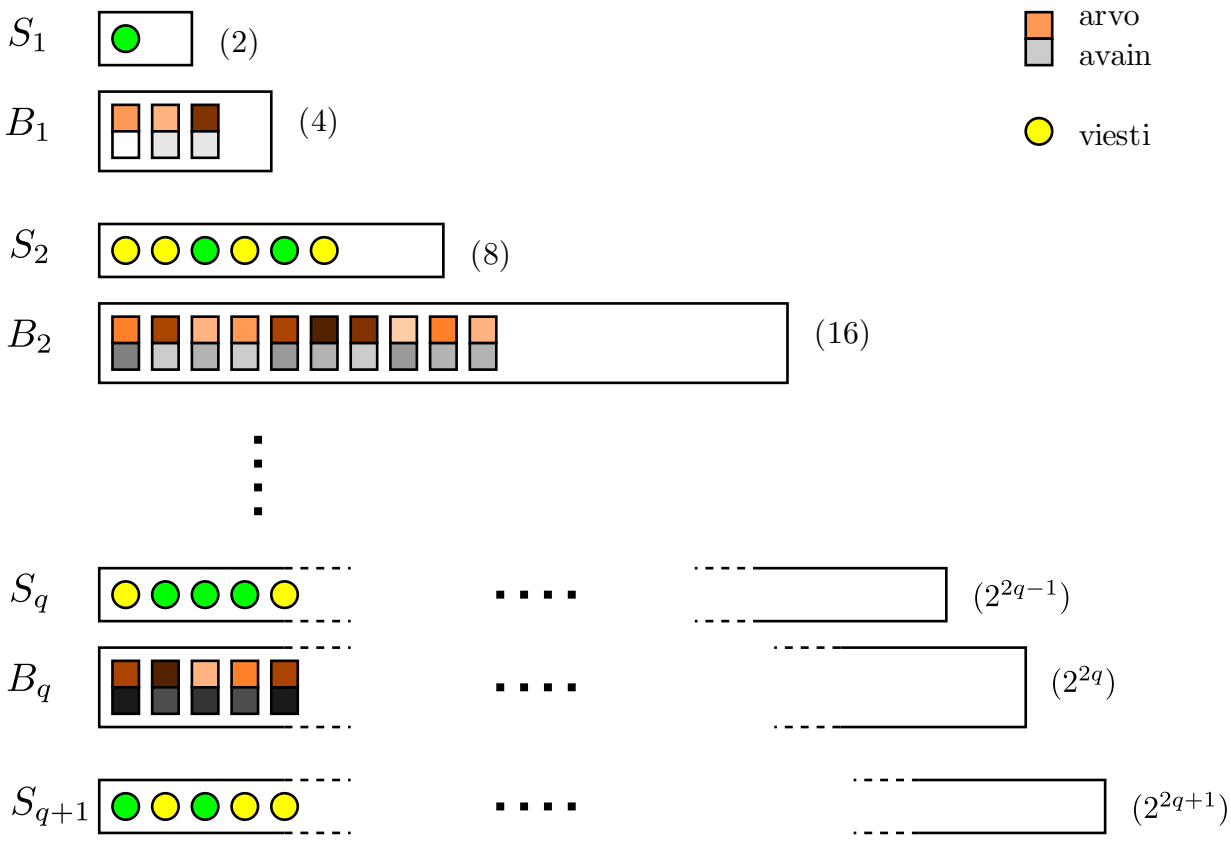
Ämpärikeko

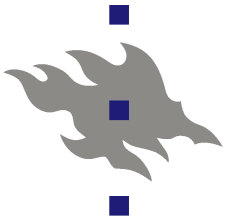
- Muistitasoriippumaton kekototeutus
- Rakenne
  - alkiot ämpäreissä $B_1, B_2, \ldots, B_q$
  - viestipuskurit $S_1, S_2, \ldots, S_{q+1}$
  - ämpäreiden ja puskureiden koot kasvavat eksponentiaalisesti
- Operaatiot
  - DELETE-MIN
  - DELETE(x)
  - UPDATE(x, k') (yhdistetty INSERT ja DECREASE-KEY)



Ämpärikeko (2)

Rakenne



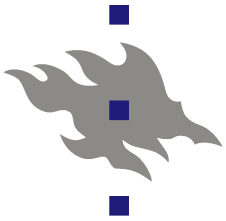


Ämpärikeko (3)

Operaatiot

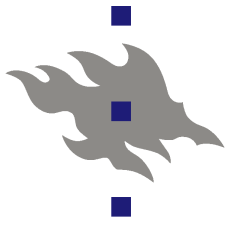
- Operaatiot suoritetaan lähettämällä *viestejä*
  - alkion poistoviesti, alkion avaimen päivitysviesti, alkion työntöviesti tasoa alemmas
  - etenevät *laiskasti*

- Apuoperaatiot
  - $\textsc{Empty}(S_i)$ ja $\textsc{Fill}(B_i)$
  - suorittavat tietorakenteen päivitykset



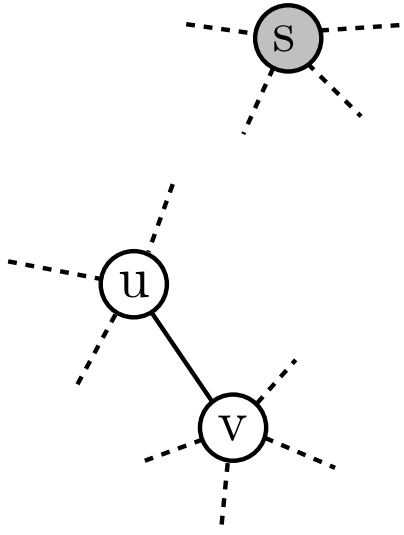
Lyhimmän polun etsintä

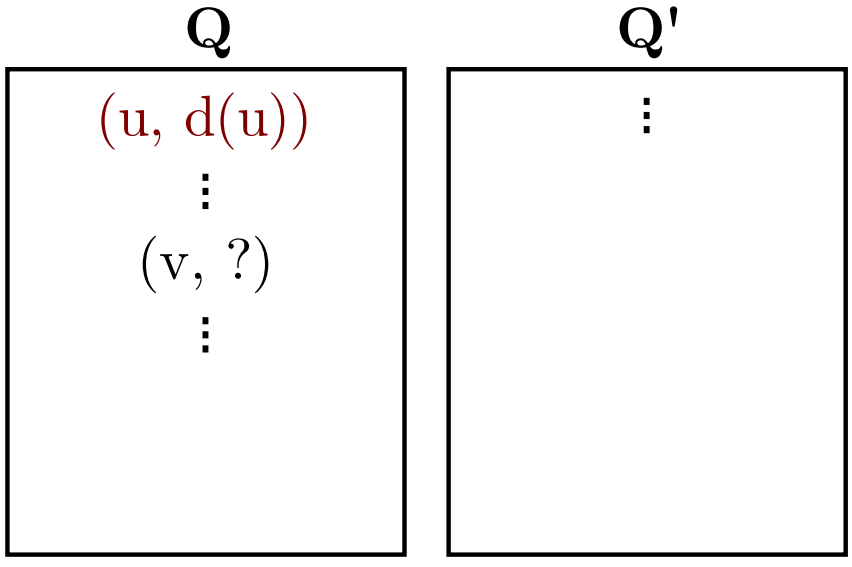
- Dijkstran algoritmin muunnelma
  - prioriteettijonona Q käytetään ämpärikekoa
  - INSERT- ja DECREASE-KEY-kutsut korvataan UPDATE-kutsuilla
  - ongelma: UPDATE voi lisätä jo poistetun alkion takaisin
- Lisäksi käytössä apuprioriteettijono Q'
  - lisätään alkio aina UPDATE-kutsun yhteydessä
  - pitää kirjaa mahdollisista jo käsiteltyjen solmujen turhista lisäyksistä takaisin jonoon Q

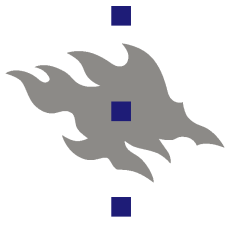


Lyhimmän polun etsintä (2)

Esimerkki

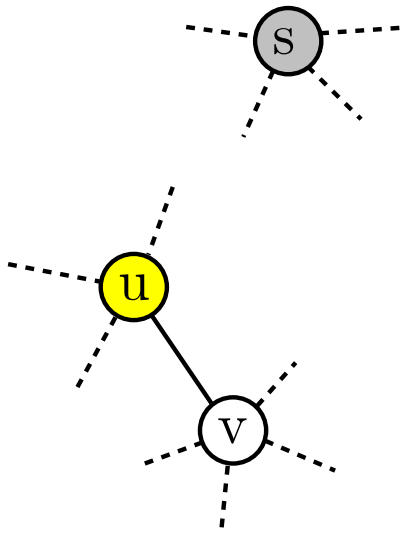


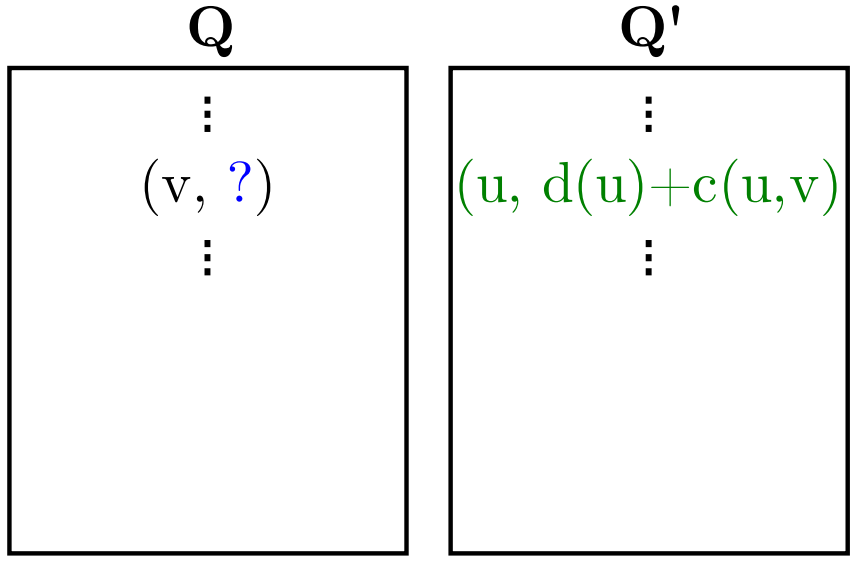


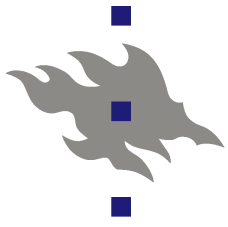


Lyhimmän polun etsintä (2)

Esimerkki (2)

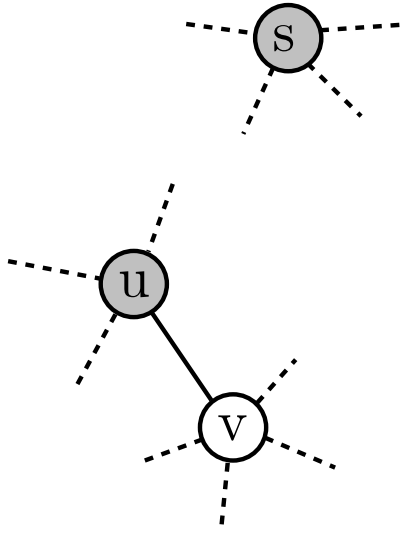


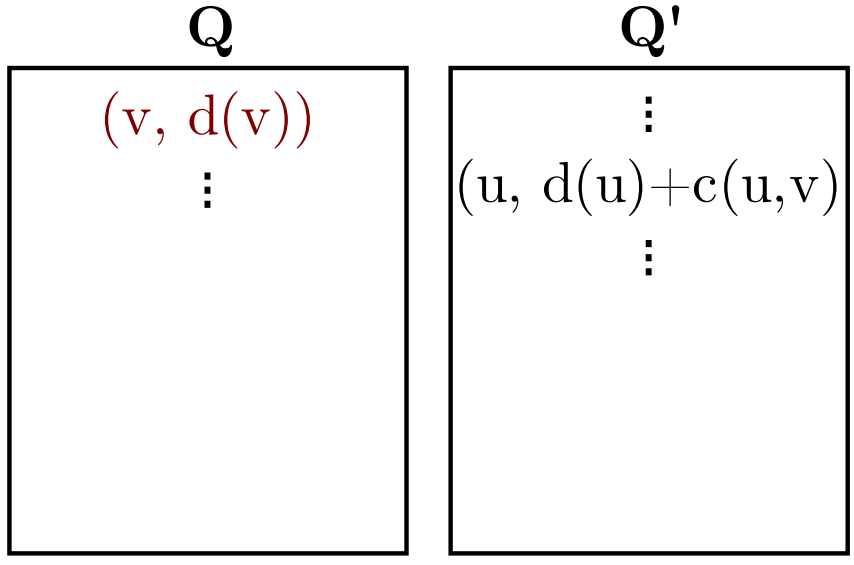


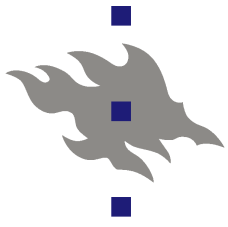


Lyhimmän polun etsintä (2)

Esimerkki (3)

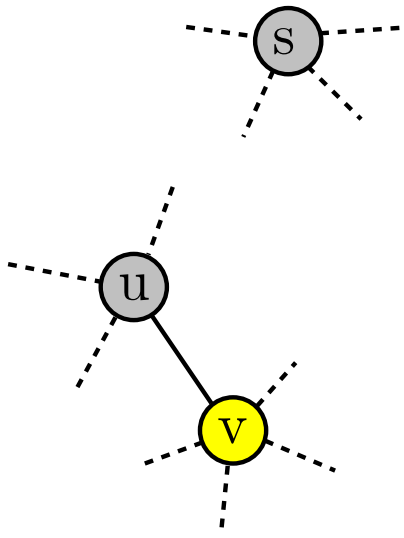


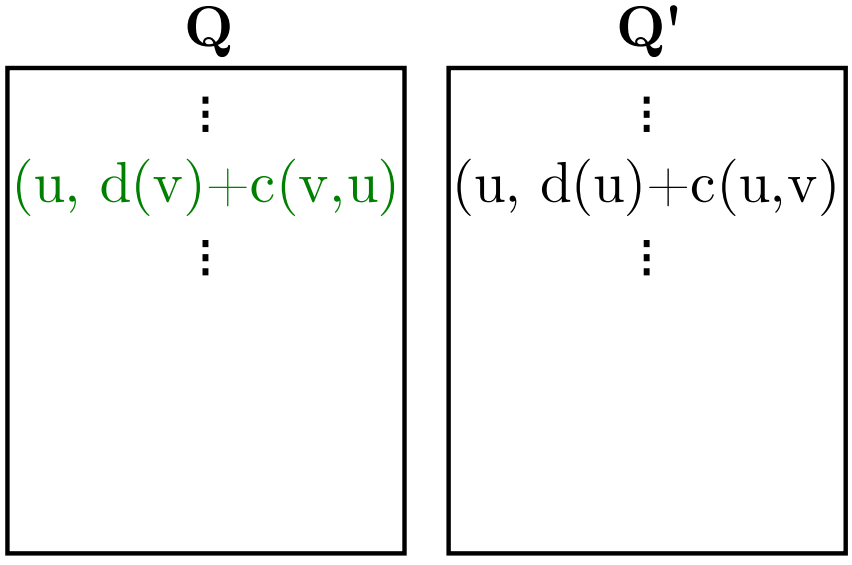


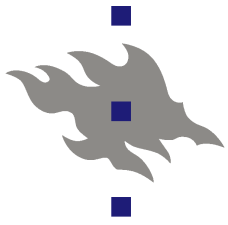


Lyhimmän polun etsintä (2)

Esimerkki (4)

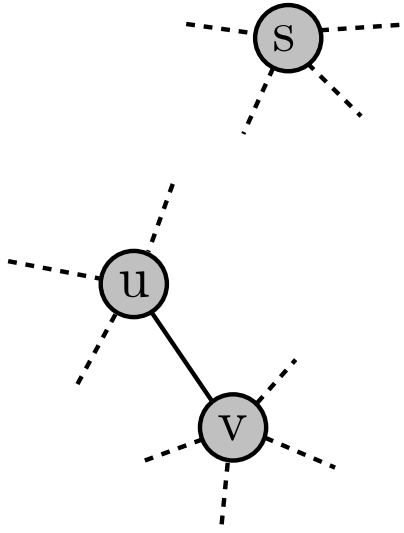


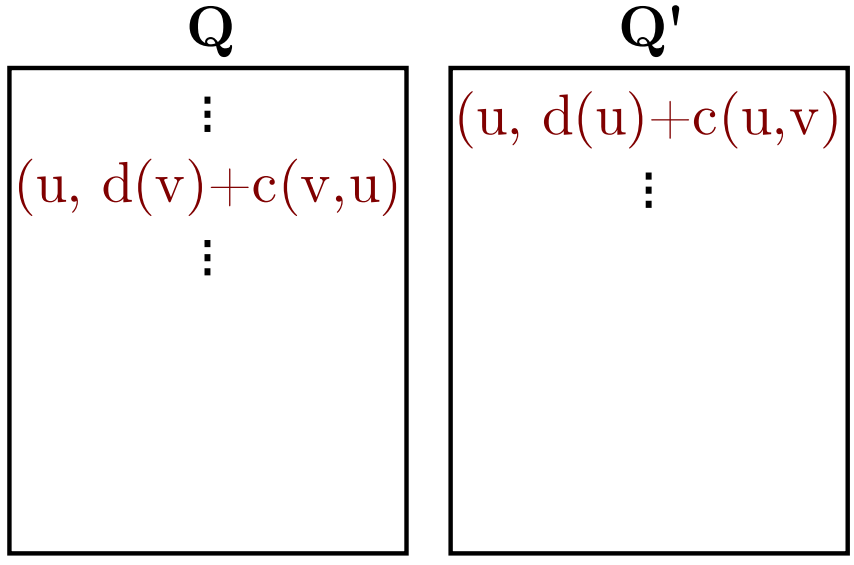


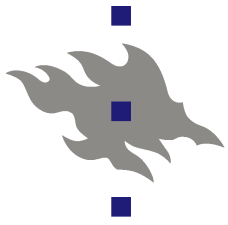


Lyhimmän polun etsintä (2)

Esimerkki (5)

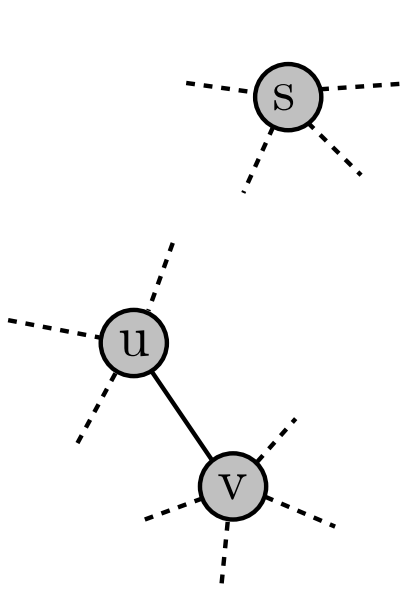


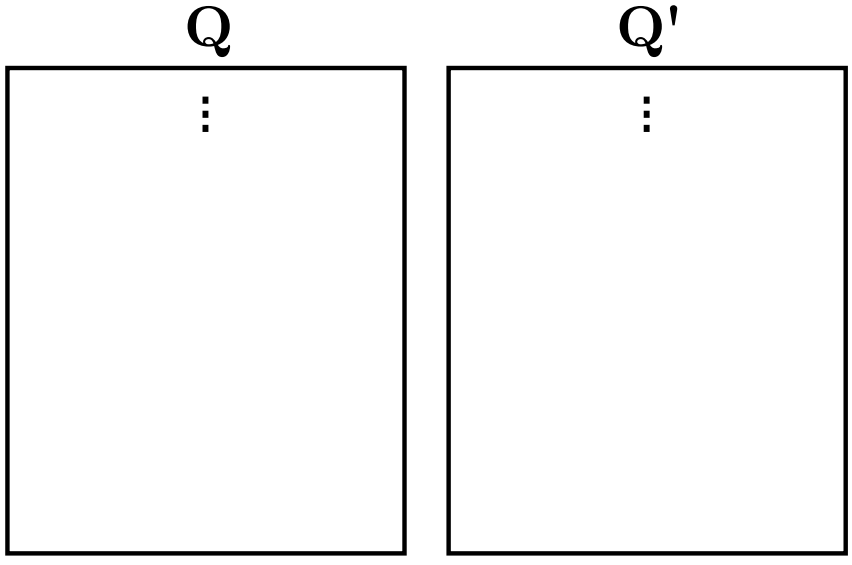


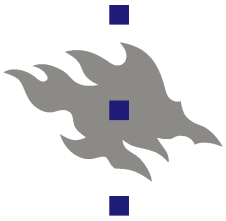


Lyhimmän polun etsintä (2)

Esimerkki (6)

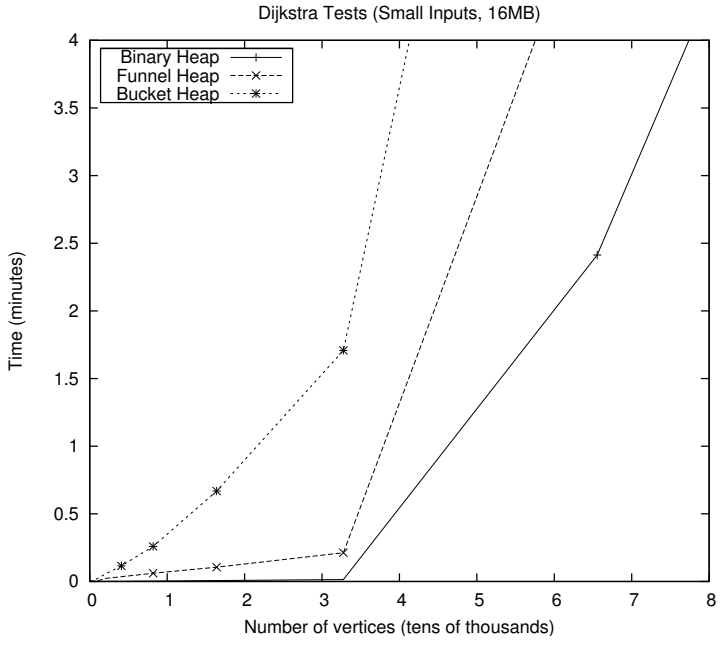


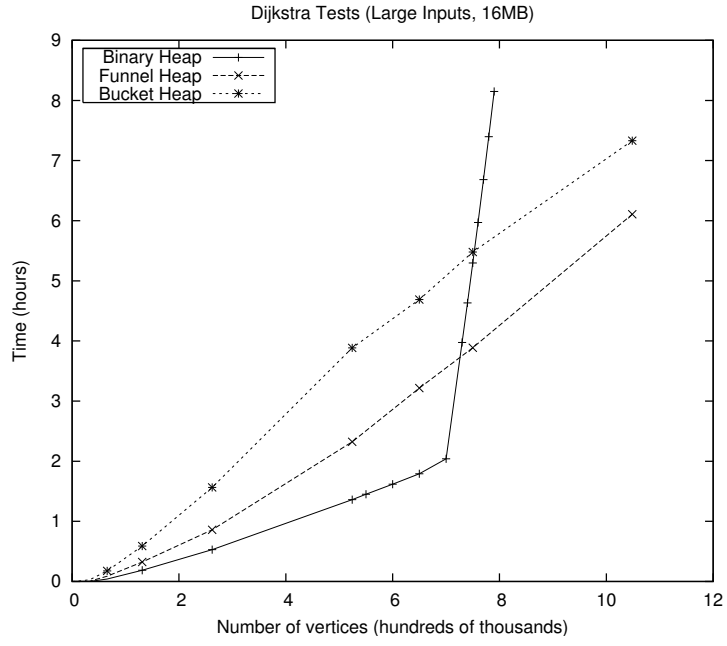


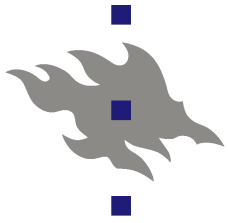


Testituloksia

Satunnaisilla verkoilla

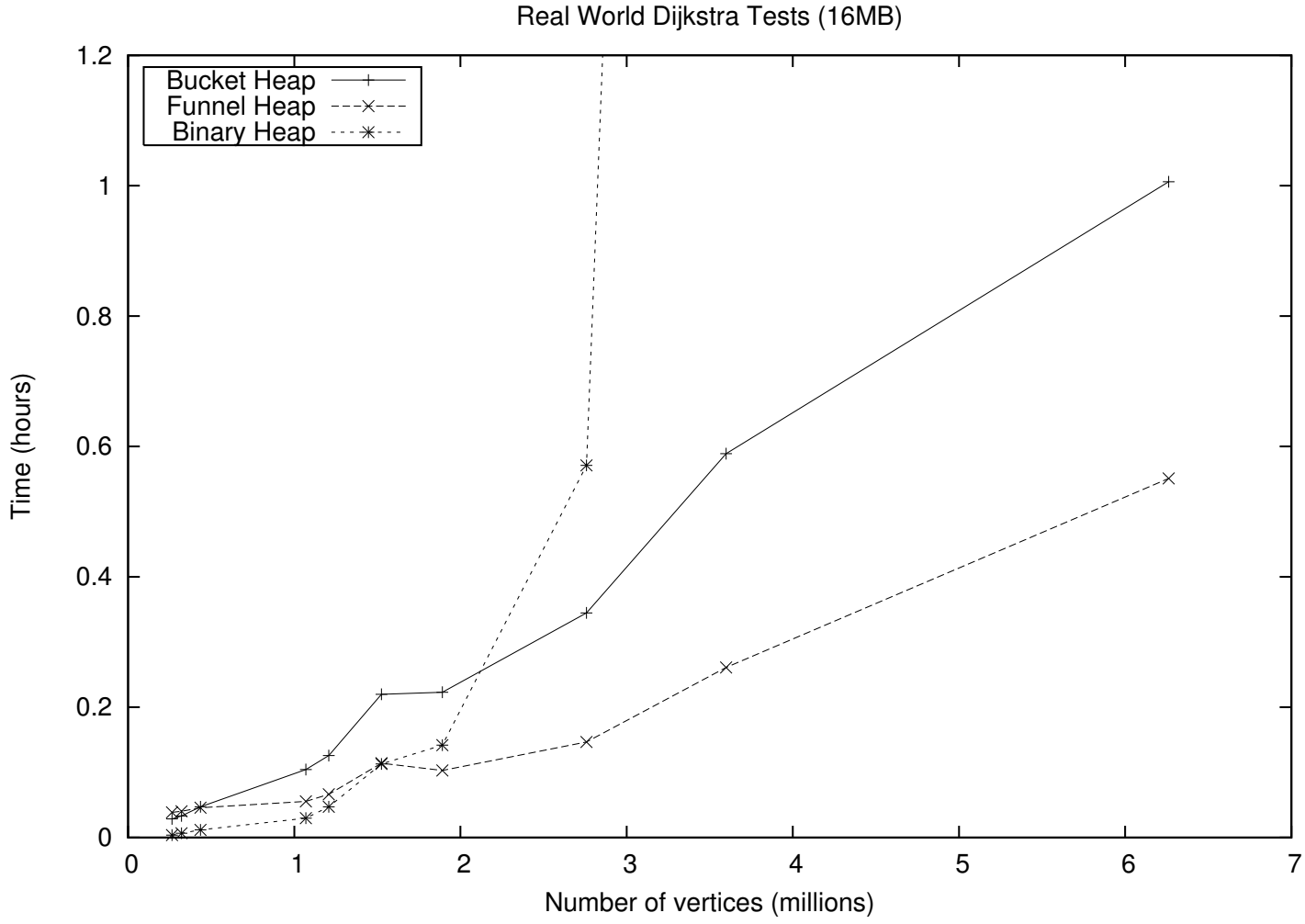


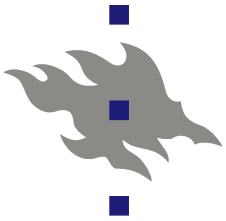




Testituloksia (2)

Todellisilla USAn tieverkostoilla





Lopuksi

- Muistitasotehokas lyhimmän polun etsintä
  - Dijkstran algoritmin muunnelmia erilaisilla prioriteettijonoilla
  - hyviä tuloksia sekä teoriassa että käytännössä
- Erikoistapauksiin muita algoritmeja
  - tasoverkot
  - rajoitetut kaarten painot