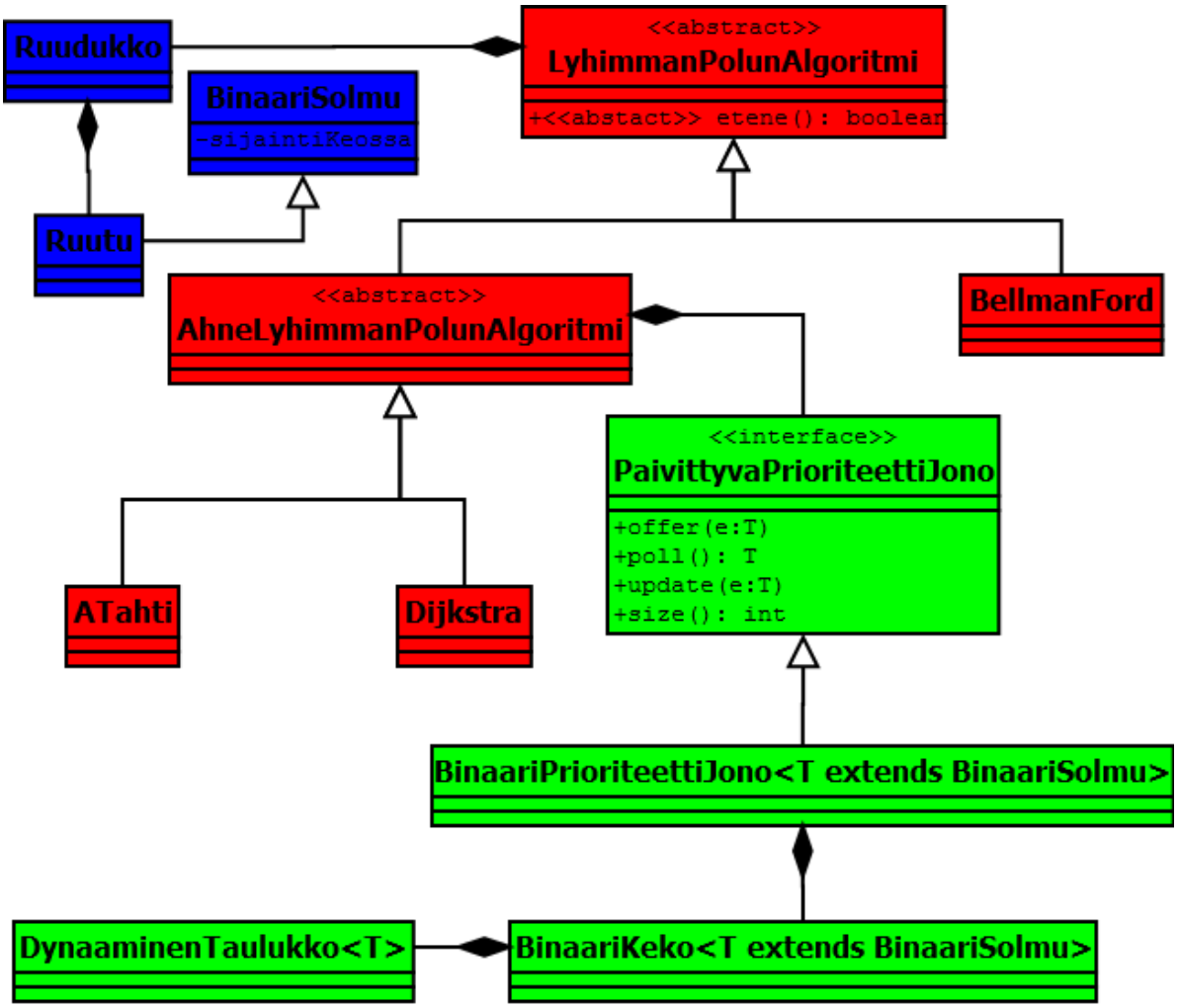
Antti Kaikkonen

Toteutusdokumentti

Ohjelman rakenne

Ohjelma koostuu 20 luokasta jotka on pyritty toiminnallisuuden mukaan jakamaan järkeviin pakkauksiin. Tärkeimpiä pakkauksia ovat ruudukko, algoritmit ja tietorakenteet. Algoritmit pakkaus sisältää kolme algoritmia: Dijkstra, Atahti ja BellmanFord. Kaikille algoritmeille yhteisiä ominaisuuksia toteuttaa ja määrittelee LyhimmanPolunAlgoritmi luokka. AhneLyhimmanPolunAlgoritmi puolestaan toteuttaa yhteisiä ominaisuuksia luokille Dijkstra ja Atahti.

Tietorakenteet pakkauksessa PaivittyvaPrioriteettiJono määrittelee eri tietorakenteille yhteiset ominaisuudet, joita algoritmit Atahti ja Dijkstra tarvitsevat. Toteutin PaivittyvaPrioriteettiJono:n vain binäärikekona, mutta ohjelmaan on jätetty myös tämän luokan toteuttava JavaPriorityQueue vertailun vuoksi. Javan prioriteettijono ei kuitenkaan kuulu kurssiin.



Ohjelman tärkeimmät luokat ja niiden väliset suhteet. Siniset kuuluvat ruudukko pakkaukseen, punaiset algoritmit pakkaukseen ja vihreät tietorakenteet pakkaukseen.

Saavutetut aika- ja tilavaativuudet (m.m. O-analyysi pseudokoodista)

Aikavaativuus

Algoritmit toimivat niin kuin oli odotettuakin eli A-tähti oli selvästi nopein ja sen aikavaativuus kasvoi keskimäärin Dijkstraa hitaammin. Bellman-Ford puolestaan oli selvästi hitaampi ja sen aikavaativuus kasvoi nopeasti syötteen koon kasvaessa. Suorituskykytestauksen (ks. testausdokumentti) pohjalta vaikuttaa siltä että tavoitellut aikavaativuudet on tullut saavutettua.

Tilavaativuus

Bellman-Fordin tilavaativuus on O(n) sillä se käy koko syötteen useaan otteeseen läpi.

Dijkstra:n ja A-tähden keko käyttää dynaamista taulukkoa, eli se käyttää tilaa sitä enemmän mitä enemmän alkioita siihen syöttää. Maksimissaan dijkstralla voi kaiketi olla n solmua käsiteltävänä, joten voimme kai turvallisesti sanoa tilavaativuuden olevan O(n). Ruudukossa kuitenkin kaikki ruudut voivat kaiketi olla käsittelyssä vain hyvin pienissä ruudukoissa kuten kahden ruudun ruudukossa.

Suorituskyky- ja O-analyysivertailu (mikäli työ vertailupainotteinen)

Testausdokemntissa.

Työn mahdolliset puutteet ja parannusehdotukset

Olen ihan tyytyväinen että sain kaikki alun perin aikomani tietorakenteet ja algoritmit toimimaan ja opin kurssin aikana paljon uusia asioita. Dijkstra:n ja A-tähden olisi kuitenkin saanut vielä vähän nopeammaksi käyttämällä Fibonacci-kekoa binäärikeon sijaan. Fibonacci-keossa decrease-key:n aikavaativuus on O(1) (amortized time) binäärikeon O(log(n)) sijaan. Toteutus Fibonacci-keolla toisi kuitenki huomattavia parannuksia vain ruudukossa jossa on paljon esteitä, koska muuten decrease-key ominaisuutta ei paljon käytetä.

Bellman-Fordia olisi myös pystynyt nopeuttamaan keskivertotapauksessa lopettamalla algoritmi jos ruudukon läpikäyminen ei tuota enää muutoksia reitteihin. Tämä nopeuttaisi keskivertotapausta huomattavasti, mutta huonoimmassa tapauksessa algoritmin aikavaativuus olisi yhä sama.