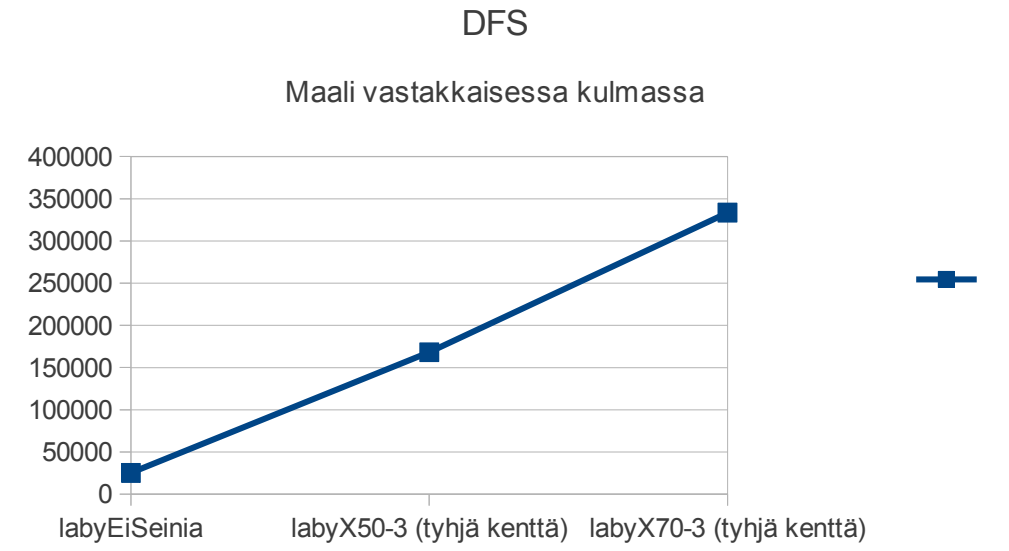
Toteutusdokumentaatio

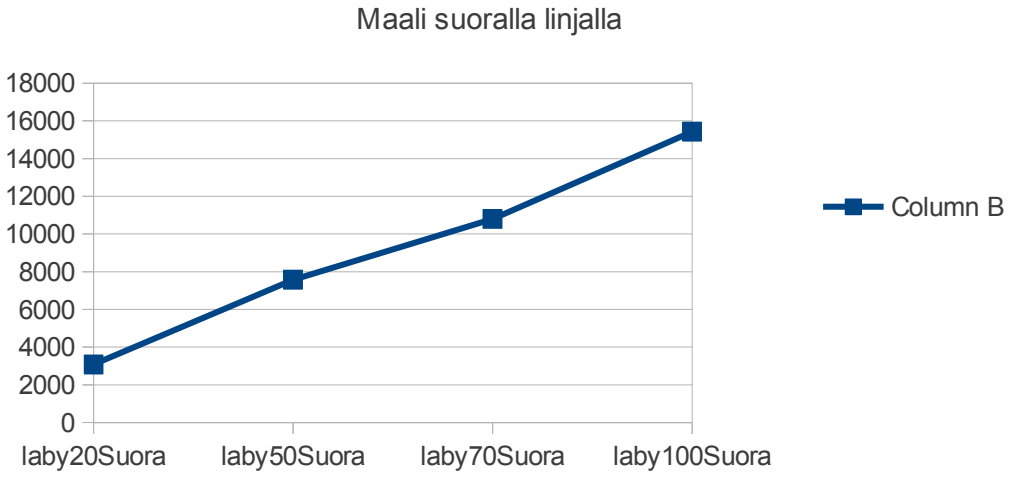
Työn rakenne muodostuu kolmesta ryhmästä:
Graafinen käyttöliittymä, polunetsintäalgoritmit ja tietorakenteet.

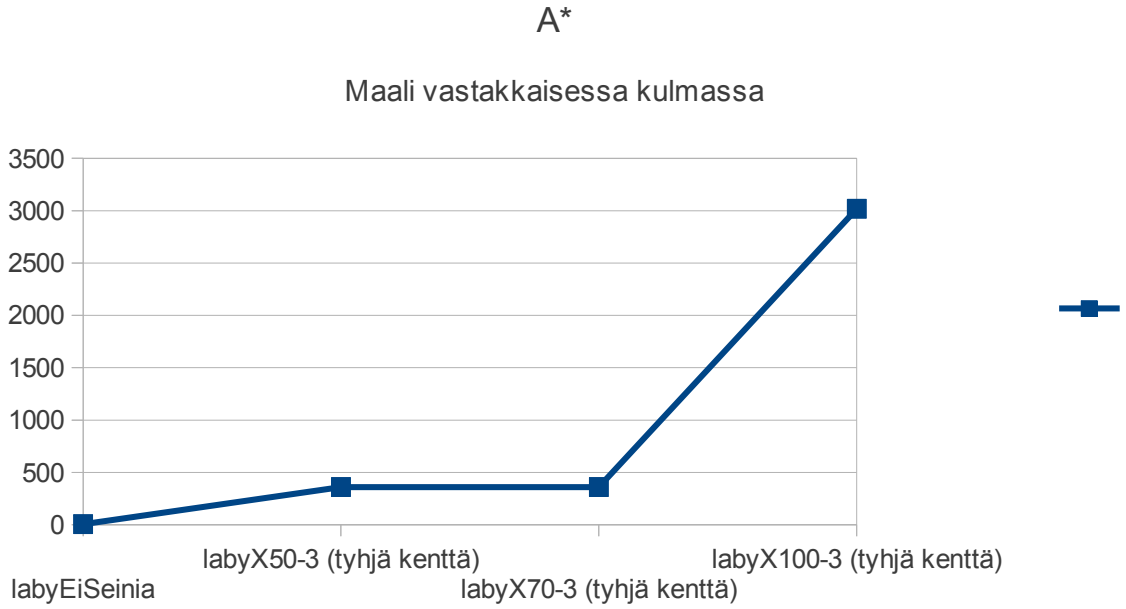
Polunetsintään käytetään A*-algoritmia, ja tietorakenteena toimiva Jarjestysjono vastaa periaatteeltaan PrioityQueue-oliota, mutta siitä saa luettua myös tietyssä indeksissä sijaitsevan arvon. Tämän lisäksi käytetään myös Lista ja Keko-nimisiä tietorakenteita. Lista on riisuttu ArrayList, jonka toiminnallisuuksia on toteutettu vain niin paljon, että ohjelma toimii virheittä. Keko-olio perii Listan, eikä varsinaisesti ole ”keko” sanan varsinaisessa merkityksessä, mutta on toiminnaltaan samansuuntainen. Työssä vertailun vuoksi käytettävä DFS-haku käyttää kekoa hyödyksi. DFS-haulla kuvasta **etsitään** maalipiste! Se ei kaikissa verkoissa anna ulos optimaalista polkua.

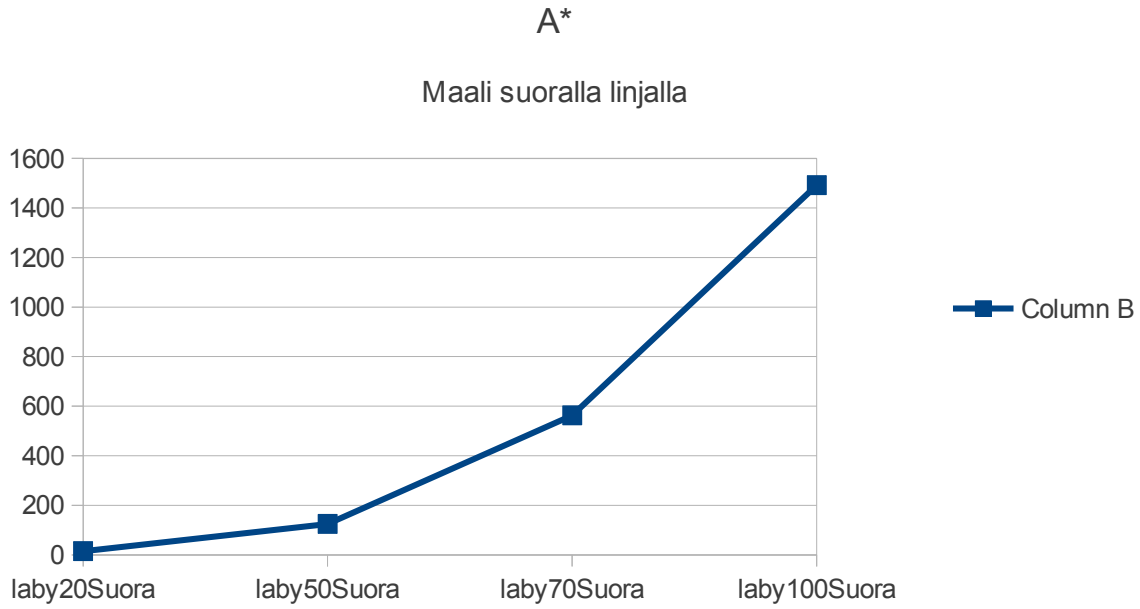
Aikavaativuusvertailu

Ohjelman aikavaativuus on A* algoritmilla eksponentiaalinen. DFS-haun aikavaativuus on laskelmien perusteella lineaarinen, mutta suurilla syötteillä päädytään StackOverflow-virheeseen, mistä johtuen ylemmässä kaaviossa on vain kolme muuttujaa.









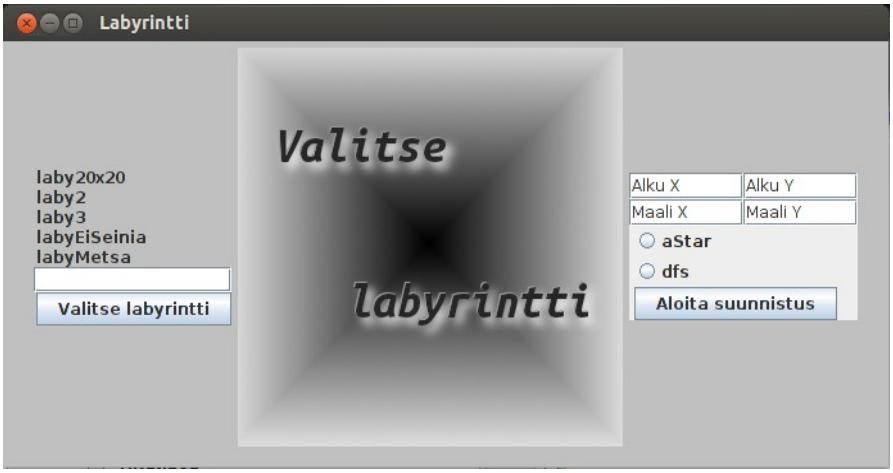
Testaukseen käytetyistä labyrinteistä vertailukelpoisia tuloksia antoivat tyhjä kenttä, jonka lähtö ja maalipiste sijaitsevat vastakkaisissa kulmissa, sekä kenttä, jonka maalit ovat samalla vaaka-akselilla, ja niiden välissä on samassa kohdalla pystylinjalla sijaitseva samanmuotoinen viiden pikselin kokoinen este.

Jos ajatellaan pelkkää aikavaativuuden funktiota, niin DFS voittaa lineaarisella aikavaativuudella. Toisaalta, jos otetaan huomioon varsinainen maalinsolmun etsintään kuluva aika, A* on huomattavasti parempi vaihtoehto, eksponentiaalisesta aikavaativuuskäyrästään huolimatta. Maalin ollessa samalla vaaka-akselilla A* oli tehokkaampi niin kauan kuin labyrintin leveys oli alle 100 solmua, ja maalin sijaitessa vastakkaisessa kulmassa testiolosuhteissa ei onnistuttu aiheuttamaan tilannetta, jossa prosessointiin kuluva aika olisi ollut samalla syötteellä sama. Lisäksi A* käy läpi vähemmän solmuja, eikä se ajautunut testattavilla syötteillä virheisiin, kun taas DFS ajautui StackOverflow-virheeseen 100*100 solmun kentässä kun esteitä oli algoritmin hakumatkalla liian

vähän. Tämä voi kuitenkin johtua myös ohjelmoijan osaamattomuudesta käsitellä virheitä.

Graafinen käyttöliittymä

Ohjelma on suunniteltu ajettavaksi graafisesta käyttöliittymästä, joka näyttää seuraavalta:



Labyrinttien nimet on kirjoitettava **täsmälleen** samalla tavalla kuin ne on kirjoitettu luettelossa, tai muuten labyrinttia ei löydy.

Puutteita ja parannusehdotuksia

Rajoitetun ajan vuoksi kaikkia toivottuja ominaisuuksia ei pystytty toteuttamaan.

- Rajoitetun toteutusajan vuoksi, ohjelman poikkeustilanteiden käsittelyssä on toivomisen varaa, ja käytönaikaisen poikkeukset ovat mahdollisia, ellei ohjelmaa ajeta juuri niin kuin käyttöohjeissa sanotaan.
- DFS-haku ei anna ulos optimaalista polkua yleisessä labyrintissa, vaan optimaalisen polun löytyminen on poikkeustapaus.
- DFS-haku ajautuu suurilla syötteillä StackOverflow-virheeseen. Tämän käsittely tai korjaus parantaisi ohjelmaa jonkin verran.
- Graafisessa käyttöliittymässä alku- ja maalipisteen koordinaatit valitaan kirjoittamalla ne tekstikenttiin, mikä saattaa aiheuttaa hankaluuksia joillekin käyttäjille. Olisi huomattavasti kätevämpää, jos pisteet olisi mahdollista valita kuvasta klikkaamalla, ja nähdä, mitä pisteitä on valinnut.
- Suunnistaja on valittava uudestaan jos koordinaatteja vaihtaa, tai muuten ohjelma olettaa, että käyttäjä aikoo ajaa saman haun uudestaan. Ohjelma olisi hyvä saada luomaan uuden suunnistajan aina kun käyttäjä painaa nappia ”Aloita suunnistus”, oli suunnistajaa valittu uudestaan tai ei.