Toteutusdokumentti

Ohjelman yleisrakenne

Ohjelmassa vertaillaan kahta reitinhakualgoritmia: A* ja Jump Point Search.

Ohjelman rakentuu:

- A* algoritmista ja siihen liittyvistä apufunktioista
- JPS algoritmista ja siihen liittyvistä apufunktioista
- Map_loader osuudesta jolla voidaan lukea sisään pelin kartta ja siihen liittyvät valmiiksi lasketut reittien pituudet alku- ja loppupisteineen
- Visualization.py osuudesta jolla visualisoidaan reitit A* ja JPS algoritmeille sisältäen A*:ssä käsitellyt pisteet ja JPS:ssä käsitellyt pisteet (jump point)
- Main.py osuudesta joka on ohjelman pääosa. Se lataa kartat, ajaa testeja ja käynnistää visualisoinnit
- Test-hakemisto kattaa testit

Saavutetut aika- ja tilavaativuudet (esim. O-analyysit pseudokoodista)

A*

Aikakompleksisuus: O(V log V) pahimmassa tapauksessa, missä V on solmujen määrä ruudukossa. Käytännössä suorituskyky on usein paljon parempi hyvän octile distance -heuristiikan ansiosta.

Tilakompleksisuus: O(V) pahimmassa tapauksessa, koska algoritmi ylläpitää useita tietorakenteita (open_set, closed_set, g_scores, f_scores, came_from), jotka voivat kaikki kasvaa V:n kokoisiksi.

JPS

Aikakompleksisuus: O(√V log V) keskimääräisessä tapauksessa, mikä on merkittävä parannus A*:n O(V log V):ään verrattuna. Paras tapaus on O(d) ja pahin tapaus O(V log V).

Tilakompleksisuus: O(√V) keskimääräisessä tapauksessa, koska JPS tutkii vain hyppypisteitä eikä kaikkia solmuja.

Metriikka	A*	JPS	Parannus
Aikakompleksisuus (keskim.)	O(V log V)	O(√V log V)	~√V kertainen
Tilakompleksisuus (keskim.)	O(V)	O(√V)	~√V kertainen
Tutkittujen solmujen määrä	V	√V	~√V kertainen
Optimaalisuus	✓	✓	Sama

Työn mahdolliset puutteet ja parannusehdotukset

- Karttojen lukeminen. Lisäkarttojen lukemiseen voisi tehdä lisätoiminnallisuutta
- A* ja JP käsiteltyjen pisteiden visualisointi ei noudata järjestystä jossa pisteet on käsitelty

Laajojen kielimallien (ChatGPT yms.) käyttö. Mainitse mitä mallia on käytetty ja miten. Mainitse myös mikäli et ole käyttänyt. Tämä on tärkeää!

- Kielimalleja on käytetty algoritmien ymmärtämiseen. Käytin teköälyä (chatGPT)tekemään yhteenvedon JPS:n ja A* teorioista
- Koodin virheiden etsintä
    - Kokeilin ChatGPT, Copilot ja Claude koodin analysoinnissa ja virheiden etsinnässä. Näistä Claude pystyi parhaiten löytämään oikeat virheet.
    - Pyysin myös niitä analysoimaan testien tuloksia ja koodia ja kertomaan missä on virhe
- Testit
    - Testattavien asioiden ehdottaminen. Käytin tähän copilotia ja claudea.
    - Testidatan luonti. Kokeilin tähän chatGPT, copilot ja claudea. Suurin osa ei osannut ehdottaa järkevää testidataa. Parhaimmillaan ne olivat testitulosten analysoinnissa
- Koodin kommentointi
    - Käytin Claudea ja ChatGPT:tä ehdottamaan Docstring -kommentteja funktioille
- Aika ja tila -analyysit
    - Käytin Claudea analysoimaan koodin

Lähteet, joita olet käyttänyt, vain ne joilla oli merkitystä työn kannalta.

- Tira kurssin Dijkstra algoritmi A* pohjaksi

- Tämän kurssin lähteissä mainitut JPS-algoritmin kuvaukset
- Wikipedia