Diskussion af algoritmer

For at beregne den korteste rute mellem attraktionerne, valgte vi i gruppen at benytte *Nearest Neighbour Algoritm*, på grund af dens hurtige eksekveringstid, der muliggøre det for brugeren af programmet, at indtaste et højt antal attraktioner, uden at det går mærkbart ud over oplevelsen med programmet. Algoritmen blev også valgt, på grund af den forholdsvis simple implementering af den, og at generelt passede godt til vores behov. Problemet ved at bruge denne algoritme, er at den ikke nødvendigvis finder den hurtigste rute. Algoritmen er upræcis, og en hvis fejlmargen bliver nødt til at accepteres, hvis ikke man vælger at gribe ind overfor algoritmen, hvis den er ved at gøre noget der tydeligt giver en længere rute end nødvendigt. Den korteste rute burde fx aldrig krydse sig selv, og det var muligvis en af de ting vi kunne have tjekket for, når algoritmen benyttes, for at sikre os at den i det mindste ikke gør det, og på den måde får en kortere rute, end hvis vi bare havde ladet den gennemføre sine beregninger.

En anden løsning, ville være at prøve alle ruter der overhovedet er for de valgte attraktioner, men eksekveringstiden stiger faktorielt med antallet af attraktioner, så der skal ikke vælges mere end et par stykker, før brugeren af programmet begynder at kunne mærke at det tager lang tid at lave beregningerne. Et scenarie vi helst gerne ville undgå.

Double Minimum Spanning Tree og Djikstras var også en algoritmer vi havde kigget på, og prøvet at implementere, men som i beskrevet i afsnittet om grafteori, så var det ikke optimalt til de behov der var for programmet.