Pathfinding algoritmer

Jens Tinggaard

1. maj 2020

Resumé

Her er en masse tekst!

Indholdsfortegnelse

1 Indledning

1.1 Målsætning

Fra projektbeskrivelsen:
Jeg vil gerne lave et program, der er i stand til at illustrere en pathfinding algoritme. Jeg vil gerne vise fordelen ved en pathfinding algoritme, dette kunne f.eks. gøres ved at sammenligne med en bruteforce algoritme. Umiddelbart vil det tage udgangspunkt i labyrinter, hvor jeg vil vise hvordan de forskellige algoritmer virker.

1.2 Krav til programmet

- Implementering af en pathfinding algoritme
- Kunne løse en labyrint
- Sammenligning med andre metoder til at løse en labyrint
- Visuel repræsentation af løsningen

1.3 Udviddelser

- Visualisering af hastighed vs. størrelse af labyrint for forskellige metoder
- Implementering af flere algoritmer (nogle af dem fra ovenstående liste)

2 Om pathfinding algoritmer

En pathfinding algoritme er en algoritme, som bruges til at finde vej over en graf. Der findes mange forskellige algoritmer, som alle har fordele og ulemper. Et par af de mest kendte er *Dijkstra*, *depthfirst*, *breadthfirst* og *A**. Alle disse algoritmer har til fælles, at de beskriver en fremgangsmåde, til at finde den korteste vej mellem to noder på en graf.

Pathfinding er brugt til alt muligt i dag, et klassisk eksempel er navigationstjenester som Google Maps. Når brugeren har indtastet en startposition og et mål, er det nu computerens opgave at finde den korteste vej derhen. Hvis man skal fra København til Rom er der måske en idé i, at optimere algoritmen, så den ikke starter med at kigge over St. Petersborg i Rusland. Alle disse overvejelser er vigtige at gøre sig, når man skal implementere en pathfinding algoritme, da forskellige algoritmer er stærke til hver deres ting.

2.1 Grafer, noder og kanter

Når man snakker om pathfinding algoritmer, vil termerne *graf* og *node* fremkomme. Begge disse typer er abstrakte og dækker over et større område inden for datalogien. En *graf* er en struktur, som indeholder et endeligt antal hjørner, også kaldet *noder*. Alle disse noder har ofte nogle bestemte attributter eller egenskaber, afhængig af hvilken type graf, der er tale om. Derudover er disse noder forbundet gennem hvad der kaldes *kanter* en kant er i bund og grund en forbindelse mellem to noder, man sætter ofte en vægt på kanterne (medmindre alle kanter er vægtet ligeligt). Denne vægt bruges også i beskæfigelsen med grafer — den er meget essentiel i forbindelse med pathfinding algoritmer.[1]

Når man har med pathfinding algoritmer at gøre, vil alle noder have følgende attributter: **Naboer** alle noder er klar over hvilke noder de ligger op ad. **Via** alle noder er klar over hvilken node der forbinder dem til startnoden. **Pris** alle noder er klar over hvor langt de har til startnoden – målt i samlet vægt af kanter op til denne – inden denne pris er bestemt vil den være ∞. **Afstand til mål** når man bruger *A**, vil alle noder også være klar over deres afstand til målnoden, denne afstand er målt direkte, såkaldt fugleflugt, hvorimod at *pris* er målt i den samlede pris fra de foregående noder + vægten af kanten fra den foregående node (*via*) til den nuværende.

2.2 Algoritmer

Det vises kort, hvordan nogle af de mest kendte algoritmer virker.

[1] `https://en.wikipedia.org/wiki/Graph_(abstract_data_type)`

2.2.1 Depthfirst

2.2.2 Breadthfirst

2.2.3 Dijkstra

2.2.4 A*

3 Brug af programmet

3.1 Krav

3.2 Eksempel

4 Udarbejdning af programmet

4.1 Idé no 1

4.2 Ændring

4.3 Idé no 2

5 Konklusion

Appendiks