Katholieke
Universiteit
Leuven

**Department of
Computer Science**

GEGEVENSSTRUCTUREN EN ALGORITMEN:

Practicum 2

Mats Fockaert r0695565
Academic year 2020–2021

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1 Inleiding

Het probleem dat we willen bekijken is het 8-puzzel probleem. We willen een algoritme maken dat efficiënt en eindig is. Wanneer een puzzel niet oplosbaar is, willen we dit meteen weten en dit melden.

We doen dit a.d.h.v. A*. Dit is een kortste pad algoritme die m.b.v. enkele heuristieken probeert zo snel mogelijk een oplossing te vinden.

2 Methode

Zoals eerder vermeld gebruiken we enkele heuristieken, namenlijk, de hammingfunctie en de manhattanfunctie. De hammingfunctie telt het aantal tegels die fout staan en de manhattanfunctie telt het aantal vakjes nodig om een tegel op de correcte plaats te zetten. Bij beide functies tellen het aantal verschuivingen om bij de toestand te geraken ook mee. Deze bereken een kost bij een bordpositie waaruit we een prioriteit kunnen opstellen. Hiervoor maken we gebruik van een prioriteitsrij die alle borden a.d.h.v. hun heuristieke kost sorteert. De prioriteitsrij wordt geinitialiseerd met het start bord. Dan worden alle buurposities – posities die te bereiken zijn door één tegel te verschuiven – in de rij opgeslagen. Het bord aan het begin van de rij heeft nu de laagste kost, deze nemen we uit de rij en herhalen het proces.

Enkele optimalisaties aan A* voor dit probleem:

- we berekenen bij de start van het algoritme of het beginbord wel effectief oplosbaar is,
- we kijken bij het zoeken naar buren of deze al voorkwamen bij het huidige bord zijn voorganger, zo niet, voegen we de buur toe aan de rij.

3 Resultaten

Puzzel	Aantal verplaatsingen	Uitvoeringstijd Hamming	Manhattan
puzzle28.txt	28	1.533	0.044
puzzle30.txt	30	3.375	0.055
puzzle32.txt	32	/	2.170
puzzle34.txt	34	/	0.472
puzzle36.txt	36	/	4.272
puzzle38.txt	38	/	4.842
puzzle40.txt	40	/	2.245
puzzle42.txt	42	/	18.950

Tabel 1: Bevat het minimum aantal verplaatsingen om de doeltoestand te bereiken en, de uitvoeringstijd (in seconden) van het A* algoritme voor de Hamming en de Manhattan prio-riteitsfuncties. Wanneer een oplossing niet binnen een redelijke tijd (¡5 minuten) gevonden kan worden, staat er een /.

4 Bespreking van de resultaten

4.1 Complexiteit van de prioriteitsfuncties

De hammingfunctie moet voor elk vak beslissen of het correct geplaatst is in de puzzel.
De manhattanfunctie moet voor elk vak kijken hoeveel vakken een tegel fout staat. Bij beide gaan we dus naar elk vak moeten kijken of het al dan niet fout is.
Voor een bord van grootte N hebben we $N \times N$ arrayelementen. We moeten dus N^2 keer over het bord gaan. De complexiteit is dus voor beide $\sim N^2$.

4.2 Implementatie isSolvable

De oplosbaarheid berekenen gebeurt als volgt:

1. Vind het lege vak. Hiervoor moeten we heel het bord afgaan. Stel we hebben een bord van grootte N dan moeten we N^2 elementen afgaan. Gemiddeld is dit $\frac{N^2}{2}$ elementen.

2. Zorg ervoor dat het gegeven bord zijn nulvak in de rechteronderhoek bevindt. Om het gemiddelde te berekenen, kijken we naar het beste en slechtste geval:

   - In het beste geval, staat het lege vak reeds correct en zijn er 0 verschuivingen nodig.
   - In het slechtste geval, staat het lege vak in de linkse bovenhoek. We hebben dan, voor een bord van grootte N, $N-1$ verplaatsingen naar beneden en naar rechts nodig. Dit geeft $2(N-1)$ verplaatsingen.

   Gemiddeld geeft dit dan $N-1$ verplaatsingen. De functie gaat één rijtoegang voor het nulvak, één voor het te verwisselen vak en één voor de wissel zelf nodig hebben. Dit geeft 3 rijtoegangen per wissel. We hebben dus $3N-3$ rijtoegangen.

3. Het opslaan van de nieuwe plaatsen. Het hele bord wordt hiervoor afgelopen, en de locatie wordt voor elk element (behalve het lege) in een nieuwe rij gestoken. Dit geeft een totaal van $2N^2-1$ aantal rijtoegangen.

4. Voer de volgende functie uit gebruikmakende van een hulpfunctie die de positie van elk gevraagd cijfer teruggeeft:

$$oplosbaar(b) = \frac{\prod_{i<j}(p(b,j) - p(b,i))}{\prod_{i<j}(j-i)}, \tag{4.1}$$

   De formule zal $\frac{N(N-1)}{2}$ keer opgeroepen kunnen worden. Hiervoor moeten we per keer twee elementen opvragen wat dus voor $N(N-1)$ rijtoegangen zorgt.

5. Kijk of oplosbaar(b) een resultaat dat ≥ 0 teruggeeft, dan is het oplosbaar.

Wanneer we dit alles optellen, krijgen we de uiteindelijke tijdscomplexiteit van de solver:

$$\frac{N^2}{2} + 3N - 3 + 2N^2 - 1 + N(N-1) = 3N^2 + \frac{N^2}{2} + 2N - 4 \rightarrow \sim 3N^2$$