Concentrische doolhoven

Datastructuren, UvA. *8 mei 2008*
0440949, Andreas van Cranenburgh

Samenvatting

Concentrische doolhoven bestaan uit ringen die zijn onderverdeeld door muren en doorgangen bevatten tussen consecutieve ringen. Vanuit een beschrijving van de posities van muren en openingen kan een graaf worden opgebouwd. Deze graaf biedt de mogelijkheid om het doolhof met een standaard breedte-eerst zoektocht op te lossen.

1 Probleemstelling

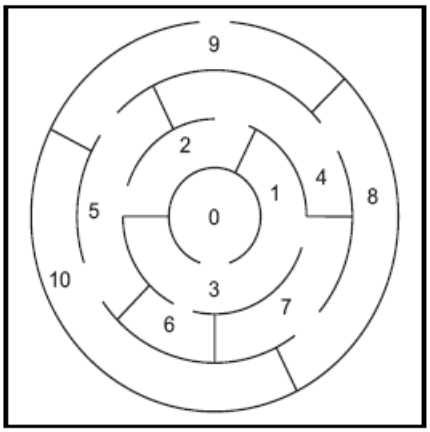
Het formaat van het doolhof is gegeven door allereerst het aantal ringen, n. De volgende n regels bevatten paren die per ring de doorgangen beschrijven. De laatste $n-1$ regels bevatten de posities van muren per ring ($n-1$ omdat de startpositie niet moet worden onderverdeeld).

Alle posities zijn gegeven in graden en lopen tegen de klok in, waarbij $0 = 360$ rechts-horizontaal staat. Het feit dat er sprake is van terugloop geeft een probleem, in het geval dat een compartiment beschreven wordt door een muur na en voor het beginpunt zal dit compartiment de 0 as overschrijden. In dit geval wordt er 360 graden opgeteld bij de "teruggelopen" posities van muren en doorgangen.

Voorbeeld:

```
4
85 95
26 40 120 130 198 215 305 319
67 90 161 180 243 256 342 360
251 288
45 135 300
0 100 225 270
65 180
```

Dit stelt het volgende doolhof voor:



2 Implementatie

De implementatie is gedaan in Java. Het doolhof wordt ingelezen op standaard invoer; deze kan eventueel gemakkelijk worden ingelezen van een bestand.

Bij het inlezen van de posities van doorgangen wordt direct voor iedere doorgang een knoop aan de graaf toegevoegd. Deze knopen worden geïdentificeerd door de ring en posities van de doorgang.

Hierna volgen de posities van de muren. Bij elke paar van muren wordt de lijst met knopen afgegaan en gekeken of de knoop een doorgang beschrijft die binnen dit paar muren valt. Dit betekent dat de positie van de doorgang binnen de positie van het paar muren moet vallen, en dat de doorgang zich op dezelfde, of op een diepere ring bevindt. Hierbij wordt rekening gehouden met posities die de 0 as overschrijden. De gevonden knoop wordt toegevoegd aan een lijst van te

verbinden knopen, en alle voorgaande knopen in deze lijst worden verbonden met de laatstgevonde knoop.

Als een volledige graaf is opgebouwd van de invoer wordt deze vervolgens doorlopen met een breedte-eerst zoektocht, beginnende bij de eerste (buitenste) doorgang, en eindigend bij het doel, de laatste (binnenste) doorgang. Dit levert het korste pad van de buitenste naar de binnenste ring op.

3 Resultaten

De resultaten tonen de knopen als doorgangen, genummerd beginnende bij 0.

```
De graaf:
0: [2]
1: [4, 5]
2: [0, 3, 6]
3: [2, 6]
4: [1, 8]
5: [1, 6]
6: [2, 3, 5]
7: [8, 9]
8: [4, 7, 9]
9: [7, 8]

Kortste route:
[0, 2, 6, 5, 1, 4, 8, 9]
```

4 Referenties

De broncode: `https://unstable.nl/andreas/ai/ds/opdr5/` (opstarten als `java Maze < maze.txt` om het voorbeeld doolhof op te lossen.)
De implementatie voor breedte-eerste zoektocht is ontleend aan Koffman and Wolfgang, Objects, Abstraction, Datastructures and Design using Java, Ch. 12.