Logisch programmeren 2012

Bonusopgave Week 8

1 Achtergrond: perfecte doolhoven

Een perfecte $n \times m$ doolhof is een acyclische gerichte graaf die elke cel van de $n \times m$ matrix bezoekt. In de bonusstartcode `maze.pl` vind je een aantal nuttige predicaten om zulke doolhoven te maken en te visualiseren.

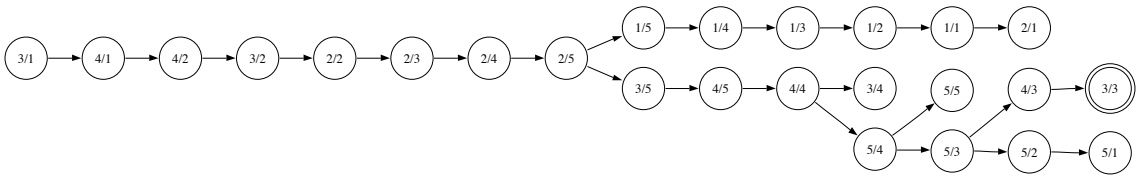
`create_maze(Maze,W,H)` maakt een perfecte doolhof met dimensie `W`×`H`. De ingang is onderaan in het midden (afgerond naar boven, als de breedte even is). De doelcel ligt in het midden (eveneens afgerond bij even dimensies). `Maze` wordt gerepresenteerd in de vorm van een recursieve datastructuur `t(Knoop,Kinderen)` — een boom. `Knoop` is een paar `X/Y:[N,E,S,W]`. `X/Y` zijn de coördinaten van een cel. `[N,E,S,W]` is een 4-bits code die aangeeft welke zijden van de cel een doorgang hebben (1) en welke dicht zijn (0).

`store_maze(Maze)` schrijft de overgangen in een doolhof weg als feiten voor `s/2`.

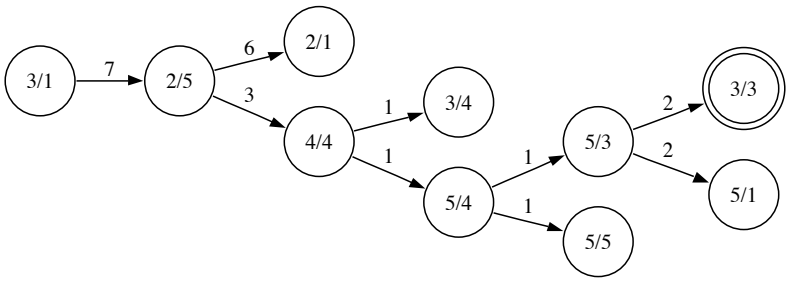
`compact(Tree,Compact)` maakt een compacte representatie van de doolhofboom `Tree` door niet-vertakkende overgangen samen te trekken. In `Compact` wordt bijgehouden hoeveel enkele stappen er zijn samengetrokken.

`store_compact(Maze)` schrijft de overgangen van de compacte representatie weg als feiten voor `s/3`.

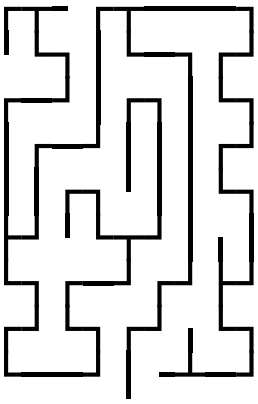
`maze2dot(Maze,W,H)` maakt een grafische visualisatie `maze.dot`. Zo'n dot bestand kan je bekijken met Graphviz. Een voorbeeld:



`compact_maze2dot(Maze,W,H)` doet hetzelfde voor de compacte representatie. Het dot bestand heet dan `compact_maze.dot`. De compacte versie van de graaf hierboven:



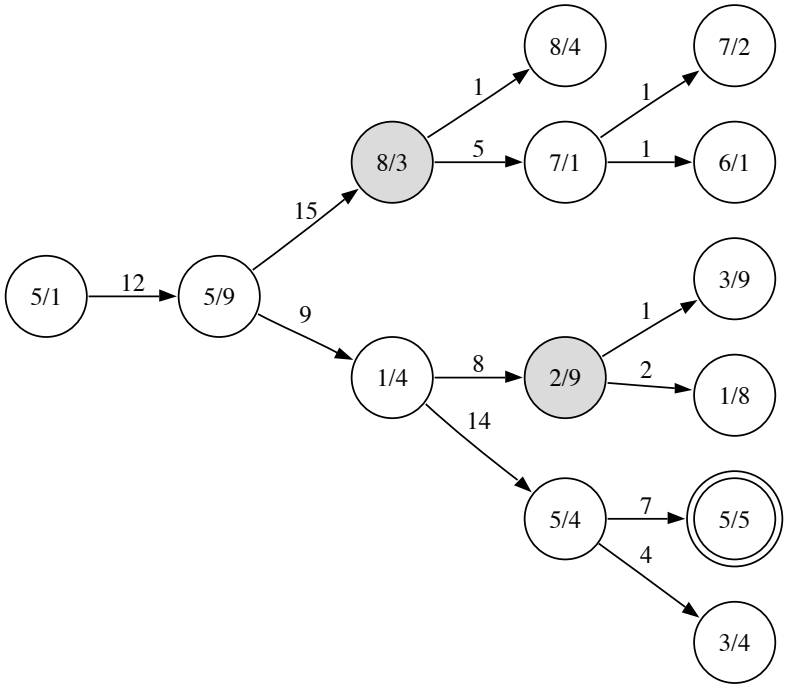
Verder wordt een testbestand `deep.pl` geladen met veertig 9×9 doolhoven. De data zijn gecodeerd als feiten van de vorm `deep(Nummer,Doolhof)`. Het bestand `deepmazes.pdf` geeft plaatjes voor de testdata. Hieronder doolhof nummer 3 uit het pdf bestand:



Met de aanroep

```
?-deep(3,Maze),compact_maze2dot(Maze,9,9),show_dot_maze.
```

krijg je de overgangsgraaf.



2 Opgave

In al deze testgevallen is de doelknoop het verst van de startknoop verwijderd. Dat houdt in dat een simpel bf regime maximaal veel overbodig werk verricht voor het pad naar de doelknoop wordt gevonden.

Implementeer best first zoeken voor de perfecte doolhoven. Gebruik de probleemspecifieke eigenschappen om een definitie voor `h/2` te schrijven waardoor het best first regime het beter doet dan het simpele bf regime.

In een perfecte doolhof is er een uniek pad van start naar doel: best first zoeken betekent hier niet zozeer dat tussen alternatieve paden het kortste pad eerst wordt gevonden, maar dat de unieke oplossing met minimaal verkennen van doodlopende paden wordt gevonden.

Hints Bekijk de voorbeelddoolhof hierboven. Op weg naar het doel kom je twee keer een tweesprong tegen op de grens van de doolhof: knopen `5/9` en `1/4`. Zulke keuzepunten kunnen een speciale rol spelen in de definitie van de heuristische functie h.

Bij de tweesprong `5/9` moet je kiezen tussen links of rechts afslaan. De keuze voor rechtsaf

is niet zinvol: ook zonder dat pad te verkennen weet je dat je nooit bij het doel zal kunnen komen zonder het pad van start naar `5/9` te kruisen. In een acyclische doolhof zijn zulke kruisingen niet toegelaten.

Voor een best first regime betekent dit dat wanneer de kandidaat opvolgers van knoop `5/9` worden geëvalueerd, de opvolger via de afslag links (`1/4`) hoger gewaardeerd moet worden dan de opvolger via de afslag rechts (`8/3`). Sterker: omdat je weet dat het pad via `8/3` nooit naar de doeltoestand kan leiden, wil je dat de evaluatiefunctie die knoop zo ongunstig inschat dat hij nooit voor verdere exploratie wordt uitgekozen.

Bij de tweesprong `1/4` doet zich dezelfde situatie voor: ook hier kan je de keuze voor rechts afslaan uitsluiten.

Zoals gezegd, de keuzepunten op de grens van de doolhof zijn speciaal: ze laten je toe om effectief een stuk van de zoekruimte weg te snoeien zonder dat je daarbij de volledigheid van het zoekalgoritme opoffert.

Behalve voor deze speciale grensgevallen moet je ook nog een goede invulling voor $h(n)$ bedenken voor knopen n die niet vanuit een punt op de grens bereikt worden. Wat is hier een goede benadering van de werkelijke afstand van n naar de doelknoop? Denk erom dat $h(n) \leq h^*(n)$: de waarde van $h(n)$ moet kleiner zijn dan (of gelijk aan) de werkelijke afstand van n tot de doelknoop, maar die werkelijke afstand liefst wel zo dicht mogelijk benaderen. De Manhattan distance is niet zo nuttig — er kunnen muurtjes in de weg zitten tussen je positie en de doelknoop ...

□