

Logisch Programmeren & Zoektechnieken 2008

Tentamen B – 18 December 2008

Instructies:

Dit tentamen bestaat uit 11 opgaven. Er zijn in totaal 100 punten te verdienen. De maximaal toegestane tijd is 120 minuten.

Geef korte duidelijke antwoorden in het Nederlands of Engels en schrijf leesbaar. Gebruik alleen het uitgereikte examenpapier en schrijf op elk blad je naam en studentnummer.

Code hoeft niet becommentarieerd te worden, maar moet wel vrij zijn van syntax errors en singleton warnings. Ook ongewenste resultaten bij backtracken moeten voorkomen worden. NB Alle gewenste built-in predicaten mogen gebruikt worden.

Opgave 1: (6 punten)

- Leg uit wat het begrip *branching factor* inhoudt gebruikmakend van de terminologie die bij het begrip *State-Space representatie* hoort.
- Bij welk bekend probleem/fenomeen dat voorkomt bij het toepassen van zoektechnieken speelt de *branching factor* een grote rol?

Opgave 2: (4 + 10 punten)

- Vertaal het volgende prolog programma naar 1e orde predicaatlogica:

<code>big(bear).</code>	<code>dark(X):-</code>
<code>small(cat).</code>	<code>black(X).</code>
<code>brown(bear).</code>	<code>dark(X):-</code>
<code>black(cat).</code>	<code>brown(X).</code>

- Laat zien hoe prolog reageert op de volgende query door het resolutie proces weer te geven:

```
?- dark(X), big(X).
```

Schrijf de benodigde formules om naar Horn-clauses en zorg dat je bewijs te volgen is. Gebruik dus regelnummers en verwijst hiernaar bij elke afleiding. Probeer de afleidingsvolgorde aan te houden die prolog aanhoudt.

Opgave 3: (10 punten)

Een bekend zoekprobleem uit het college is de sliding puzzle.

Stel nu dat we met behulp van een ongeïnformeerde zoektechniek een zo kort mogelijke (optimale) oplossing willen vinden:

7	2	4
5		6
8	3	1

	1	2
3	4	5
6	7	8

(Ook al bestaan voor dit probleem goede heuristieken en ligt dus een geïnformeerde zoektechniek als A normaalgesproken meer voor de hand.)*

- Welke ongeïnformeerde zoektechniek levert de beste resultaten met betrekking tot de tijdscomplexiteit? Geef deze complexiteit in big O notatie (worst case, zonder berekening). Leg daarnaast ook in eigen woorden uit hoe deze techniek hier zijn voordeel behaalt ten opzichte van andere ongeïnformeerde technieken.
- Welke ongeïnformeerde zoektechniek levert de beste resultaten met betrekking tot de geheugencomplexiteit? Geef ook hier de complexiteit in big O notatie (worst case, zonder berekening) en leg weer in eigen woorden uit hoe het algoritme zijn voordeel behaalt.

Opgave 4: (12 punten)

Gegeven de volgende puzzel:

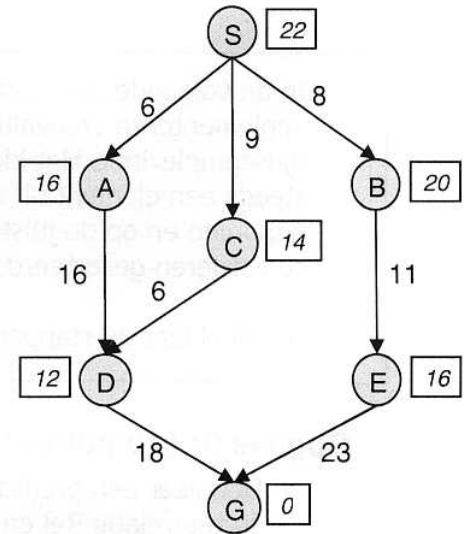
"Een boer heeft inkopen gedaan en moet een rivier oversteken om thuis te komen. Hij heeft een geit, een vos en een zak bonen gekocht. In de boot past naast de boer steeds maar hoogstens één aankoop. Als de geit en vos alleen gelaten worden zal de vos de geit opeten. Als de geit en de zak bonen alleen worden gelaten zal de geit zijn honger stillen. Kan de boer veilig zijn aankopen thuisbrengen?"

Definieer de predicaten *move/2* en *goal/1*, zodat dit probleem op te lossen is met de in de cursus gebruikte zoektechnieken.

Geef de representatie van de begintoestand en leg aan de hand hiervan uit hoe je toestandsrepresentatie in elkaar zit.

Opgave 5: (10 punten)

Bekijk de State-Space gedefinieerd door de gegeven graaf. Knoop S is de begintoestand, G is de doeltoestand. De kosten van de overgangen zijn aangegeven naast de lijnen. De getallen in de rechthoeken naast de knopen geven de heuristiekwaarde voor de toestanden weer.



In welke volgorde zal het A* algoritme deze zoekruimte doorlopen?

Laat bij elke stap zien welke kandidaatpaden in het geheugen staan en geef de berekening van hun f -waarde.

Opgave 6: (8 punten)

In de cursus hebben we gekeken naar het *knightstour* zoekprobleem: oplossingen zijn hierbij paden waarbij het paard elk vak op het schaakbord precies éénmaal bezoekt. Voor dit probleem is de regel van *H. C. von Warnsdorf* bekend. Hierbij wordt steeds gesprongen naar het veld van waaruit zo min mogelijk onbezochte velden te bereiken zijn. Deze regel is op te vatten als een heuristiek:

$$h(n) = \text{het aantal bereikbare onbezochte velden vanuit een positie.}$$

Is deze heuristiek in de context van A* *admissible* of niet?

Definieer in je antwoord het begrip toelaatbaarheid en geef een berekening of duidelijke argumentatie voor je conclusie.

Opgave 7: (8 punten)

Definieer het predicaat *Estimate/2* dat gegeven een toestand in het *knightstour* zoekprobleem de warnsdorf heuristiek uit vraag 6 berekent.

NB. Je mag er vanuit gaan dat de gangbare zoekpredicaten gedefinieerd zijn. Merk op dat de gebruikte specifieke toestandsrepresentatie niet van belang is voor dit predicaat.

Opgave 8: (6 punten)

Leg kort uit hoe het algoritme *Greedy best first search* werkt. Geef een voorbeeld van een State-Space waarbij een suboptimaal pad gevonden wordt.

In de volgende drie opdrachten zullen we het algoritme: *insertionsort* implementeren en evalueren voor wat betreft de tijdscomplexiteit. Het idee van het algoritme is dat steeds een element uit de ongesorteerde lijst wordt genomen en op de juiste plaats ingevoegd in de te construeren gesorteerde lijst.

De tabel laat de stappen zien voor de lijst 3, 7, 2, 5.

Input:	Output:
3, 7, 2, 5	
7, 2, 5	3
2, 5	3, 7
5	2, 3, 7
	2, 3, 5, 7

Opgave 9: (10 punten)

Definieer een predicaat *insert/4* dat voor een gegeven positief element X , een sorteerrelatie Rel en een gesorteerde lijst L , een nieuwe gesorteerde lijst teruggeeft waarbij element X op de juiste plaats in L is ingevoegd. VB:

```
?- insert(5, <, [2, 3, 6, 10], S).
S = [2, 3, 5, 6, 10].
```

Je mag ervan uitgaan dat het *check/3* predicaat dat we in het cursus hebben gebruikt beschikbaar is:

```
check(Rel, A, B):-
    Goal=.. [Rel, A, B],
    call(Goal).
```

Opgave 10: (10 punten)

Definieer het complete predicaat *insertsort/3* dat voor een gegeven sorteerrelatie Rel en een lijst L de gesorteerde versie teruggeeft. VB:

```
?- insertsort(<, [4, 1, 9, 6, 5, 2, 9, 1, 8], X).
X = [1, 1, 2, 4, 5, 6, 8, 9, 9].
```

Opgave 11: (6 punten)

Bij veel algoritmen verschilt de *worst case complexity* van de *average case complexity* en de *best case complexity*.

Geef aan met welke soort input de *Best case time complexity* van ons sorteerpredicaat wordt gehaald. Wat is deze complexiteit in big O notatie? Geef hiervoor een argumentatie / informele berekening.

Succes!