Antwoorden
AI Kaleidoscoop, Deeltentamen I, Oktober 2004

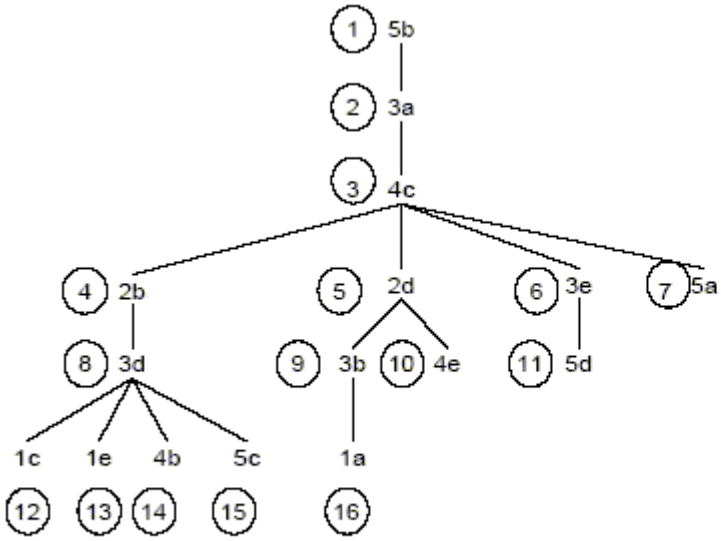
Beoordeling

Eindcijfer is score gedeeld door 10.

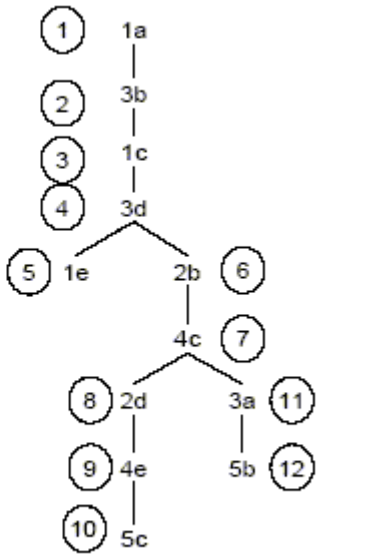
Vraag	1a	1b	1c	1d	2a	2b	2c	3a	3b	4a	4b	5a	5b
	10	10	5	10	10	5	5	10	5	10	5	10	5

Vraag 1: Depth-first en breadth-first

Vraag 1a,b:



Breadth-first achterwaards



Depth-first voorwaar

Vraag 1c:

- breadth-first:
  - voordelen:
    vindt altijd een oplossing (als die bestaat);
    vindt altijd de optimale oplossing eerst; raakt nooit verdwaald in oneindige takken
  - nadeel: heeft exponentieel veel geheugenruimte nodig
- depth-first:
  - voordelen:
    heeft maar weinig (lineaire) geheugenruimte nodig;
  - nadelen:
    vindt soms geen oplossing (want kan verdwaald raken in oneindige takken);
    vindt niet altijd de optimale oplossing eerst

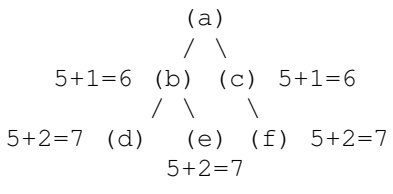
Vraag 1d:

1. Er moet een eenduidige begintoestand [eindtoestand] bestaan voor voorwaards [achterwaards] zoeken
2. kies de richting waarin de branching factor het kleinst is.

Op grond hiervan kan niet gekozen worden voor bovenstaande puzzel: er bestaat zowel een eenduidige begintoestand en eindtoestand, en de branching factor zal in het algemeen gelijk zijn (immers, elke zet kan zowel vooruit als achteruit gedaan worden).

Vraag 2: Algorithme A

Vraag 2a: Het algorithme wordt dan gelijk aan breadth-first search (zie de opmerking in boek en slides dat A* gelijk wordt aan breadth-first search als voor alle n geldt h(n)=0; voor elke andere waarde van h(n) gedraagt het algorithme zich hetzelfde).
Vb: neem h(n)=5



Het algorithme bezoekt de knopen in volgorde van de waarde f(n)=h(n)+g(n). Omdat h(n) konstant is, wordt de volgorde van f(n) volledig bepaald door de waarde van g(n), dwz op volgorde van de diepte in de boom, en dat is precies breadth-first.

Vraag 2b: Nee, want voor A* moet overal gelden $h(n) \leqslant h^*(n)$. Bij de doelknopen geldt $h^*(n)=0$, maar h(n)=5, dus geldt deze eis niet.

Vraag 2c: Als $0 \leqslant h2(n) \leqslant h1(n) \leqslant h^*(n)$.
Voordeel van een beter geïnformeerde heuristiek is dat hij sneller tot het doel zal leiden; Nadeel is dat hij (meestal) duurder is om te berekenen.

Vraag 3: Twee-speler bomen

Vraag 3a: Het algorithme heet *minimax*.
De stappen zijn:

1. geef de waarde 1 [0] aan alle bladeren waar MAX [MIN] gewonnen heeft
2. berekend de waarde van een ouder-knoop door het maximum [minimum] te nemen als de ouder een MAX [MIN] knoop is
3. als de waarde van de wortel van de boom 1 [0] is, dan heeft MAX [MIN] gewonnen

Vraag 3b: Minimax is niet bruikbaar in de praktijk omdat de volledige zoekboom bekend moet zijn, en die is voor praktische situaties niet te berekenen.

Vraag 4: Zekerheidsfactoren

Vraag 4a:
CF(tij=springvloed) = -1
CF(weer=storm) = 0.6
CF(windrichting=zuidwest) = 1

CF(sluiten) op basis van regel 1: -1 x 0.5 = -0.5
CF(sluiten) op basis van regel 2: min(0.6, 1) x 0.9 = 0.54
combinatie van deze twee waarden levert voor
CF(sluiten) = (-0.5 + 0.54) / (1 - min(|-0,5|,|0.54|)) = 0.04/0.5 = **0.08**

Vraag 4b:

1. Nadeel van de Bayesiaanse methode: heeft meestal heel veel data over voorwaardelijke kansen nodig, en die getallen zijn meestal niet beschikbaar. Dat is dus
2. Nadeel van zekerheidsfactoren: ze komen niet overeen met de wiskundig correcte conclusies van de Bayesiaanse methode.
3. Nadeel van zekerheidsfactoren: de uitkomst kan afhangen van de volgorde van de berekeningen

Vraag 5: Productie-regels

Vraag 5a:

Werkgeheugen	Conflict-set	toegepaste regel
a,c	3	3
a,c,b	1,2,3	1
a,b,d	2,3	2
a,b,d,e	2,3,4	2
a,b,d,e,e	2,3,4	2
...	...	...

Vraag 5b:

- **Refractie**: nooit tweemaal een regel vuren met dezelfde elementen uit het werkgeheugen. Dat voorkomt dat het systeem in een loop raakt.
- **Recency**: verkies matches met de nieuwste elementen in het werkgeheugen. Dit zorgt dat er een bepaalde "lijn" van redeneren wordt gevolgt, en voorkomt dat het systeem springt tussen het ene redeneerpad en het andere.
- **Specificity**: prefereer regels met de meeste condities. Regels met meer condities zijn specifieker voor de huidige situatie, en dus te verkiezen boven algemene regels die altijd toepasbaar zijn.
- **Wegingsfactoren**: prefereer regels met de hoogste wegingsfactor. Hiermee kan een programmeur zelf de volgorde van de regelselectie nauwkeurige van te voren bepalen. De wegingsfactor kan staan voor het belang van de regel, de urgentie, de betrouwbaarheid, etc.