

AI

Kunstmatige Intelligentie (AI)

Hoofdstuk 3 (vanaf 3.5) en 4 van Russell/Norvig = [RN]
Gericht zoeken en verder

voorjaar 2011 — College 5, 8 maart 2011

www.liacs.nl/home/kosters/AI/

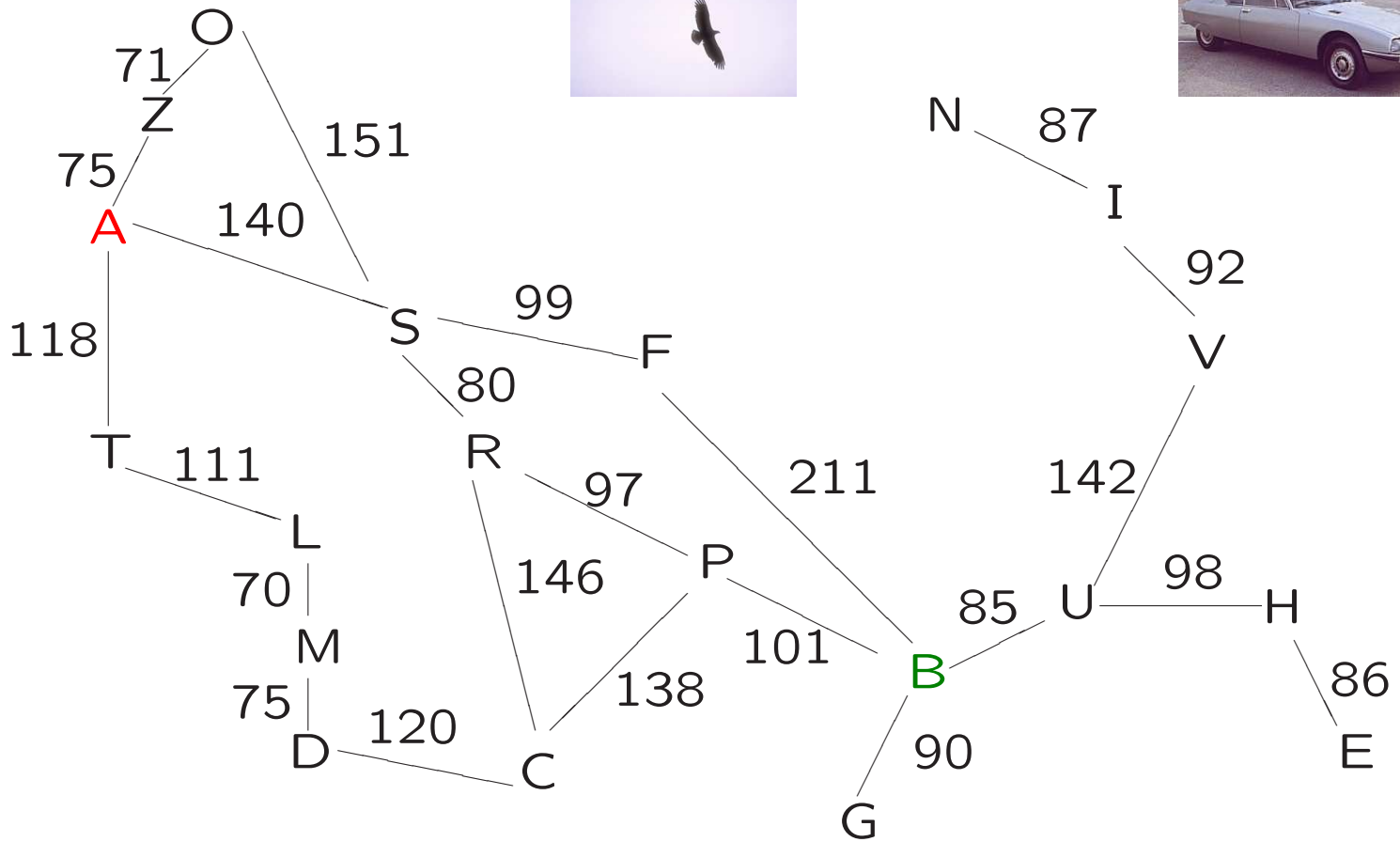
Een **informed** zoekmethode gebruikt meer dan alleen de probleem-omschrijving, ten einde beter (**gericht**) te kunnen zoeken.

We bekijken **best first** methoden, die een knoop n kiezen met de laagste waarde voor een geschikte evaluatiefunctie $f(n)$. Deze gebruikt een gegeven heuristische functie $h(n)$ die de afstand vanuit knoop n tot het doel schat; altijd moet gelden dat $h(n) = 0$ voor doelknoten.

Greedy best-first search probeert de afstand tot het doel zo snel mogelijk te verkleinen.

A* probeert de totale afstand/kosten zo laag mogelijk te houden.

Om geheugen te sparen kun je IDA* gebruiken.



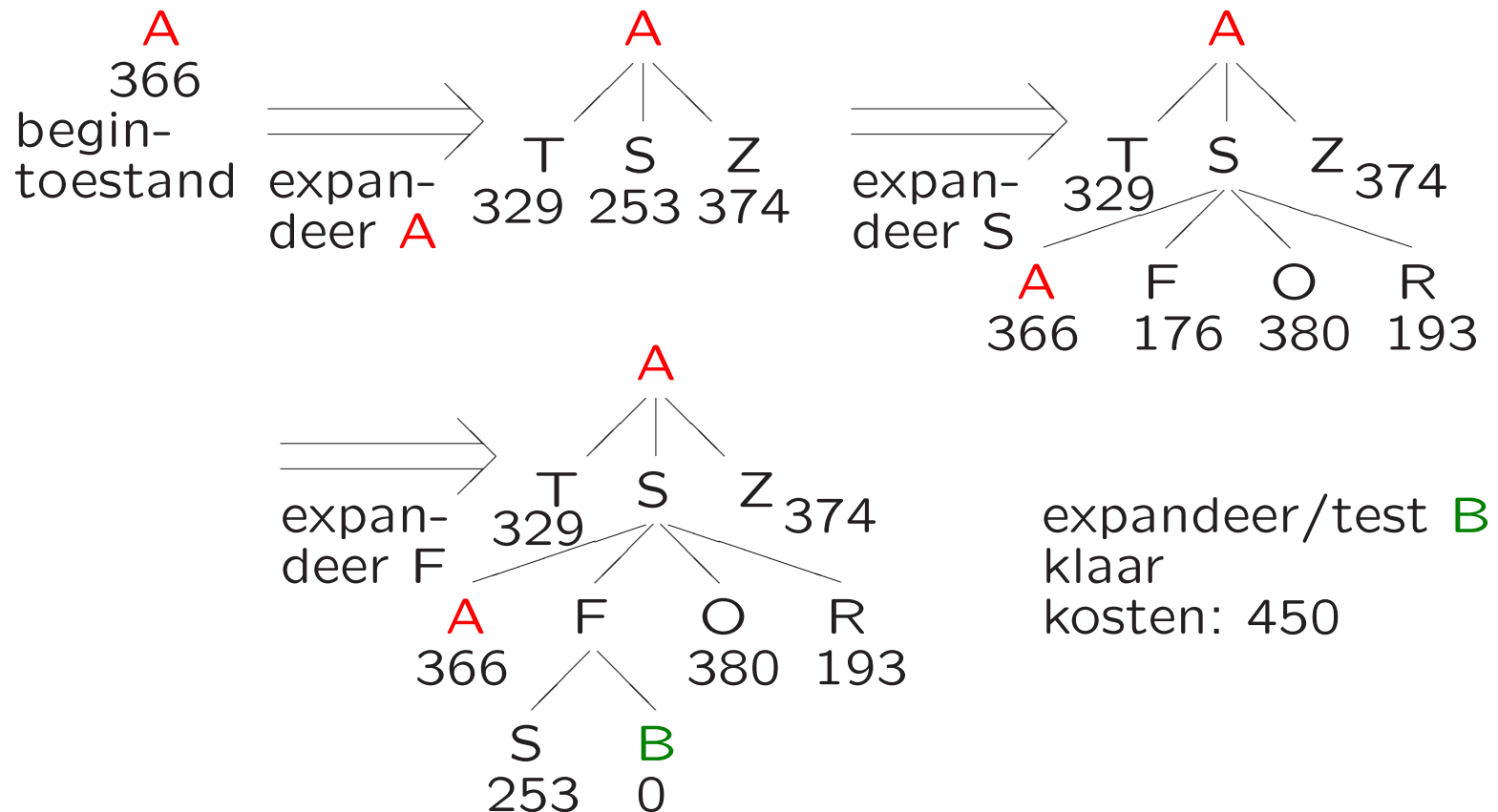
We bekijken weer de reis van **Arad** naar **Bucharest** in Roemenië.

We gebruiken als heuristiek h_{SLD} , de vogelvlucht afstand (straight line distance) tot het doel:

A	B	C	D	E	F	G	H	I	L
366	0	160	242	161	176	77	151	226	244
M	N	O	P	R	S	T	U	V	Z
241	234	380	100	193	253	329	80	199	374

Merk op dat $h_{SLD}(\mathbf{B}) = 0$.

Greedy best-first search: ontwikkel (fringe-)knoop n met de laagste $h(n)$, in dit geval $h = h_{SLD}$. Oftewel: $f(n) = h(n)$.

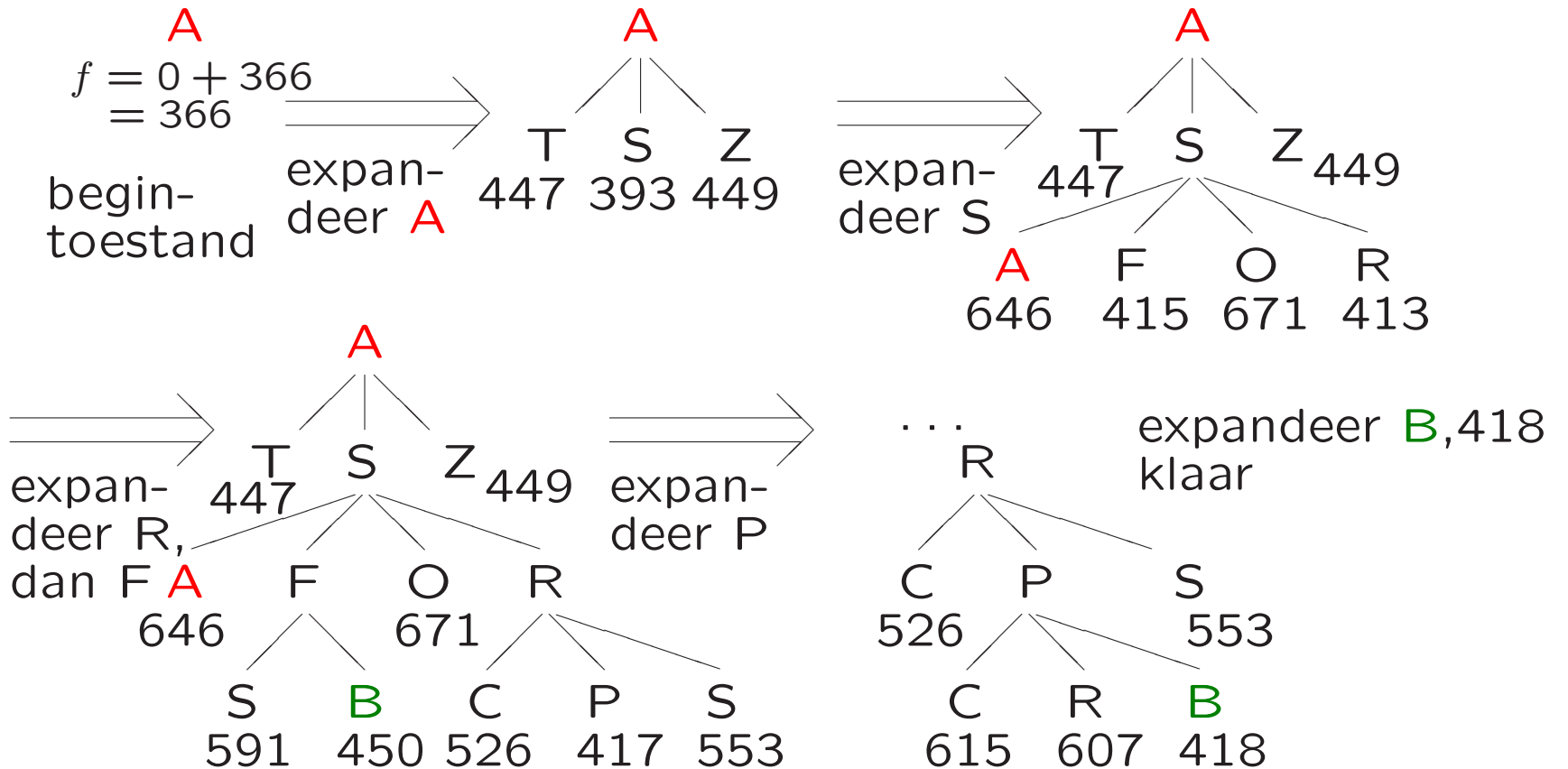


Greedy best-first search heeft tijd- en ruimtecomplexiteit $O(b^m)$, met b de vertakkingsgraad en m de maximale diepte van de zoekruimte.

Helaas: niet compleet (oneindige paden ...) en niet optimaal (korter via R en P, zie straks)!

Beter idee: breng ook de verbruikte afstand in rekening.

Voor **A*** (“A-ster”) definiëren we $f(n) = g(n) + h(n)$, een schatting voor het totale pad van begin naar doel.



Het **A*-algoritme** is dus als volgt:

Expandeer steeds de (een) knoop uit de “fringe” met de laagste f -waarde, waarbij voor knoop n : $f(n) = g(n) + h(n)$. Hierbij is $g(n)$ de exacte waarde van de reeds gemaakte kosten, en $h(n)$ de (toelaatbare) schatting voor de rest van de kosten. Stop als je een doelknoop expandeert.

Onder redelijke voorwaarden (zie straks) is A* compleet en optimaal; er zijn allerlei handige aanpassingen mogelijk. Let op “herbezoeken” van locaties.

Een heuristiek h heet **admissibel** = **toelaatbaar** als hij nooit de kosten naar het doel overschat.

Voorbeeld: h_{SLD} . Of (flauw): $h(n) = 0$ voor alle n — dit is overigens hetzelfde als uniform-cost search.

Bewering: Als h admissibel is, is A^* optimaal: A^* vindt het “beste” doel — als A^* er een ontdekt, tenminste.

Bewijs: Stel dat je een niet-optimaal doel G_2 op de “fringe” hebt. De kosten naar een optimaal doel zijn C^* , zeg. Dan: $f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$. Pak een knoop n op de “fringe”, die op een pad naar een optimale oplossing zit (zo’n knoop is er). Dan: $f(n) = g(n) + h(n) \leq C^*$ (h admissibel). Dus $f(n) \leq C^* < f(G_2)$. Dus n wordt voor G_2 ge-expandeerd.

Een heuristiek h heet **monotoon** = **consistent** wanneer altijd — als je via actie a van knoop n naar knoop n' gaat — geldt $h(n) \leq c(n, a, n') + h(n')$. Dit is een soort **driehoeksongelijkheid**.

In dat geval zijn de waarden van f op ieder pad niet-dalend:

$$\begin{aligned} f(n') &= g(n') + h(n') = g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) = f(n). \end{aligned}$$

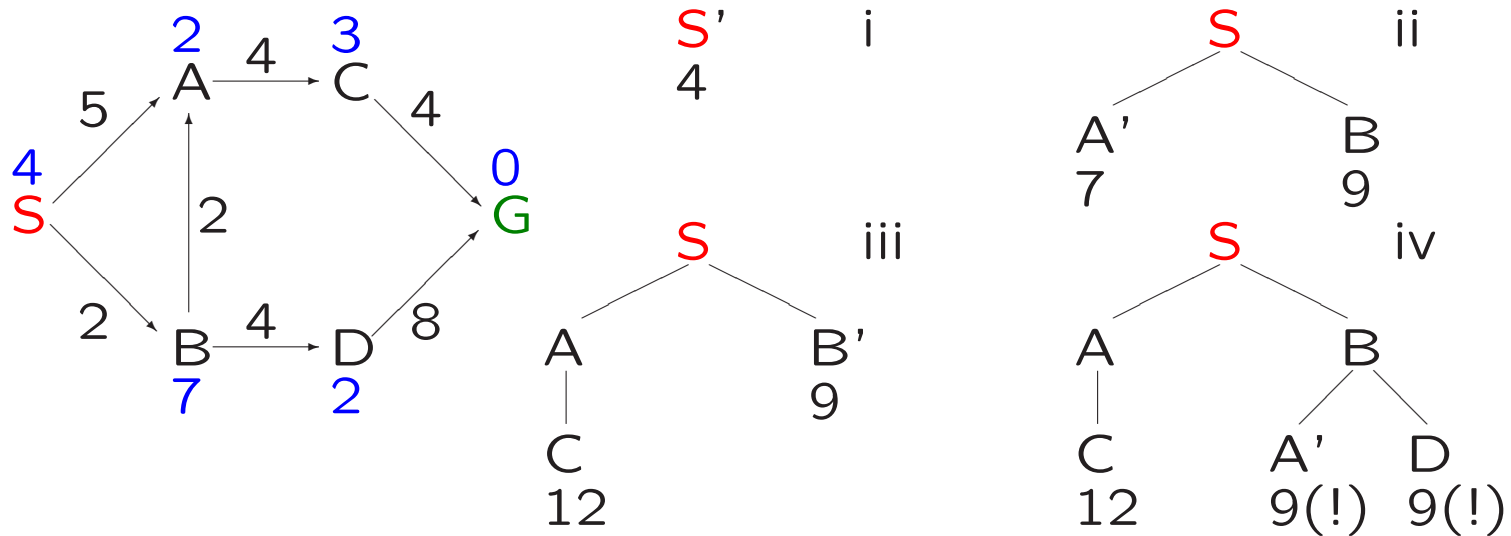
En A^* expandeert alle knopen met $f(n) < C^*$ en een of meer knopen met $f(n) = C^*$ (met C^* de kosten van een optimale oplossing) — mits dat er *eindig veel* zijn, en is dan dus ook compleet. Je krijgt een soort contourlijnen.

Stel dat de f -waarde daalt langs een zeker pad — de heuristiek is dan blijkbaar niet consistent. Wat kun je daaraan doen? Oplossing: de **pathmax-vergelijking** toepassen:

$$f(n') = \max(f(n), g(n') + h(n')).$$

Merk op: consistent impliceert admissibel (maar niet omgekeerd). Bewijs: met inductie naar de afstand tot het doel. OK als die 0 is. Stel > 0 voor zekere n . Kies de eerstvolgende n' op een kortste pad naar het doel; die heeft kleinere afstand tot het doel en dus (inductie) $h(n') \leq$ afstand tot het doel. Nu $h(n) \leq c(n, a, n') + h(n') \leq c(n, a, n') +$ afstand van n' tot het doel = afstand van n tot het doel.

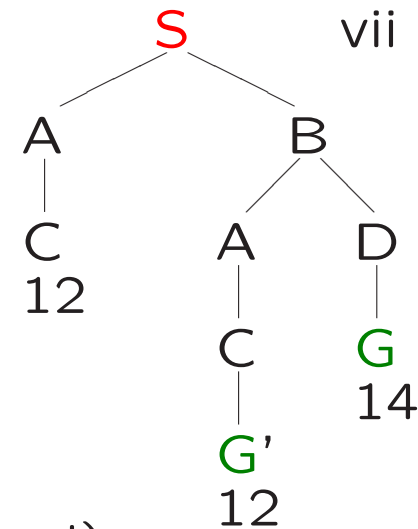
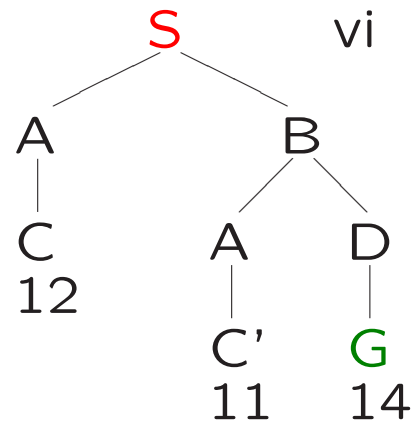
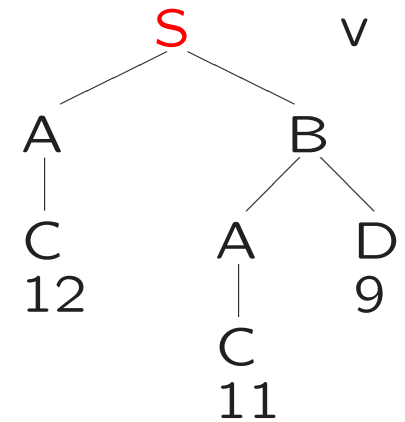
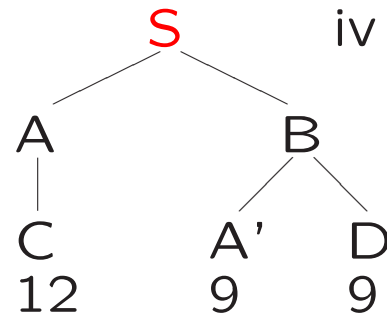
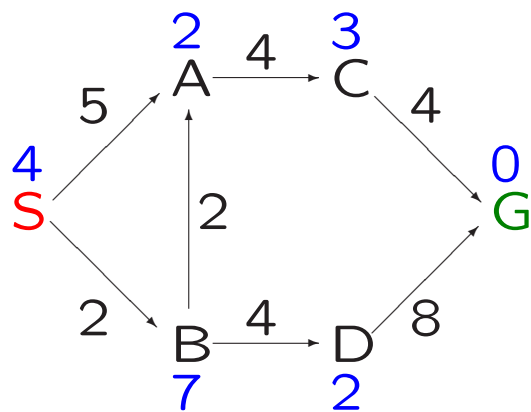
Consistent wordt wel “locaal optimistisch” genoemd, admissibel “globaal optimistisch”.



S is het Start-punt, G is de Goal (= doel).

Er staat steeds een ' bij de knoop die ontwikkeld wordt.

Merk op dat de **heuristiek** admissibel is — maar niet consistent, zie (!). Bij A' in iv krijg je eigenlijk $f = 4 + 2 = 6$, maar de ouder had 9. Dus, dankzij pathmax, 9. Idem D.

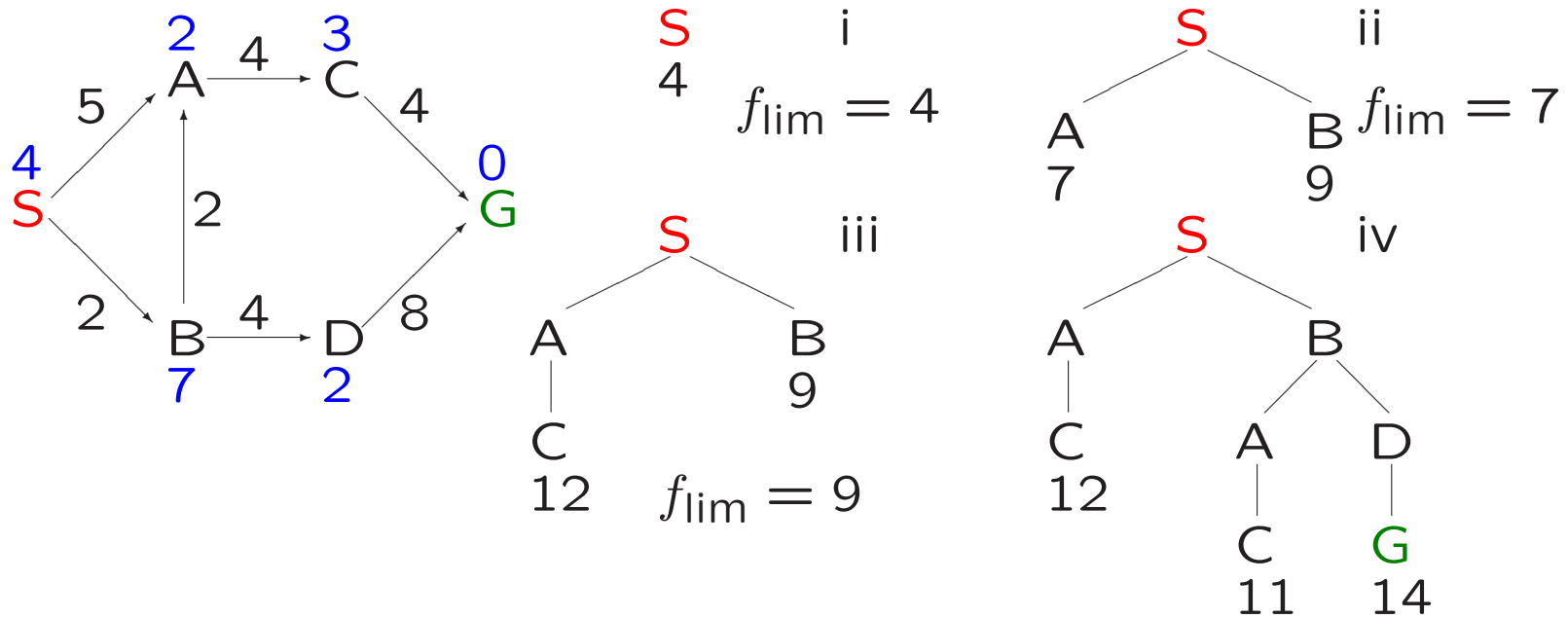


Bij iv mag je ook eerst D ontwikkelen,
 en bij vii eventueel eerst C (G is een doel).
 De kortste weg (S-B-A-C-G) is dus 12 lang.

De overgang van A* naar IDA* is analoog aan de overgang van DFS via Depth limited search naar Iterative deepening.

Bij IDA* is iedere stap een complete DFS-wandeling, waarbij je iedere knoop ontwikkelt die een f -waarde heeft hoogstens gelijk aan de laagste f -waarde f_{lim} die nog “open stond” uit de vorige stap. In de eerste doorgang is f_{lim} gelijk aan de f -waarde van de beginknoop, oftewel diens h -waarde.

Er zijn nog meer varianten: RBFS, MA*, SMA*, ... Deze laatste “vergeet” knopen met hoge f -waarden als het geheugen vol raakt.



Na iedere stap (een DFS-wandeling) is de boom steeds weg!

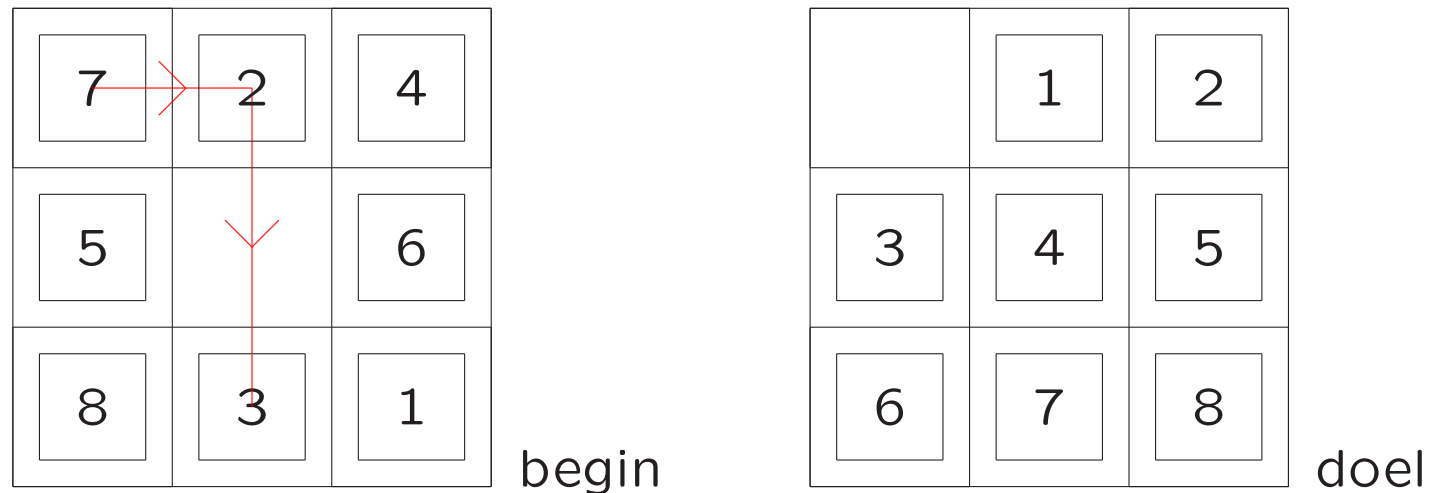
Na stap iv wordt $f_{lim} = 11$, in de volgende stap 12 en wordt dezelfde boom als in stap vii van A* doorlopen, maar nu met ook het kind G,13 van de linker C erbij.

Kortom, wat is nu het *verschil* tussen A* en IDA*, Iterative Deepening A*?

Bij A* expandeer je steeds de knoop met de laagste f -waarde uit de “fringe” die in het geheugen zit. De grootte van die “fringe” is ruwweg de “breedte” van de boom. Er is maar één boom in het spel.

Bij IDA* maak je steeds een DFS-wandeling door een (bij iedere ronde groter wordende) boom; je loopt steeds tot en met de laagste open staande f -waarde uit de vorige ronde. Nu heb je altijd alleen een pad uit de boom in je geheugen, en dat heeft met “diepte” te maken.

Er zijn meestal verschillende heuristische functies mogelijk.



Voor h_1 nemen we het aantal “tegels” dat uit positie is; voor h_2 de som van de afstanden van de “tegels” tot hun uiteindelijke positie. Als afstand kiezen we de **Manhattan-afstand** (= **city block distance**). Hier: $h_1 = 8$ en $h_2 = 18$.

Een goede maat voor de kwaliteit van heuristieken is aan de hand van de **effectieve vertakkingsgraad** b^* . Stel dat A^* N knopen genereert en een oplossing vindt op diepte d . Dan is b^* de vertakkingsgraad die een volledig gevulde boom met $N + 1$ knopen van diepte d heeft. Dus:

$$N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d.$$

Als A^* een oplossing vindt op diepte 5, met behulp van 52 knopen, geldt $b^* = 1,92$.

Hoe dichter b^* bij 1 zit, hoe beter (dan vind je “diepe” oplossingen).

Als het ware meet b^* hoe “breed” je zoekt.

Voor de 8-puzzel (gemiddeldes over 100 problemen; IDS = Iterative deepening search; d = diepte oplossing):

d	aantal ge-exp. knopen			eff. vertakkingsgraad		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	—	539	113	—	1.44	1.23
16	—	1301	211	—	1.45	1.25
18	—	3056	363	—	1.46	1.26
20	—	7276	676	—	1.47	1.27

Als voor iedere knoop n geldt dat $h_2(n) \geq h_1(n)$, zeggen we: h_2 **domineert** h_1 . A^* met h_2 zal dan doorgaans minder knopen expanderen dan h_1 (zie ook Experiment), “want” elke knoop n met $h(n) < C^* - g(n)$ wordt ge-expandeerd.

Conclusie: kies altijd een heuristiek met zo hoog mogelijke waarden — mits admissibel (nooit overschatten).

Als h_1, \dots, h_m admissibele heuristieken zijn, domineert

$$h(n) = \max\{h_1(n), \dots, h_m(n)\}$$

ze allemaal.

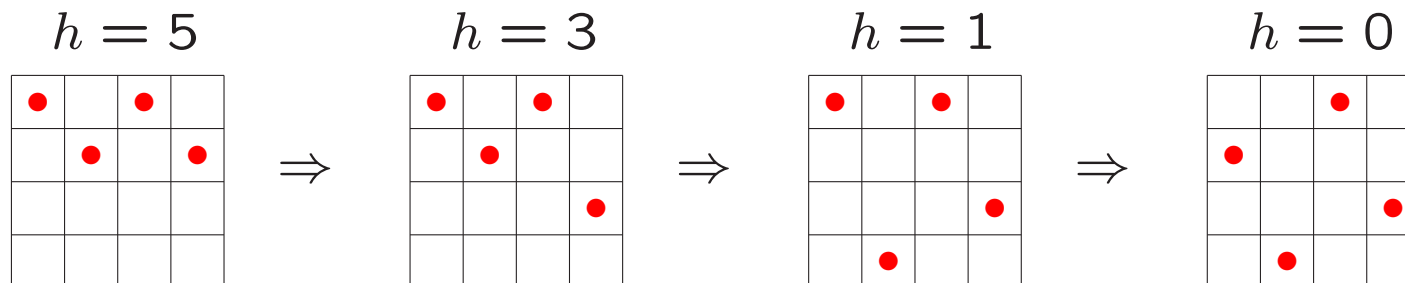
Een handige manier om een heuristiek te vinden is het exact oplossen van een vereenvoudigd (afgezwakt, **relaxed**) probleem. De optimale oplossing hiervan kost hooguit even veel als die van het oorspronkelijke probleem.

Voor de 8-puzzel: een “tegel” mag van A naar B bewegen mits A en B aangrenzend zijn *en* B leeg is. Dit zwakken we af tot:

- Een “tegel” mag van A naar B bewegen mits A en B aangrenzend zijn ($\Rightarrow h_2$)
- Een “tegel” mag van A naar B bewegen mits B leeg is
- Een “tegel” mag (altijd) van A naar B bewegen ($\Rightarrow h_1$)

Voor het dames-op-schaakbord probleem: een heuristiek h is het aantal ruziënde paren. Een actie is het verplaatsen van een dame in haar eigen kolom: op het n bij n bord $n(n - 1)$ opvolgers.

Dit suggereert de volgende aanpak. Een **hill-climber** kiest random uit de beste opvolgers. Dit is een **greedy = gretige** strategie — met locale minima!



Als het pad naar de oplossing er niet toe doet (zoals bij het dames-probleem), kunnen we **local search** technieken gebruiken. Je verbetert dan in elke stap één of meer huidige oplossingen. Bijkomend voordeel: je hebt altijd een complete (redelijke) oplossing.

Enkele problemen hiermee:

- locale maxima
- plateau's
- “heuvelruggen” (= ridges)

We zullen later apart **Genetische algoritmen** bestuderen.

We combineren de hill-climber (= gradient descent) met random bewegingen als volgt tot **Simulated Annealing**:

```
for tijd ← 1 to ∞ do
  Temperatuur ← schedule[tijd]
  if Temperatuur = 0 then return huidig
  volgende ← random opvolger van huidig
   $\Delta E$  ← waarde[volgende] – waarde[huidig]
  if  $\Delta E > 0$  then huidig ← volgende
  else huidig ← volgende met kans  $e^{\Delta E / \textit{Temperatuur}}$ 
```

Het algoritme is geïnspireerd op natuurkundige ideeën en het **Metropolis-algoritme** uit 1953.

Er zijn nog allerlei geavanceerde zaken:

- **pattern databases** om exacte oplossingen van deelproblemen in op te slaan;
- **(local) beam search** houdt een k -tal beste oplossingen bij, die elkaar “stimuleren”;
- en nog meer: Hoofdstuk 4.2, 4.3, 4.4 en 4.5 van [RN] — niet in dit college.

Het huiswerk voor de volgende keer (dinsdag 15 maart 2010): lees **Hoofdstuk 5**, p. 161–201 van [RN] eens door. Kijk ook eens naar de vragen bij dit hoofdstuk.

Denk tevens aan de tweede opgave: **Robocode!** En de vragenuren (maandagen, 13.45 uur, kamer 156a).

Maak de opgaven 1, 5, 22 en 37 van de “compilatie van oude tentamens” van de website. Dit zijn oude tentamenopgaven. Deze komen ook aan de orde bij het werkcollege van maandag 21 maart 2011.