CITS4211 Mid-semester test 2011

Fifty minutes, answer all four questions, total marks 60

Question 1. (12 marks) Briefly describe the principles, operation, and performance issues of *iterative deepening*.

Illustrate your answer using the tree in Figure 1. Include enough detail to make it clear that you understand how the algorithm works, particularly which nodes are expanded, in what order, and why.

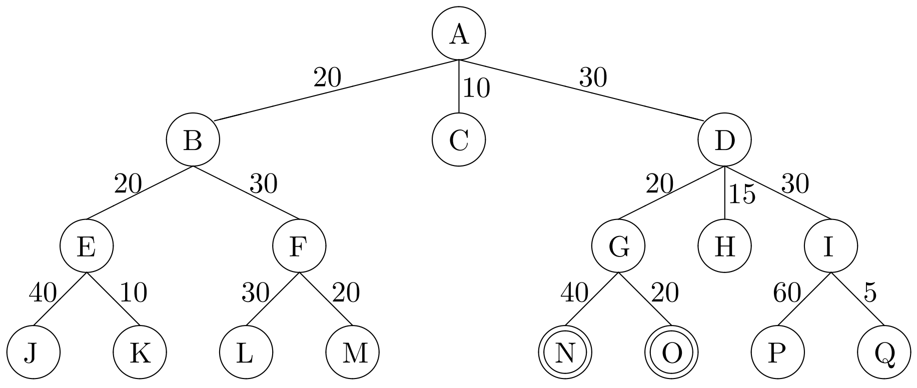

Figure 1: The search tree for Questions 1 and 2. Each arc is labeled with its cost. N and O are the only goal states.

Question 2. (16 marks) Briefly describe the principles, operation, and performance issues of *A**.

Illustrate your answer using the tree in Figure 1 and the heuristic function in Figure 2. Include enough detail to make it clear that you understand how the algorithm works, particularly which nodes are expanded, in what order, and why.

A	45	D	30	G	14	J	8	M	19	P	13
B	45	E	35	H	10	K	11	N	0	Q	13
C	52	F	30	I	25	L	2	O	0		

Figure 2: An admissible heuristic function for the tree in Figure 1.

Question 3. (12 marks) Briefly describe the principles, operation, and performance issues of *$\alpha - \beta$ pruning.*

Illustrate your answer using the tree in Figure 3. Include enough detail to make it clear that you understand how the algorithm works, particularly how the labels of the non-terminals evolve and why, and which terminals are ignored and why.

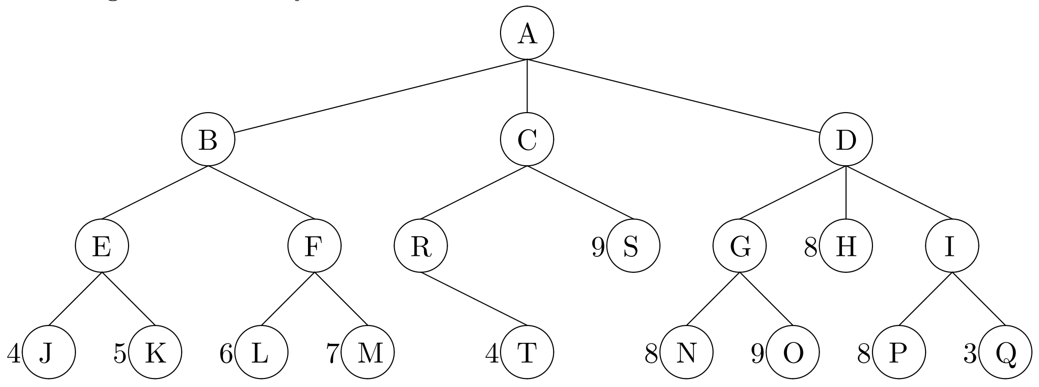

Figure 3: The game tree for Question 3. Each terminal node is labeled with its evaluation: higher values represent better positions for the first player.

Question 4. (20 marks) Briefly describe the principles, operation, and performance issues of *policy iteration.*

Illustrate your answer using the problem in Figure 4. Include enough detail to make it clear that you understand how the algorithm works, particularly how utilities are calculated, and how new policies are derived and justified. (Focus on doing one iteration correctly and thoroughly.)

-16	T 160
S -16	-27.2

$\downarrow$	$-$
$\uparrow$	$\uparrow$

Figure 4: The path-planning problem for Question 4, and the initial policy. S is the start state, T is the only terminal state, and each state is labeled with its reward. Every attempted move has an 80% chance of working, a 10% chance of veering left 90°, and a 10% chance of veering right 90°. Walking into a wall means the agent stays put.

1. *Iterative deepening* is a sequence of depth-first, depth-limited searches where the limit starts at 1 and increases by 1 each time. It stops as soon as a goal state is found.

ID has the linear space behaviour of depth-first search, but because of the depth-limit it is complete, and because Node X always precedes Node Y if X is shallower than Y, it is also optimal (it always finds the shallowest goal state). The principal drawback is that some nodes are examined multiple times.

In the example the nodes are expanded in this order:

A; A,B,C,D; A,B,E,F,C,D,G,H,I; A,B,E,J,K,F,L,M,C,D,G,N

2. *A** maintains a pool of nodes, which starts with just the root. The cost-estimate of each node X is the sum of the cost from the root to X and the heuristic estimate of the cost from X to the goal state. The next node expanded is one which has the lowest cost-estimate. It stops as soon as a goal state is found.

*A** is provably optimal (it always finds the cheapest goal state) as long as its heuristic is admissible, i.e. it never over-estimates the cost from X to the goal state. Variants of *A** (SMA*, IDA*) have been described which also have good space behaviour.

In the example the pool of nodes evolves like this:

A	$0 + 45 = 45$

D	$30 + 30 = 60$
C	$10 + 52 = 62$
B	$20 + 45 = 65$

H	$45 + 10 = 55 \rightarrow 60$
C	$10 + 52 = 62$
G	$50 + 14 = 64$
B	$20 + 45 = 65$
I	$60 + 25 = 85$

B	20 + 45 = 65
O	70 + 0 = 70
I	60 + 25 = 85
N	90 + 0 = 90

O	70 + 0 = 70
E	40 + 35 = 75
F	50 + 30 = 80
I	60 + 25 = 85
N	90 + 0 = 90

3. $\alpha-\beta$ *pruning* implements *minimax*, but it ignores (prunes) nodes when it has enough information to infer that inspecting the node would not change the value assigned to one of the node's ancestors.

$\alpha-\beta$ obviously in general inspects fewer nodes than *minimax*: the saving will be bigger if it inspects relatively poor nodes as early as possible in the process.

In the example the labels of the non-terminals evolve like this if evaluation is left-to-right:

- $J = 4 \rightarrow E \geq 4$
- $K = 5 \rightarrow E = 5 \rightarrow B \leq 5$
- $L = 6 \rightarrow F \geq 6 \rightarrow B = 5 \rightarrow M$ is irrelevant and $A \geq 5$
- $T = 4 \rightarrow R = 4 \rightarrow C \leq 4 \rightarrow S$ is irrelevant
- $N = 8 \rightarrow G \geq 8$
- $O = 9 \rightarrow G = 9 \rightarrow D \leq 9$
- $H = 8 \rightarrow D \leq 8$
- $P = 8 \rightarrow I \geq 8 \rightarrow D = 8 \rightarrow Q$ is irrelevant and $A = 8$

and like this if it is right-to-left:

- $Q = 3 \rightarrow I \geq 3$
- $P = 8 \rightarrow I = 8 \rightarrow D \leq 8$
- $H = 8$ changes nothing
- $O = 9 \rightarrow G \geq 9 \rightarrow D = 8 \rightarrow N$ is irrelevant and $A \geq 8$
- $S = 9 \rightarrow C \leq 9$
- $T = 4 \rightarrow R = 4 \rightarrow C = 4$
- $M = 7 \rightarrow F \geq 7$
- $L = 6 \rightarrow F = 7 \rightarrow B \leq 7 \rightarrow A = 8 \rightarrow E, J, K$ are irrelevant

4. *Policy iteration* starts with an arbitrary policy; in each iteration it calculates the value of each non-terminal state under the current policy, then the optimal move in each non-terminal given those values. If the policy needs to be updated, it iterates.

Policy iteration is generally faster than the alternative *Value iteration*, because it reaches equilibrium without calculating the precise value of each state.

In the example the first iteration of the algorithm goes like this (x is the top-left state, y is bottom-left, z is bottom-right):

$$\begin{aligned} x &= -16 + 0.8y + 0.1x + 16 \\ y &= -16 + 0.8x + 0.1y + 0.1z \\ z &= -27.2 + 128 + 0.1y + 0.1z \end{aligned}$$

Simplify and multiply by 10 throughout.

$$\begin{aligned} 9x &= 8y \\ 9y &= -160 + 8x + z \\ 9z &= 1008 + y \end{aligned}$$

Substitute for y and simplify.

$$\begin{aligned} 17x/8 &= -160 + z \\ z &= 112 + x/8 \end{aligned}$$

Substitute for z, and solve for x, then z, then y.

$$x = -24, z = 109, y = -27$$

Top-left: only right is positive.

$$\begin{aligned} up: 0.9(-24) + 16 &< 0 \\ down: 0.1(-24) + 16 + 0.8(-27) &< 0 \\ right: 0.1(-24) + 128 + 0.1(-27) &> 0 \\ left: 0.9(-24) + 0.1(-27) &< 0 \end{aligned}$$

Bottom-left: only right is positive.

$$\begin{aligned} up: 0.8(-24) + 0.1(-27) + 0.1(109) &< 0 \\ down: 0.9(-27) + 0.1(109) &< 0 \\ right: 0.1(-24) + 0.1(-27) + 0.8(109) &> 0 \\ left: 0.9(-27) + 0.1(-24) &< 0 \end{aligned}$$

Bottom-right: up is clearly the biggest.

$$\begin{aligned} up: 128 + 0.1(-27) + 0.1(109) &\approx 136 \\ down: 0.9(109) + 0.1(-27) &\approx 95 \\ right: 16 + 0.9(109) &\approx 114 \\ left: 16 + 0.8(-27) + 0.1(109) &\approx 5 \end{aligned}$$

$\rightarrow$	$-$
$\rightarrow$	$\uparrow$

Figure 5: Policy after one iteration.