CS4725/CS6705 - Fall 2017
Midterm
Wednesday, October 25, 2017

Name: SAMPLE SOLUTIONS

Student ID: _____

- The midterm is out of 30 marks.
- You will have 50 minutes to write the midterm. Manage your time wisely. If you are unsure about a question, move on to the rest of the midterm and come back to any problematic questions later.

1.	/ 3
2.	/ 6
3.	/ 6
4.	/ 5
5.	/ 4
6.	/ 6
Total	/ 30

---

1. **Uninformed search (3 marks)**

   (a) (1.5 marks) Describe one **advantage** of depth-first search, when compared to breadth-first search.

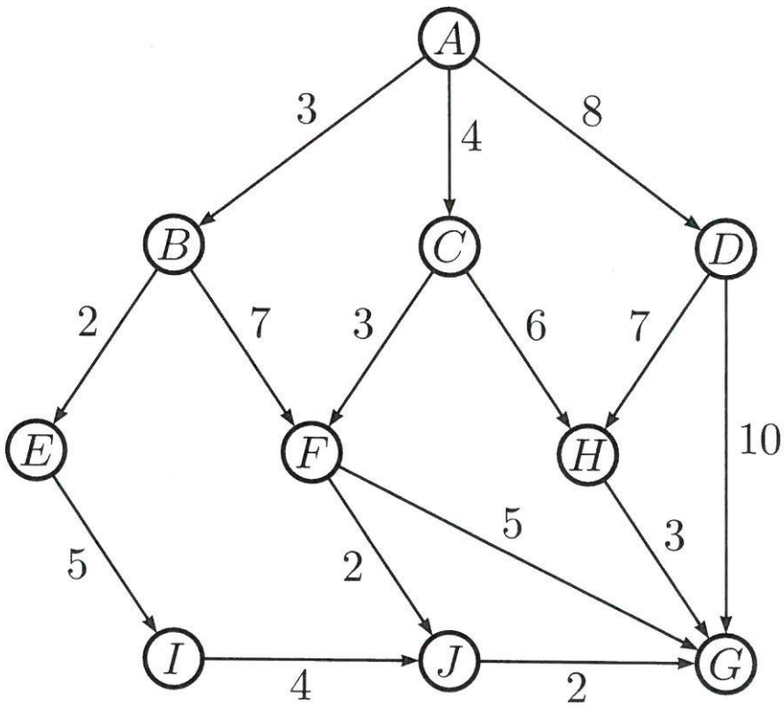
   DFS has lower space complexity than BFS.

   (b) (1.5 marks) Describe one **disadvantage** of depth-first search, when compared to breadth-first search.

   DFS is not guaranteed to be complete, while BFS is (as long as the branching factor is finite).

2. **Uninformed search (6 marks)**

Consider the state space shown below. A is the start state and G is the goal state. Arrows represent possible actions, and the numbers on the arrows represent action costs. Note that the arrows are directed; for example, $A \rightarrow B$ is a possible action, but $B \rightarrow A$ is not.



For each of the following search algorithms, what is the solution path that would be returned and what is its cost?

Assume that the depth-first and breadth-first searches will explore the children of a given node in **alphabetical order**.

(a) **Depth-First Search**

Path returned: A → B → E → I → J → G

Cost: 16

(b) **Breadth-First Search**

Path returned: A → D → G

Cost: 18

(c) **Uniform-Cost Search**

Path returned: A → C → F → J → G

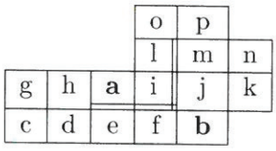
Cost: 11

3. **Informed search (6 marks)**

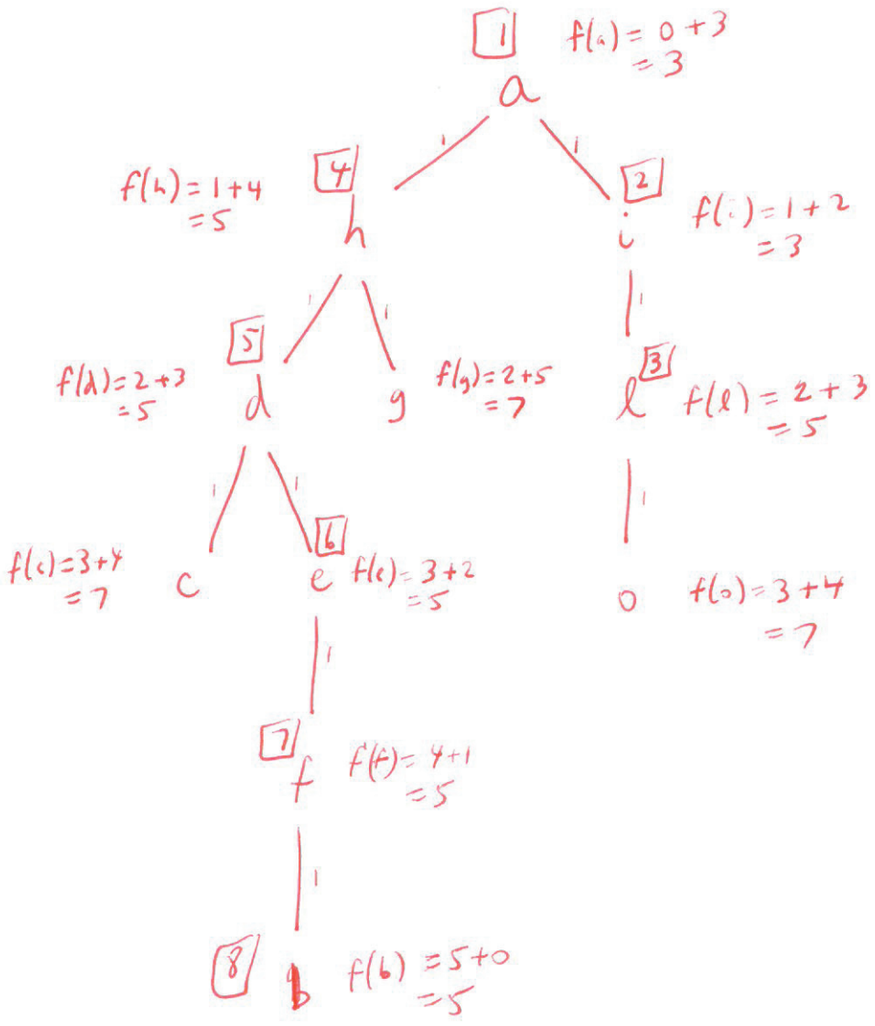
Consider the simple maze shown below, where the successors of a cell are the cells directly to the east, west, north and south of the cell, except that you are not allowed to pass through the thick wall indicated by the double line. For example, the successors of cell m are $\{j, n, p\}$ (and not cell l).

Trace the operation of A* search applied to the problem of getting from cell a to cell b. The heuristic is the Manhattan distance heuristic, but assuming that the thick wall is not there. The heuristic values for each node are summarized for you in the table to the right.

- Draw the search tree, starting from cell a. Beside each node N in your tree, show the calculation of $f(N)$. Assume that the cost of each move is 1.
- When you expand a node, label it with a number indicating the step at which it was expanded. (In other words, put a [1] beside the root, a [2] beside the second node you expand, *etc.*)
- You can avoid moves that involve going back to the cell that you just came from. For example, you should not consider going north from cell c to cell g if you had just reached cell c by going south from cell g.
- If two or more nodes are tied for the lowest f value, choose the one with minimum h value. Use alphabetical order if there is still a tie.



cell	h(cell)	cell	h(cell)
a	3	i	2
b	0	j	1
c	4	k	2
d	3	l	3
e	2	m	2
f	1	n	3
g	5	o	4
h	4	p	3



4. **Short-answer questions (5 marks)**

(a) (2 marks) Which of the following statements is true? Insert a check mark in the appropriate box.

- ☐ All consistent heuristics are admissible, and all admissible heuristics are consistent.
- ☑ All consistent heuristics are admissible, but it is possible for an admissible heuristic to be inconsistent.
- ☐ All admissible heuristics are consistent, but it is possible for a consistent heuristic to be inadmissible.
- ☐ It is possible for an admissible heuristic to be inconsistent, and it is possible for a consistent heuristic to be inadmissible.

(b) (3 marks) Consider a best-first search technique that uses the following objective function, where w is an adjustable weight value:

$$f(n) = w \cdot g(n) \; + \; (2 - w) \cdot h(n)$$

i. When $w = 1$, this search technique will behave exactly like which search algorithm we have studied in class?

- ☐ breadth-first search
- ☐ depth-first search
- ☐ uniform cost search
- ☐ greedy best-first search
- ☑ A* search

ii. When $w = 2$, this search technique will behave exactly like which search algorithm we have studied in class?

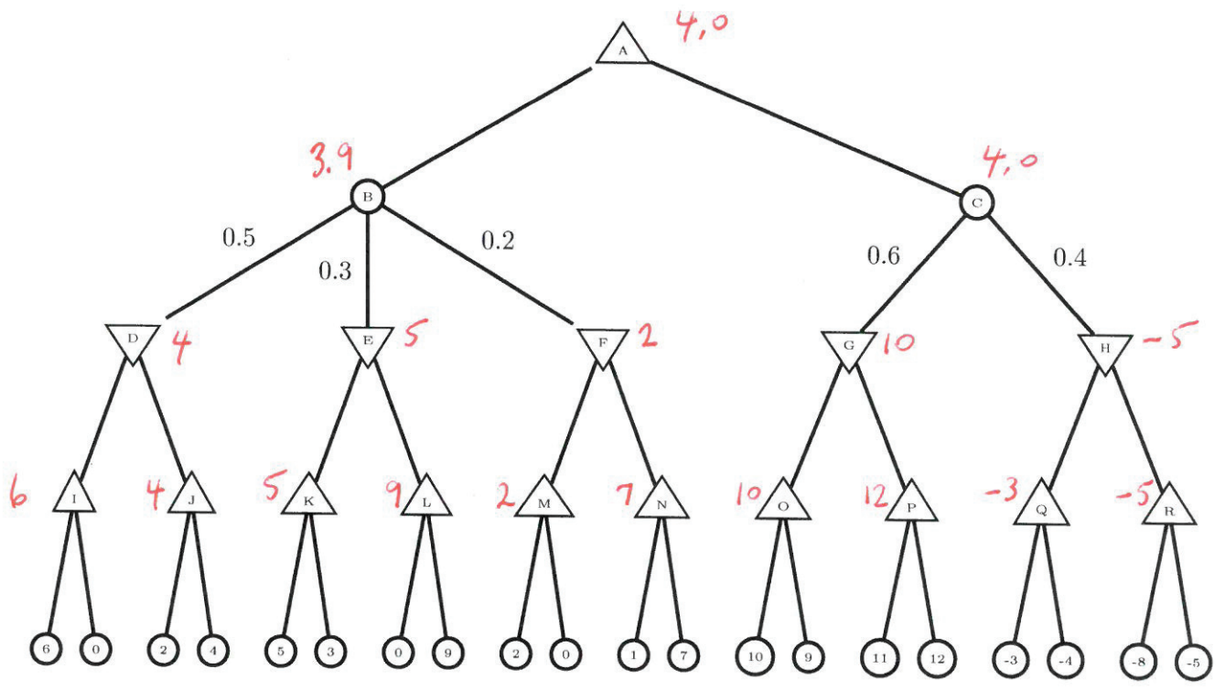
- ☐ breadth-first search
- ☐ depth-first search
- ☑ uniform cost search
- ☐ greedy best-first search
- ☐ A* search

iii. When $w = 0$, this search technique will behave exactly like which search algorithm we have studied in class?

- ☐ breadth-first search
- ☐ depth-first search
- ☐ uniform cost search
- ☑ greedy best-first search
- ☐ A* search

5. **Adversarial Search (4 marks)**

Consider the game tree below, representing a game in which there is an element of chance involved. Nodes A and I-R are MAX nodes, and nodes D-H are MIN nodes. Nodes B and C are chance nodes, with the probabilities of different chance events indicated on the edges leaving the nodes.



Write the expectiminimax value of each node beside the node. Show your calculations for the chance nodes here:

value(B) = 0.5(4) + 0.3(5) + 0.2(2)
= 2.0 + 1.5 + 0.4
= 3.9

value(C) = 0.6(10) + 0.4(-5)
= 6.0 − 2.0
= 4.0

What is the best move for MAX at node A: to go to node B or C?

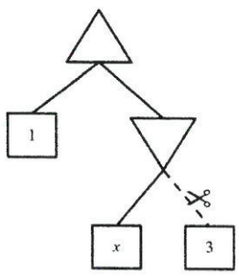
C is the better move: expected value of 4.0 vs. 3.9

6. **Alpha-beta pruning (6 marks)**

[Credit to CS188 instructors at UC Berkeley]

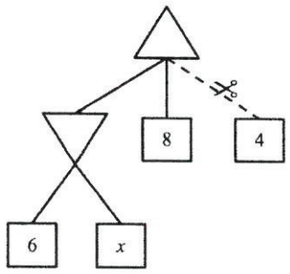
For each of the game trees shown below, state for which range of values of x the dashed branch with the scissors would be pruned. If the pruning will not happen for any value of x, write "none". If pruning will happen for all values of x, write "all".

Example Tree



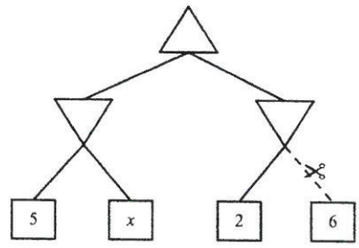
Answer: $\mathbf{x \leq 1}$

(a)



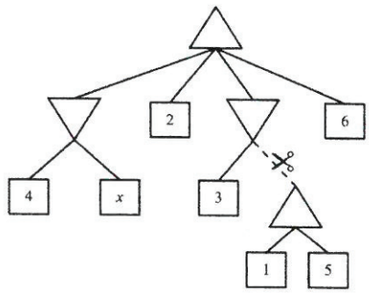
Answer: none

(b)



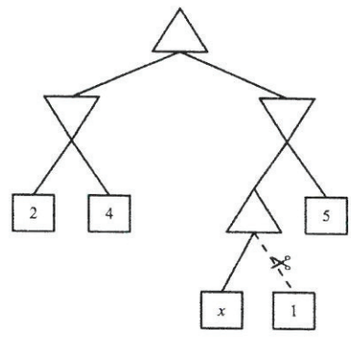
Answer: $x \geq 2$

(c)



Answer: $x \geq 3$

(d)



Answer: none

If it is helpful, the relevant pseudocode algorithms from class are presented below.

```
function MAX-VALUE(state, α, β) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← -∞
    for each action in ACTIONS(state) do
        v ← MAX(v, MIN-VALUE(RESULT(state,action), α, β))
        if v ≥ β then return v        // ignore/prune all other actions
        α ← MAX(α, v)
return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← +∞
    for each action in ACTIONS(state) do
        v ← MIN(v, MAX-VALUE(RESULT(state,action), α, β))
        if v ≤ α then return v        // ignore/prune all other actions
        β ← MIN(β, v)
return v
```