Part 1. Search

1. (15 pts) We call a search algorithm a *graph search* if it checks for redundant paths and a *tree-like search* if it does not.

   Best-First Search employs a general *priority queue* data structure to store the frontier, and we can use this algorithm to implement various search strategies—e.g., Breadth-First Search, Depth-First Search, Uniform-Cost Search, etc.—by specifying the type of queue used to store the frontier (or "fringe") nodes of the search.

   Recall, the three basic types of queues: (1) *first-in-first-out* (FIFO) *queue* whose pop method returns the node that was added to the queue first; (2) *last-in-first-out* (LIFO) *queue* (a *stack*), where pop returns the node that was most recently added; (3) *priority queue*, where pop returns the node n with $f(n) = \min_m f(m)$, for some evaluation function f.

   Select the answer that correctly completes the following sentences.

   (a) Depth-First Search uses
   - ☐ a priority queue for some evaluation function, f.
   - ☐ a FIFO queue.
   - √ ***a LIFO queue.***
   - ☐ None of these.

   (b) Breadth-First Search uses
   - ☐ a priority queue for some evaluation function, f.
   - √ ***a FIFO queue.***
   - ☐ a LIFO queue.
   - ☐ None of these.

   (c) Uniform-Cost Search uses
   - √ ***a priority queue for some evaluation function, f.***
   - ☐ a FIFO queue
   - ☐ a LIFO queue
   - ☐ None of these.

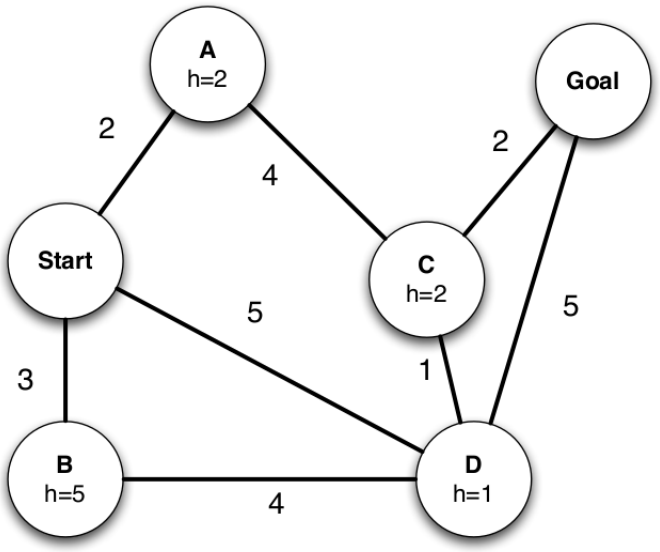
   (d) Suppose the search space is *finite*. Which of the following search strategies is complete? (Select all that apply.)
   - ☐ Breadth-First Search
   - ☐ Depth-First Search
   - ☐ Iterative Deepening Search
   - √ ***All of the above.***
   - ☐ None of the above.

   Explanation. If the search space is finite, then each of these methods will eventually find a solution.

   (e) In general, Depth-First Search is the preferred uninformed search method when the search state space is *infinite*. ☐ True √ **False**

   Explanation. Depth-First Search is not complete when the search space is infinite, so usually Iterative Deepening would be preferred in that case.

2. (16 pts) Consider the graph below. Arcs are labeled with their weights.



For each graph search strategy below, work out the order in which the states are expanded and the path returned by graph search. Values of an heuristic function h are shown, though you may not need them to answer some of the question below. Remember that in graph search, a state is expanded only once. Assume that ties are broken alphabetically (so a partial plan S→X→A would be expanded before S→X→B and S→A→Z would be expanded before S→B→A). You may find it helpful to execute your searches on the blank pages provided at the end of the exam.

(a) In what order are states expanded by Breadth-first Search?

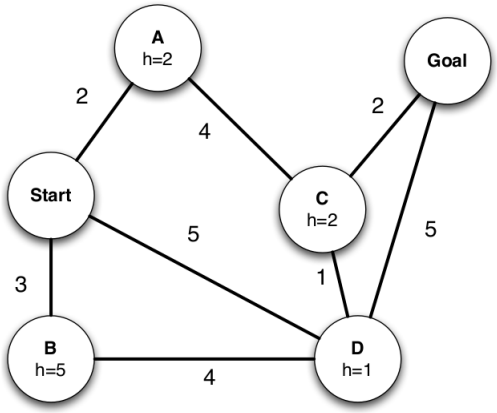
- □ Start, A, B, C, D, Goal
- √ ***Start, A, B, D, C, Goal***
- □ Start, A, C, D, B, Goal
- □ Start, A, D, C, Goal
- □ Start, D, Goal

(b) What path does Breadth-first Search return?

- □ Start-A-C-Goal
- □ Start-A-C-D-Goal
- □ Start-B-D-Goal
- □ Start-B-D-C-Goal
- √ ***Start-D-Goal***

(c) In what order are states expanded by Uniform-cost Search?

- □ Start, A, B, C, D, Goal
- √ ***Start, A, B, D, C, Goal***
- □ Start, A, C, D, B, Goal
- □ Start, A, D, C, Goal
- □ Start, D, Goal



(d) What path does Uniform-cost Search return?

- √ ***Start-A-C-Goal***
- □ Start-A-C-D-Goal
- □ Start-B-D-Goal
- □ Start-B-D-C-Goal
- □ Start-D-Goal

(e) In what order are states expanded by Greedy Search?

- □ Start, A, B, C, D, Goal
- □ Start, A, B, D, C, Goal
- □ Start, A, C, D, B, Goal
- □ Start, A, D, C, Goal
- √ ***Start, D, Goal***

(f) What path does Greedy Search return?

- □ Start-A-C-Goal
- □ Start-A-C-D-Goal
- □ Start-B-D-Goal
- □ Start-B-D-C-Goal
- √ ***Start-D-Goal***

(g) In what order are states expanded by A∗ Search?

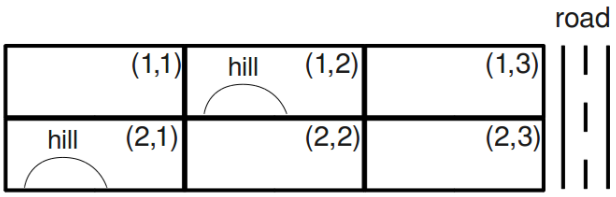
- □ Start, A, B, C, D, Goal
- □ Start, A, B, D, C, Goal
- □ Start, A, C, D, B, Goal
- √ ***Start, A, D, C, Goal***
- □ Start, D, Goal

(h) What path does A∗ Search return?

- √ ***Start-A-C-Goal***
- □ Start-A-C-D-Goal
- □ Start-B-D-Goal
- □ Start-B-D-C-Goal
- □ Start-D-Goal

Part 2. CSP

3. (15 pts) (Campus Layout) You are asked to determine the layout of a new, small college. The campus will have four structures: an administration structure (A), a bus stop (B), a classroom (C), and a dormitory (D). Each structure (including the bus stop) must be placed somewhere on the grid shown below.



The layout must satisfy the following constraints:

(i) The bus stop (B) must be adjacent to the road.

(ii) The administration structure (A) and the classroom (C) must both be adjacent to the bus stop (B).

(iii) The classroom (C) must be adjacent to the dormitory (D).

(iv) The administration structure (A) must not be adjacent to the dormitory (D).

(v) The administration structure (A) must not be on a hill.

(vi) The dormitory (D) must be on a hill or adjacent to the road.

(vii) All structures must be in different grid squares.

Here, adjacent means that the structures must share a grid edge, not just a corner. We recommend you work out the solutions to the following questions on a sheet of scratch paper, and then enter your results below.

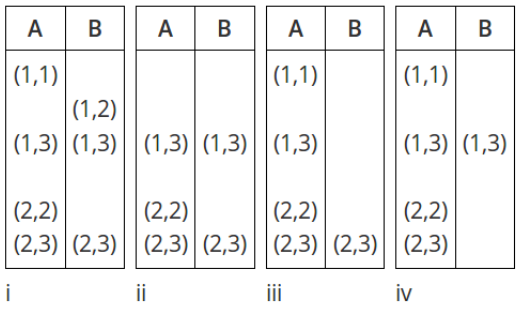
(a) Which of the constraints above are unary constraints?
√ ***(i)*** □ (ii) □ (iii) □ (iv) √ ***(v)*** √ ***(vi)*** □ (vii) □ None of these

(b) Select the domains of all variables after unary constraints have been applied.
A: √ ***(1,1)*** □ (1,2) √ ***(1,3)*** □ (2,1) √ ***(2,2)*** √ ***(2,3)***
B: □ (1,1) □ (1,2) √ ***(1,3)*** □ (2,1) □ (2,2) √ ***(2,3)***
C: √ ***(1,1)*** √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** √ ***(2,2)*** √ ***(2,3)***
D: □ (1,1) √ ***(1,2)*** √ ***(1,3)*** √ ***(2,1)*** □ (2,2) √ ***(2,3)***

(c) Let's start from the table above (the answer to Part b) and enforce arc consistency. Initally, the queue contains all arcs (in alphabetical order). Let's examine what happens when enforcing A → B. After enforcing unary constraints, the domains of A and B are:

A	B
(1,1)	
(1,3)	(1,3)
(2,2)	
(2,3)	(2,3)

Which of the following contains the correct domains after enforcing A → B? Pay attention to which variable's domain changes and which side of the arc it's on.



□ i ✓ ***ii*** □ iii □ iv

(d) Starting from the answer to Part b (in which unary constraints are enforced), select the domains of all variables after A → B is enforced.

A: □ (1,1) □ (1,2) ✓ ***(1,3)*** □ (2,1) ✓ ***(2,2)*** ✓ ***(2,3)***
B: □ (1,1) □ (1,2) ✓ ***(1,3)*** □ (2,1) □ (2,2) ✓ ***(2,3)***
C: ✓ ***(1,1)*** ✓ ***(1,2)*** ✓ ***(1,3)*** ✓ ***(2,1)*** ✓ ***(2,2)*** ✓ ***(2,3)***
D: □ (1,1) ✓ ***(1,2)*** ✓ ***(1,3)*** ✓ ***(2,1)*** □ (2,2) ✓ ***(2,3)***

(e) After enforcing these arcs, the next is C → B. Continuing from the previous parts, select the domains of all variables after C → B is enforced.

A: □ (1,1) □ (1,2) ✓ ***(1,3)*** □ (2,1) ✓ ***(2,2)*** ✓ ***(2,3)***
B: □ (1,1) □ (1,2) ✓ ***(1,3)*** □ (2,1) □ (2,2) ✓ ***(2,3)***
C: □ (1,1) ✓ ***(1,2)*** ✓ ***(1,3)*** □ (2,1) ✓ ***(2,2)*** ✓ ***(2,3)***
D: □ (1,1) ✓ ***(1,2)*** ✓ ***(1,3)*** ✓ ***(2,1)*** □ (2,2) ✓ ***(2,3)***

Explanation.

(a) Unary constraints are constraints that involve a single variable. (i) only constrains (B), (v) only constrains (A), and (vi) only constrains (D). All the other constraints involve more than one variable.

(b) (i) limits (B) to (1,3) or (2,3), because those are the only positions adjacent to the road.
(v) removes (1,2) and (2,1) from the domain of (A) because those two positions have hills.
(vi) removes (1,1) and (2,2) from the domain of (D) because those two positions are neither on a hill, nor adjacent to the road; (C) has no unary constraints, and thus can still be any value after unary constraints are applied.

(c) After an arc is enforced, every value in the domain of the tail has at least one possible value in the domain of the head such that the two values satisfy all constraints with each other. If a value in the tail's domain does not satisfy this, that value can be removed, because it is guaranteed not to be a part of a solution without backtracking on some previously selected variable. In this case, (1,1) in A's domain is not adjacent to either (1,3) or (2,3), so B has no possible values and (1,1) can be removed from A's domain.

(d) These are the same as part 2, except with (1,1) removed from A's domain due to enforcing A → B.

(e) (1,1) and (2,1) are removed from C, because if either value is assigned to C, there is no possible value in B's domain that satisfies all of the constraints. A and D's domains do not change.

Part 3. Markov Decision Processes

4. (18 pts) In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. If your total score is 6 or higher, the game ends, and you receive a utility of 0. When you Stop, your utility is equal to your total score (up to 5), and the game ends. When you Draw, you receive no utility. There is no discount ($\gamma = 1$). Let's formulate this problem as an MDP with the following states: 0, 2, 3, 4, 5, and a *Done* state, for when the game ends.

   (a) Fill in the correct values (probabilities) of the transition function for this MDP.

   T(s, Stop, Done) = ***1***

   T(0, Draw, s') = ***1/3*** for s' $\in \{2, 3, 4\}$

   T(2, Draw, s') = ***1/3*** for s' $\in$ {4, 5, Done}

   T(3, Draw, 5) = ***1/3*** T(3, Draw, Done) = ***2/3***

   T(4, Draw, Done) = ***1*** T(5, Draw, Done) = ***1***

   T(s, a, s') = ***0*** for all other values of s, a, and s'.

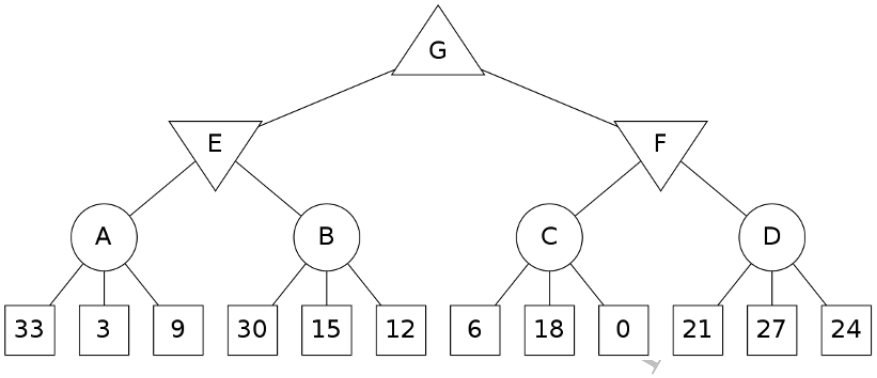
   (b) Fill in the correct values of the reward function for this MDP.

   R(s, Stop, Done) = ***s*** for s ≤ 5 R(s, a, s') = 0 for s > 5

   (c) Fill in the remaining values in the table below by performing policy iteration for one step of this MDP, starting from the fixed policy shown in the π_i row of the table.

States	0	2	3	4	5
π_i	Draw	Stop	Draw	Stop	Draw
V^{π_i}	2	2	0	4	0
π_{i+1}	Draw	Stop	Stop	Stop	Stop

5. (18 pts) Consider the game tree shown below. Triangles that point up, such as the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. The circular nodes represent chance nodes in which each of the possible actions may be taken with equal probability. The square nodes at the bottom represent leaf nodes.



(a) Assuming both players act optimally, carry out the expectiminimax search algorithm. Enter the values for the letter nodes in the boxes below the tree.

A: ***15*** B: ***19*** C: ***8*** D: ***24***

E: ***15*** F: ***8*** G: ***15***

Explanation. The value for each circular node is equal to the expectation of the values of its children, which is found by adding up each of the child values and dividing by the number of children.

E and F take the minimums from their respective children.

G takes 15, the max of E and F

(b) Suppose your goal is simply to compute the value of G and the optimal action (left or right) that results in that value.

i. Assuming you carry out expectimax starting from the left-hand side and working your way towards the right of the tree, which part of the tree (if any) could you "prune" (avoid checking) while still obtaining the correct (optimal) expectimax value?

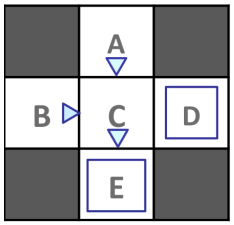
Answer. D and its leaves (21, 27, 24).

ii. What is the optimal action (left or right) using expectimax?

Answer. left (towards E)

Part 4. Model-based and Model-free Reinforcement Learning

6. (18 pts) Shown below are some observed outcomes of a MDP.



Observed Episodes (Training)

Episode 1
A, south, C, -1
C, south, E, -1
E, exit, x, +10

Episode 2
B, east, C, -1
C, south, D, -1
D, exit, x, -10

Episode 3
B, east, C, -1
C, south, E, -1
E, exit, x, +10

Episode 4
A, south, C, -1
C, south, E, -1
E, exit, x, +10

(a) What model would be learned from the observations shown above?
(Check the box to the left of the correct answer.)

i. T(A, south, C) = ☐ 0 ☐ 1/4 ☐ 1/2 ☐ 3/4 √ ***1***

ii. T(B, east, C) = ☐ 0 ☐ 1/4 ☐ 1/2 ☐ 3/4 √ ***1***

iii. T(C, south, E) = ☐ 0 ☐ 1/4 ☐ 1/2 √ ***3/4*** ☐ 1

iv. T(C, south, D) = ☐ 0 √ ***1/4*** ☐ 1/2 ☐ 3/4 ☐ 1

Explanation.

i. The action south is taken twice from state A, and both times results in state C. $2/2 = 1$.
ii. The action east is taken twice from state B, and both times results in state C. $2/2 = 1$.
iii. The action south is taken four times from state C, and results in state E three times. $3/4 = 0.75$
iv. The action south is taken four times from state C, and results in state D one time. $1/4 = 0.25$.

(b) What are the estimates for the following quantities as obtained by **direct evaluation**?
(Check the box to the left of the correct answer.)

Example. $\hat{V}^{\pi}(\text{A})$ = ☐ -10 ☐ -4 ☐ -2 ☐ 0 ☐ 2 ☐ 4 ☑ 8 ☐ 10

(Recall, direct evaluation computes $\hat{V}^{\pi}(s)$ as the average achieved starting from s.)

i. $\hat{V}^{\pi}(\text{A})$ = ☐ -10 ☐ -4 ☐ -2 ☐ 0 ☐ 2 ☐ 4 √ ***8*** ☐ 10

ii. $\hat{V}^{\pi}(\text{B})$ = ☐ -10 ☐ -4 √ ***-2*** ☐ 0 ☐ 2 ☐ 4 ☐ 8 ☐ 10

iii. $\hat{V}^{\pi}(\text{C})$ = ☐ -10 ☐ -4 ☐ -2 ☐ 0 ☐ 2 √ ***4*** ☐ 8 ☐ 10

iv. $\hat{V}^{\pi}(\text{D})$ = √ ***-10*** ☐ -4 ☐ -2 ☐ 0 ☐ 2 ☐ 4 ☐ 8 ☐ 10

v. $\hat{V}^{\pi}(\text{E})$ = ☐ -10 ☐ -4 ☐ -2 ☐ 0 ☐ 2 ☐ 4 ☐ 8 √ ***10***

Explanation.

The estimated value of $\hat{V}^{\pi}(s)$ is equal to the average value achieved starting from that state.

$\hat{V}^{\pi}$(A): Episodes 1 and 4 start from state A; both result in a utility of 8, yielding (8+8)/2 = 8.

$\hat{V}^{\pi}$(B): Episodes 2 and 3 start from B; they result in -12 and 8, respectively, yielding (8-12)/2 = -2.

$\hat{V}^{\pi}$(C): State C is visited in every episode. The remaining rewards from C in episodes 1, 3, and 4 total 9, while those in episode 2 total -11, yielding (9+9+9-11)/4 = 4.

$\hat{V}^{\pi}$(D): State D is only visited in episode 2 and has remaining utility -10.

$\hat{V}^{\pi}$(E): State E is visited in episodes 1, 3, and 4 and has remaining utility 10 in each state, yielding (10 + 10 + 10)/3 = 10.