15-381 Artificial Intelligence: Representation and Problem Solving, Fall 2009

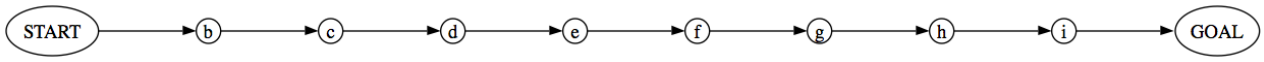
MIDTERM EXAM

1. Personal info:
   - Name:
   - Andrew account:
   - E-mail address:
2. There should be 18 numbered pages in this exam (including this cover sheet).
3. No extra material other than your 1 sheet of notes is allowed. Calculators are allowed.
4. If you need more room to work out your answer to a question, use the back of the page and clearly mark on the front of the page.
5. You have 80 minutes.
6. Good luck!

Question	Topic	Max. score	Score
1	Uninformed Search	10	
2	A*	15	
3	Minimax	15	
4	CSPs	15	
5	Planning	10	
6	Bayes Net	15	
TOTAL		80	

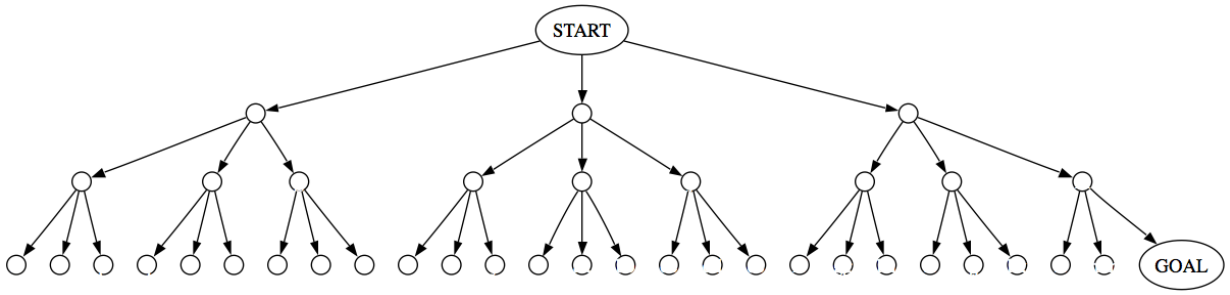
1 Uninformed Search [10 points]

1. [4 points] Would IDS perform better or worse than DFS on the graph below, in terms of running time? Explain why.



★ **SOLUTION:** IDS would perform worse, since DFS is optimal but IDS will restart every time it reaches a new node.

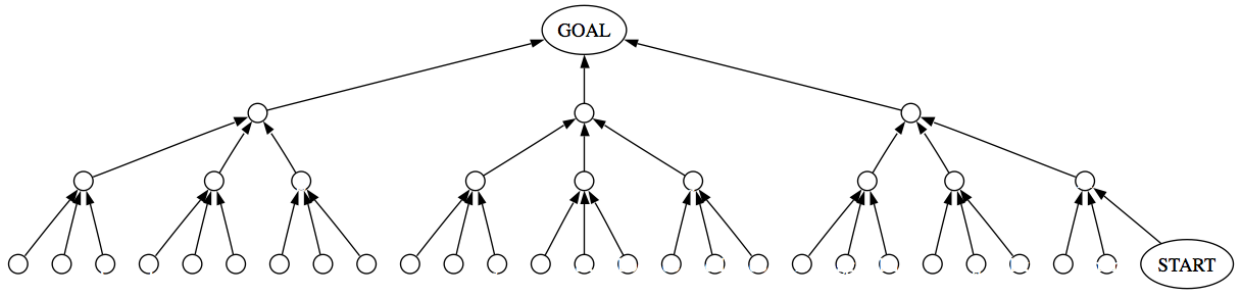
2. [3 points] Assuming back-pointers are given, which algorithm would perform better on the following graph in terms of running time: bi-directional BFS or BFS? Explain why.



★ **SOLUTION:** Bi-directional BFS would outperform BFS. BFS will have to expand every node to find the path to the goal, whereas using Bi-direction BFS, we will find it in time linear to the depth of the tree since the path from the goal to the start node is found in linear time by the backward search part of bi-directional BFS.

3. [3 points] Assuming back-pointers are given, which algorithm would perform better on the following graph in terms of running time: bi-directional BFS or BFS? Explain why.

★ **SOLUTION:** BFS will outperform bi-directional BFS. BFS will find the path to the goal in time steps equal to the depth of the tree. Bi-directional BFS will waste effort on the backward search portion, (at least) doubling the search time.



2 A* [15 points]

Consider an A* search from start node S to goal node G with the sequence of priority queue contents as shown below as nodes and corresponding f values, using an admissible heuristic h. All the edges have cost ≥ 1.

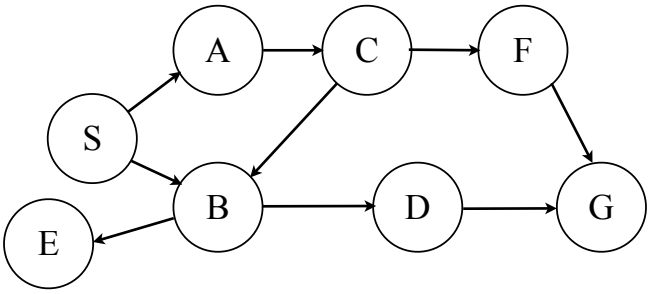
Step	f values		
1	S,8		
2	A,10	B,14	
3	C,4	B,14	
4	F,10	B,11	
5	G,14	B,11	
6	D,11	G,14	E,18
7	G,11	G,14	E,18
8	G,14	E,18	

1. [1 point] Did the A* search terminate? Why or why not?

   ★ **SOLUTION:** Yes. At time step 7, G, the goal node, has the smallest f value, so it was popped then and A* terminated.

2. [2 points] Draw the graph consistent with the A* search above with the fewest number of edges, specifying vertices and edges (but not the edge costs).

★ **SOLUTION:** :

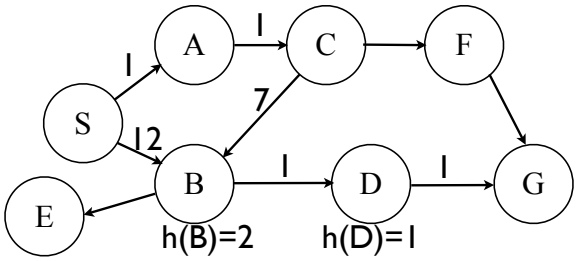


3. [2 point] What is the path found by A*? Is it optimal?

   ★ **SOLUTION:** The path is: $S \to A \to C \to B \to D \to G$. It is optimal since h is an admissible heuristic.

4. Based on the A* search, give the values for the following quantities or say that they are unknown. Additionally, you know that the cost of edge (D, G) is 1, the cost of edge (C, B) is 7, and that $h(B) = 2$.

   ★ **SOLUTION:** Here are the numbers we can infer from the information given.



   (a) [1 point] $h(S)$

   ★ **SOLUTION:** 8

   (b) [1 point] the number of children of A

   ★ **SOLUTION:** Unknown: 1 or 2

   (c) [1 point] the cost of the shortest path from S to G

★ **SOLUTION:** 11

(d) [1 point] $h(D)$

★ **SOLUTION:** 1

(e) [1 point] $g(D)$

★ **SOLUTION:** 10

(f) [1 point] the cost of the shortest path from S to B

★ **SOLUTION:** 9

(g) [1 point] $g(C)$

★ **SOLUTION:** 2

(h) [1 point] $g(F)$

★ **SOLUTION:** Unknown

(i) [1 point] the cost of the shortest path from S to A

★ **SOLUTION:** 1

(j) [1 point] $h(A)$

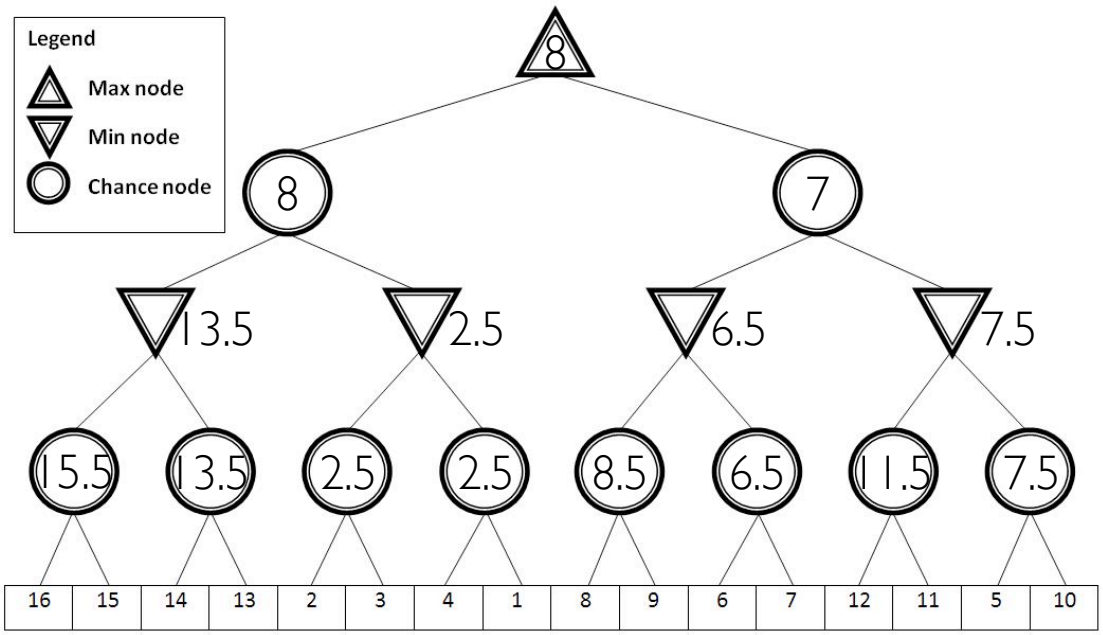
★ **SOLUTION:** 9

3 Minimax [15 points]

1. **Minimax with Chance.** Consider the game tree below. At every chance node, an unbiased coin is flipped to decide which way to proceed down the game tree. The numbers at the leaves of the tree are the utilities for each outcome of the game.

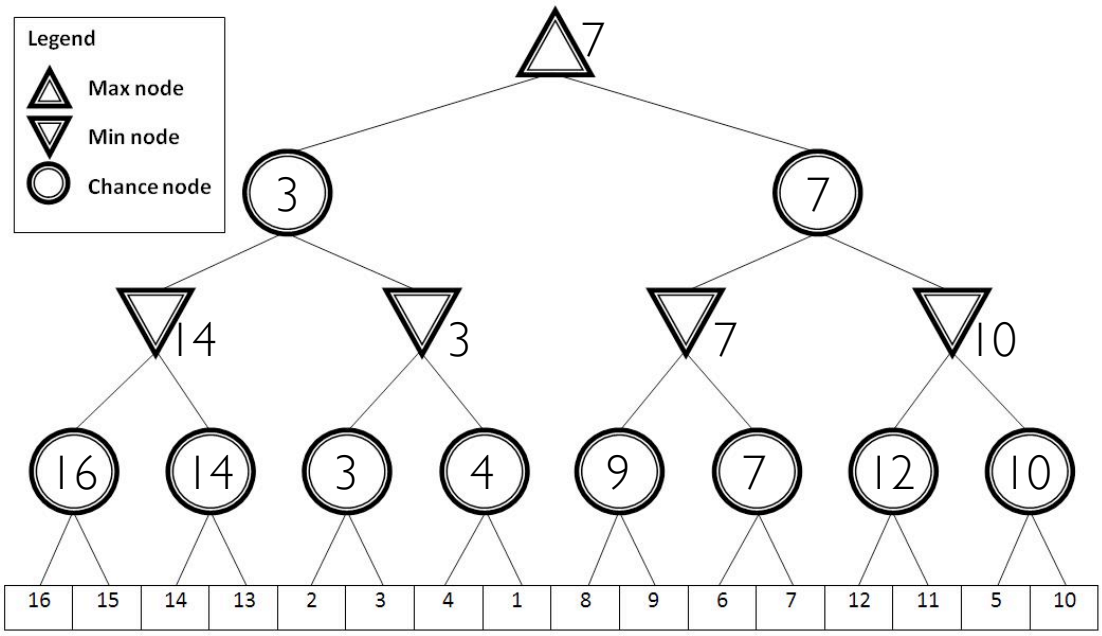
   (a) [2 points] Fill in the utility of each node of the following game tree, dealing with chance nodes as we did in class, by using expected value.

   ★ **SOLUTION:** :



(b) [2 points] Fill in the utility of each node of the following game tree. Deal with chance nodes pessimistically, by assuming the worst case for the player who cares (the player right above the chance node).

★ **SOLUTION:** :



2. **Order preserving operations with chance.** Consider a minimax game tree without chance. Let us define the term "order" to be the order of the leaves if you sorted them by their utility. A nice property of minimax is that you can change the utilities of all the leaves and minimax will tell you to make the same moves as it did before, as long as you preserve the "order". Examples of order preserving operations would be multiplying the utility of each leaf by 100, or adding 10 to the utility of the 2 best leaves. After you change the utilities of the leaves, minimax will still pick the same path through the tree.

   Now consider a minimax game tree with chance.

   (a) [3 points] If chance nodes are handled with expected value, as in part 4.1(a), can order preserving operations change what path minimax picks through the tree? Explain why or why not.

   ★ **SOLUTION:** Yes, it can change the path. For example, if the leaves with values 1,2,3,4 in the tree above were multiplied by 1/100, the top left chance node would have value less than 7, and the max player will choose the top right chance node instead of left as in part 1(a).

(b) [3 points] If chance nodes are handled with the worst case, as in part 4.1(b), can order preserving operations change what path minimax picks through the tree? Explain why or why not.

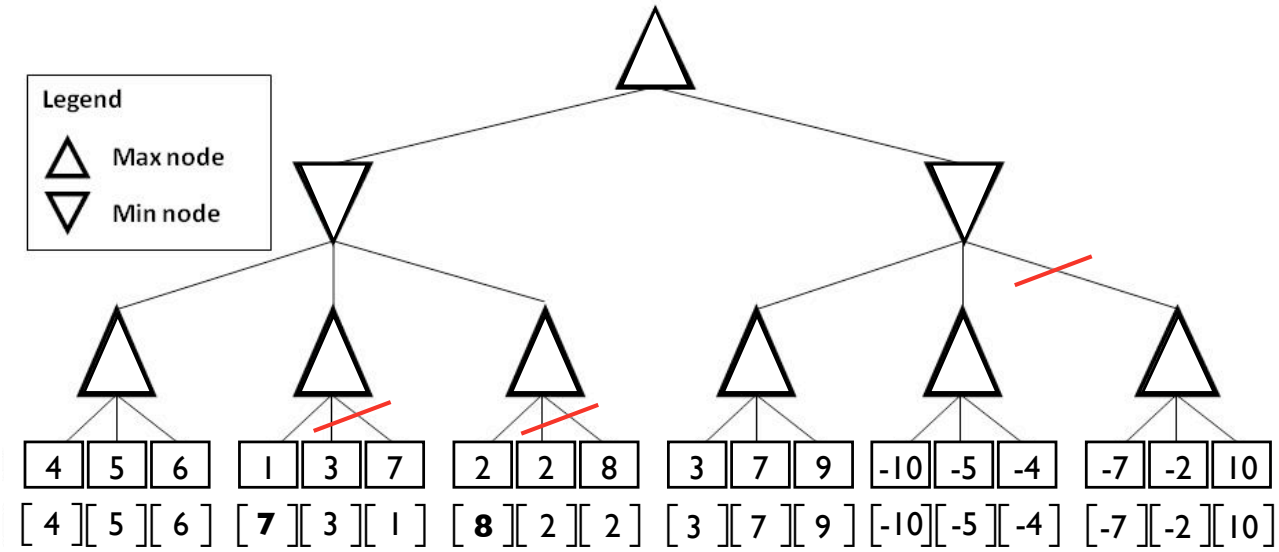
★ **SOLUTION:** No. It's deterministic, so the chance nodes will always have the value of the same node even if the node's value has been changed by an order preserving operation.

3. **Alphabeta pruning.** Consider the following minimax game tree, on which we will perform alphabeta pruning.

(a) [3 points] You have seen that amount of pruning you can do depends largely on the ordering of the leaves. Change the ordering of the leaves **within each group of three leaves** to maximize pruning, reordering only if needed to maximize pruning. Write down the new numbers in the brackets below each leaf.

(b) [2 point] Now draw a line across each edge that is pruned.

★ **SOLUTION:** Only the bolded numbers are required to be in their place:



4 CSPs [15 points]

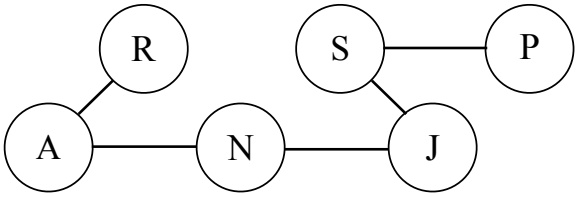
You've been asked to help in room and time-slot assignments for the following classes: Artificial Intelligence (A), Robotics (R), Java (J), Systems (S), Philosophy (P), and Networks (N). You have the option of two time slots for each of the courses (AM or PM). There are six rooms available for these courses: $R1$, $R2$, $R3$, $R4$, $R5$, $R6$. Unfortunately, video conferencing only works in $R4$ but you can assume that all the rooms have a whiteboard and projector. We have overlapping students for classes A and N. We also have overlapping students in classes P and S. Furthermore, class P is being taught remotely from Qatar so it requires a late time slot and videoconferencing. Additionally, professor Veloso is teaching both classes A and R and professor von Ahn is teaching both classes N and J (this is a hypothetical world). Finally, class R needs to be taught in the robotics lab ($R1$), and both classes J and S need to be taught in a cluster ($R3$).

1. [6 points] Formulate this problem as a CSP clearly identifying the varibles, their possible values (range), and the constraints. Also draw the constraint graph.

   ★ **SOLUTION:** Here is one formulation. We don't need to worry about room constraints since there are 6 rooms and 6 classes. Once the times are assigned, the rooms can be assigned accordingly.

   Variables: A,R,J,S,P,N. Values: 0 for AM, 1 for PM. Constraints:

   $$P = 1$$
   $$A \neq R$$
   $$N \neq J$$
   $$J \neq S$$
   $$P \neq S$$
   $$A \neq N$$



   Another formulation would be to assign two variables to each class: (1) its time and (2) room assignment. In this case the constraint graph will be a complete graph, since no two rooms should be assigned to the same time slot and the same room.

2. [3 points] Identify and justify a variable selection heuristic you would use for solving this CSP.

   ★ **SOLUTION:** Min remaining values heuristic. This heuristic will assign a value to P first and then the solution will follow.

3. [3 points] Identify and justify a **value** selection heuristic you would use for solving this CSP.

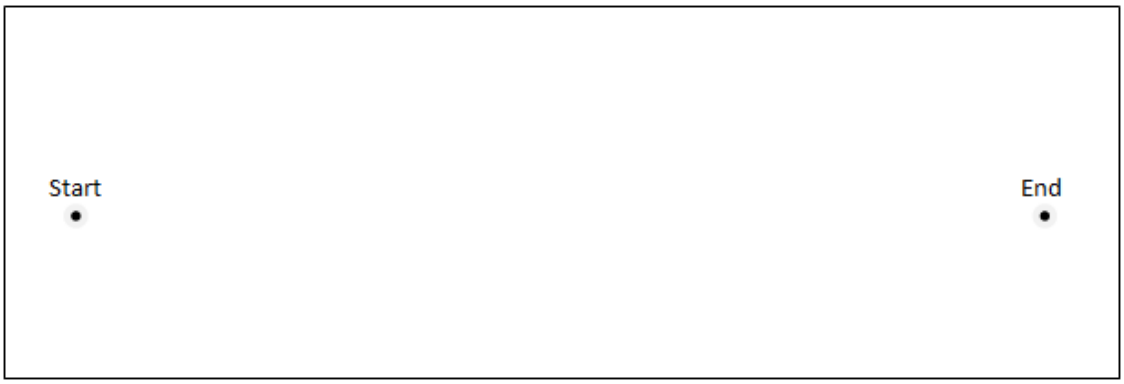
   ★ **SOLUTION:** Least constraining value heuristic.

4. [3 points] Explain why you would choose DFS over BFS for solving a CSP.

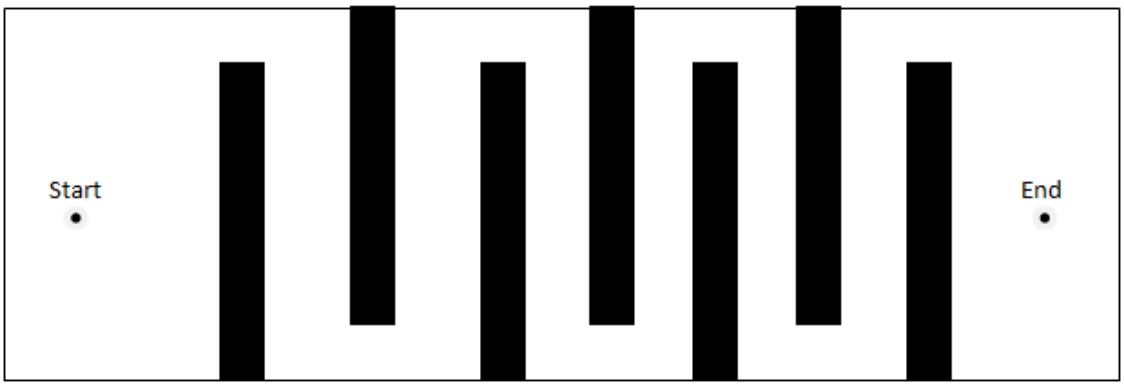
   ★ **SOLUTION:** Since the goal node is at a leaf for a CSP, and BFS has to visit all non-leaf nodes before visiting the leaves, it is guaranteed to take a long time, whereas DFS would only do this in some (bad) cases.

5 Planning [10 points]

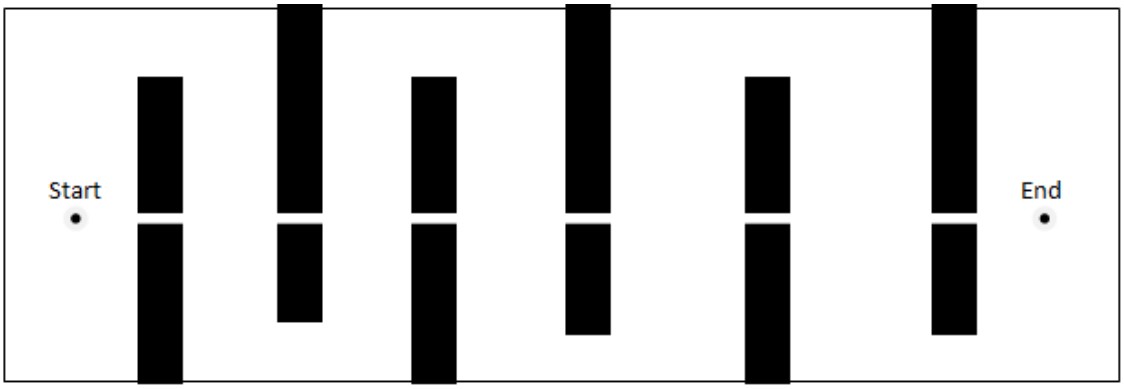
1. Consider the three worlds:



World 1



World 2



World 3

You are given three choices of an algorithm to find a path from the Start position to the End position: (1) PRM with uniform sampling, (2) RRT with p=1, and (3) RRT with p=0.75. For each of the World 1, World 2, and World 3, state which algorithm you would select, justifying your choice over the two other choices.

(a) [2 points] World 1

★ **SOLUTION:** PRM with uniform sampling would need to sample the entire space before planning a path - expensive. RRT with p = 1, would choose a direct path, and at lower values of p (e.g. 0.75), there would be minimal exploration, and the path planning efficiency would be higher than PRM.

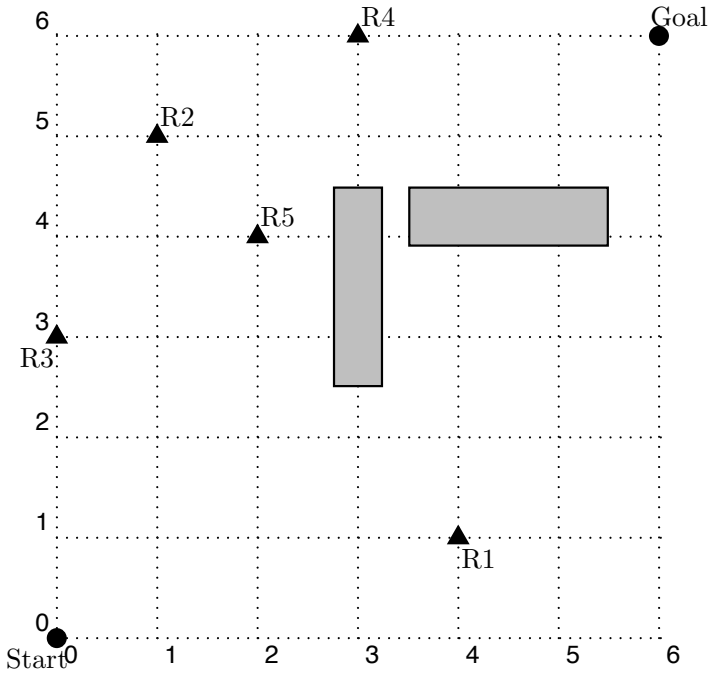
(b) [2 points] World 2

★ **SOLUTION:** PRM with uniform sampling and K large enough would place points within the corridors, and perform very well. RRT with p = 1 would simply not solve the perform. RRT with p = .75 would perform poorly, since it would tend to try to go directly towards the end point, which would cause it to fail most of the time. Also, as there is mainly only one "hard-to-find" path, the RRT's random exploration would take some time to find it.

(c) [2 points] World 3

★ **SOLUTION:** In this world, there exists a direct narrow path from the start to end location. If the direct path was so narrow that PRM's uniform sampling did not place any points in the narrow corridors, then a path generated by PRM would involve making the large turns. RRT, with p = 1, would always find the direct path.

2. In this problem, you will trace the RRT probabilistic path planning algorithm. A robot is in a 6×6 grid world depicted below:



It wants to move from the start location at $(0, 0)$ to the goal location at $(6, 6)$. There are obstacles in the world, depicted by the shaded boxes. R1, R2, R3, R4, R5 are random target points in sequence to be used by the path-planning algorithm (the first time a random target point is needed, R1 should be used; the second time a random target point is needed, R2 should be used, and so on).

Here are a couple of notes about how you will perform the trace:

- When extending the path from an existing node to a target state, the algorithm does it up to the next intersecting grid line. For instance, when the algorithm extends the path from the Start state to the Goal state, the new node will be at coordinate $(1, 1)$. Similarly, when the algorithm extends the path from the Start state to R1, the new node will be at coordinate $(1, 0.25)$.
- If the extended path is blocked by an obstacle before intersecting any grid line, the algorithm will not incorporate that path in the path tree (no new node is created).
- Label the nodes in the path tree in the order they are created. So the first node created will be labeled as 1, and so on. If at a particular step, the algorithm fails to extend a path due to an obstacle, the label should be added to the node from which the path is supposed to originate. For instance, if at step 8, the algorithm tries to extend a path from node 5 to some target state but fails, node 5 is relabeled as node 5, 8.

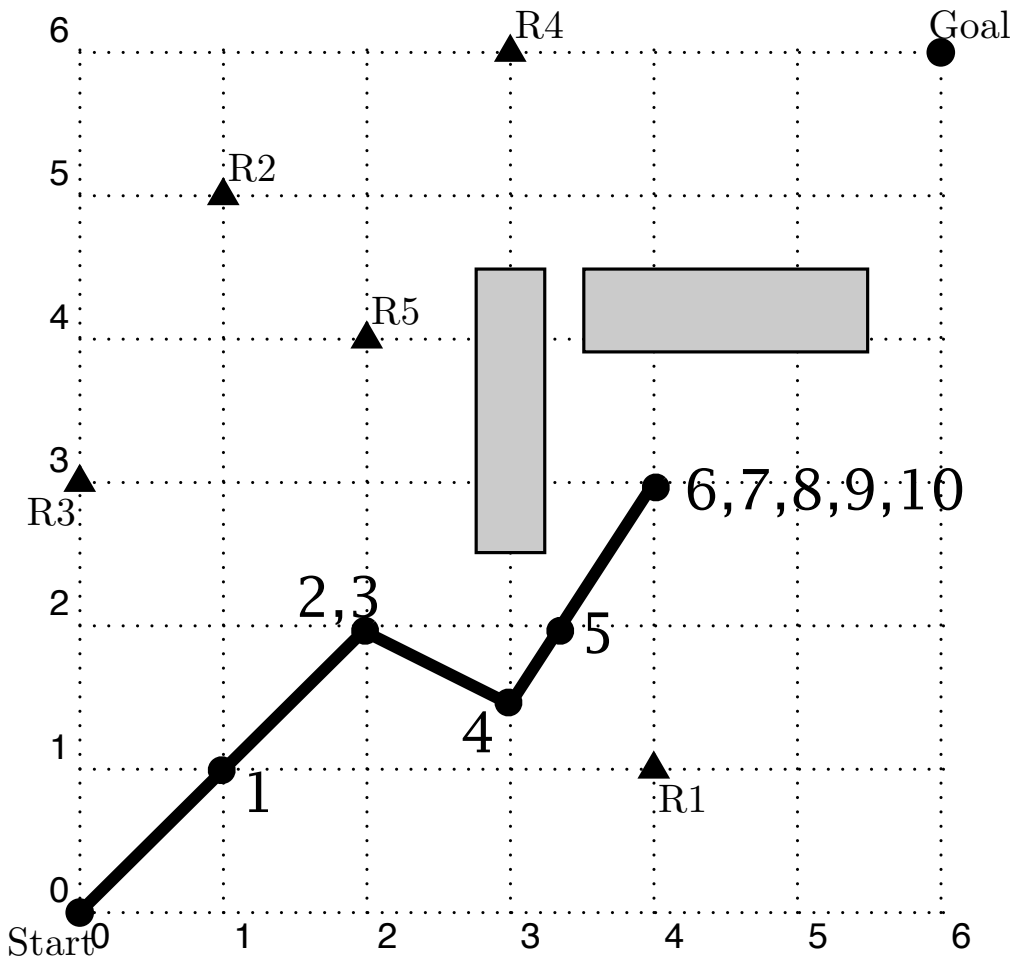
The following is a sequence of 10 uniform random numbers that the algorithm generates:

$$0.72, 0.47, 0.28, 0.87, 0.32, 0.67, 0.73, 0.25, 0.41, 0.65.$$

The RRT algorithm extends the path to the Goal state when the drawn random number is less than p, and otherwise extends the path to the next random point.

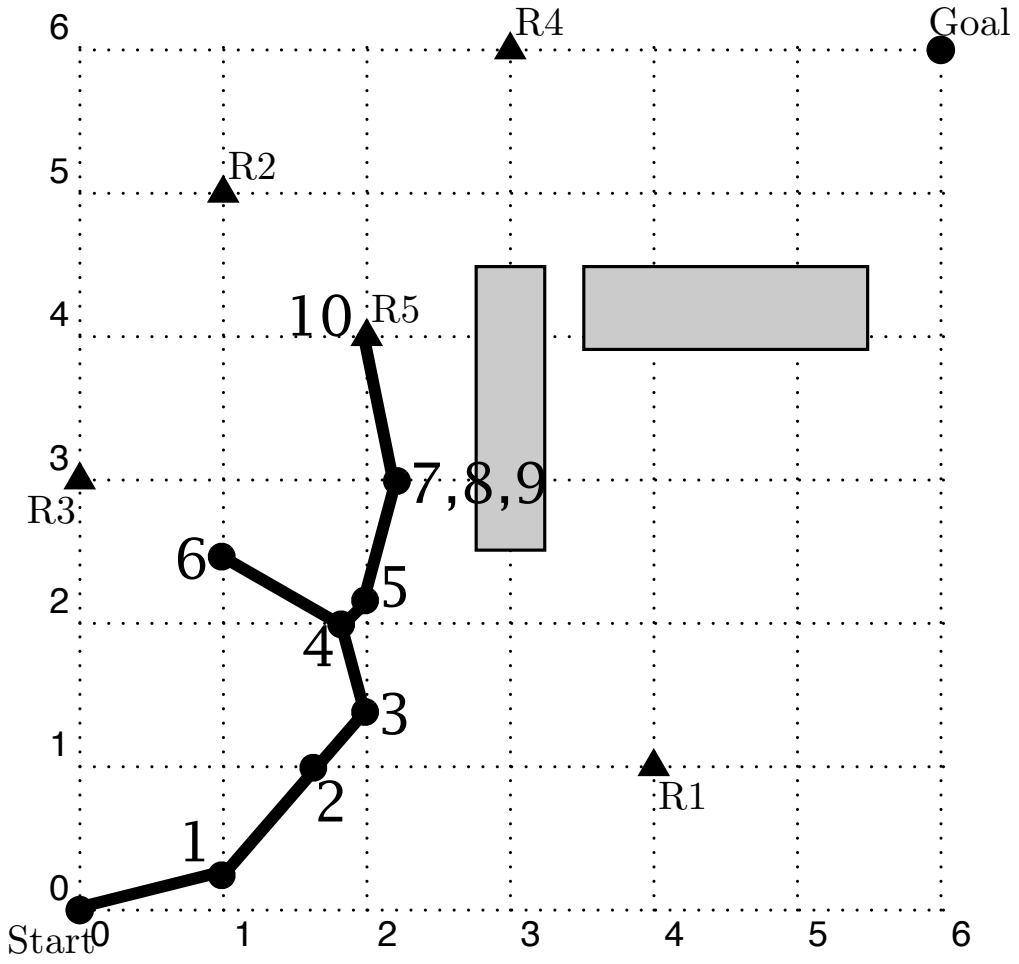
(a) [2 points] Trace the RRT algorithm for $p = 0.8$ for 10 steps on the grid below:

★ **SOLUTION:** :



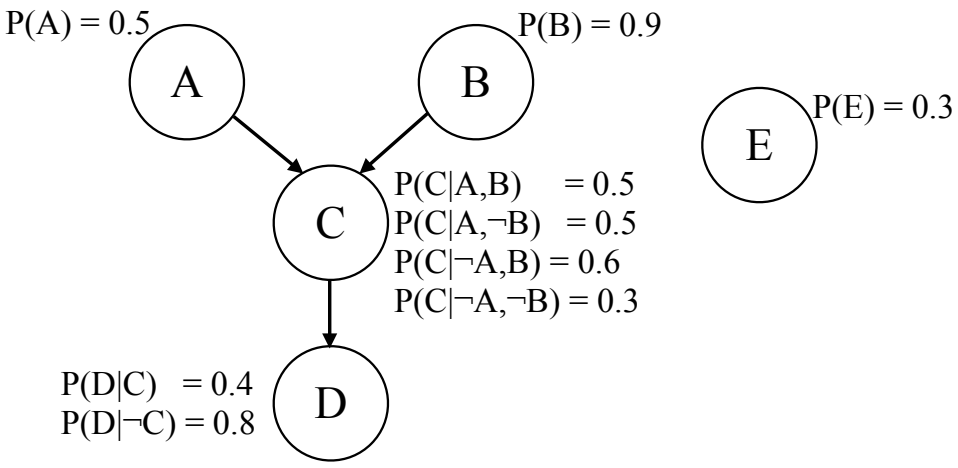
(b) [2 points] Trace the RRT algorithm for $p = 0.6$ for 10 steps on the grid below:

★ **SOLUTION:** :



6 Bayes Nets [15 points]

Consider the following Bayes Net with five random variables.



1. State whether each of the following statements is true or false. Justify each answer in one sentence.

   (a) [1 point] B and E are independent.

   ★ **SOLUTION:** True, since E is independent of all other variables.

   (b) [1 point] A and D are independent.

   ★ **SOLUTION:** False, since they are dependent unless given C.

   (c) [1 point] $P(\neg D \mid \neg C) = 0.6$

   ★ **SOLUTION:** False. $P(\neg D \mid \neg C) = 0.2$

   (d) [1 point] The joint probability function of all variables has 15 entries.

   ★ **SOLUTION:** False. Since there are 5 variables, there are 32 entries. (31 if you only count how many numbers are *required* to define the joint probability function.)

2. Compute:

   (a) [1 point] $P(D \mid \neg C, E)$

★ **SOLUTION:** $P(D|\neg C, E) = P(D|\neg C) = 0.8$

(b) [1 point] $P(\neg C \mid \neg A, B)$

★ **SOLUTION:** $P(\neg C \mid \neg A, B) = 1 - P(C \mid \neg A, B) = 1 - 0.6 = 0.4.$

(c) [3 points] $P(A, \neg B, C, D)$

★ **SOLUTION:**

$$\begin{aligned} P(A, \neg B, C, D) &= P(A)P(\neg B)P(C|A, \neg B)P(D|C) \\ &= 0.5 * 0.1 * 0.5 * 0.4 \\ &= 0.01 \end{aligned}$$

(d) [3 points] $P(D \mid A, \neg B, C)$

★ **SOLUTION:** $P(D \mid A, \neg B, C) = P(D \mid C) = 0.4$

(e) [3 points] $P(A, D)$

★ **SOLUTION:**

$$\begin{aligned} P(A, D) &= \sum_{b,c} P(A, b, c, D) \\ &= \sum_{b,c} P(A)P(b)P(c|A, b)P(D|c) \\ &= P(A) \sum_{b} P(b) \sum_{c} P(c|A, b)P(D|c) \\ &= P(A)[P(B)(P(C|A, B)P(D|C) + P(\neg C|A, B)P(D|\neg C)) \\ &\quad +P(\neg B)(P(C|A, \neg B)P(D|C) + P(\neg C|A, \neg B)P(D|\neg C))] \\ &= 0.5 * (0.9 * (0.5 * 0.4 + 0.5 * 0.8) + 0.1 * (0.5 * 0.4 + 0.5 * 0.8)) \\ &= 0.3 \end{aligned}$$