CS 574 Midterm Review

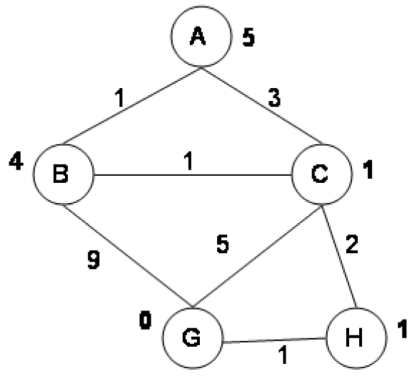
Relevant readings that may be covered on the midterm: The following list is the official reading list that covers material relevant for the Oct 3rd midterm.

- Russel and Norvig text: ch 3.1-3.5, 5.1-5.3, 6.1-6.4
- Sutton and Barto text: ch 3
- Bertsekas (Dynamic Programming and Optimal Control Vol I) text: ch 1.1-1.3, ch 5.1-5.4

Instructions: We suggest that you complete this midterm review in its entirety before lecture on Monday Oct 1 when we will go over the solutions in class. This will be the best way for you to test your command of the material and identify areas that you need to study up on. This is meant as a *study guide only* and covers material that is most likely to be tested on the midterm. Note that you are responsible for material in the readings and this may include material that is not represented in this review guide. You should additionally review material from the reading list above and study from the problems labeled "www" in the Dynamic Programming text which have solutions posted on the textbook website.

1. **A* search and properties of A* search**

Consider the graph shown below where the numbers on the links are link costs and the numbers next to the states are heuristic estimates. Note that the arcs are undirected. Let A be the start state and G be the goal state.



a) Simulate A* search with a strict expanded list on this graph. At each step, show the path to the state of the node that's being expanded, the length of that path, the total estimated cost of the path (actual + heuristic), and the current value of the expanded list (as a list of states). You are welcome to use scratch paper or the back of the exam pages to simulate the search. However, please transcribe (only) the information requested into the table given below.

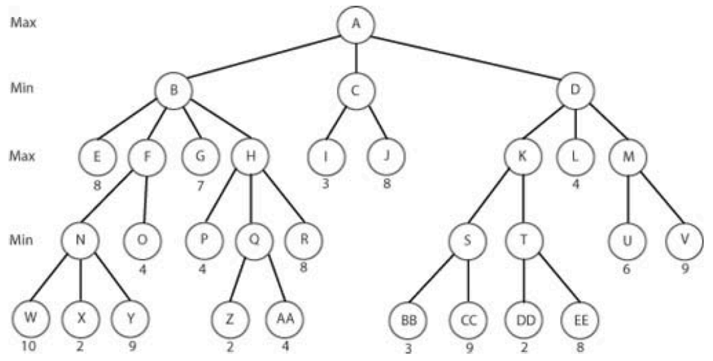
Path to State Expanded	Length of Path	Total Estimated Cost	Expanded List
A	0	5	(A)

b) Is the heuristic given in Problem 2 admissible? Explain.

c) Is the heuristic given in Problem 2 consistent? Explain.

d) Did the A* algorithm with strict expanded list find the optimal path in the previous example? If it did find the optimal path, explain why you would expect that. If it didn't find the optimal path, explain why you would expect that and give a simple (specific) change of state values of the heuristic that would be sufficient to get the correct behavior.

2. **Adversarial search (minimax and alpha-beta pruning):**
You are playing a game of chess against your online boyfriend/girlfriend on some online game engine. It's almost the end of the game when your opponent requests to pause the game in order to use the bathroom. You agree but while you wait you decide to sketch a game tree of the remaining moves. The game tree is displayed below:



a) After looking at the game tree, you decide to figure out what your best move is. Use minimax to determine the optimal path through the game tree. Show the minimax value for each node on the diagram, then list the nodes on the best-move path.

b) Investigate which branches can be eliminated by completing alpha-beta pruning. You are to assume that you have already done the search down the left branch of the tree, discovering that the minimax value of node B is 4. Continue processing the tree from left to right, and **circle the nodes** below for which static values **will be** computed by alpha-beta. Also**, cross out the nodes** for which static values **will not be** computed.

I	J	BB	CC	DD	EE	L	U	V

3. **Constraint Satisfaction Problems**

Consider assigning colors to a checkerboard so that squares that are adjacent vertically or horizontally do not have the same color. We know that this can be done with only two colors, say red (R) and black (B). We will limit our discussion to five squares on a 3x3 board, numbered as follows:

```
1 | 2 | 3
---------
4 | 5 |
---------
  |   |
```

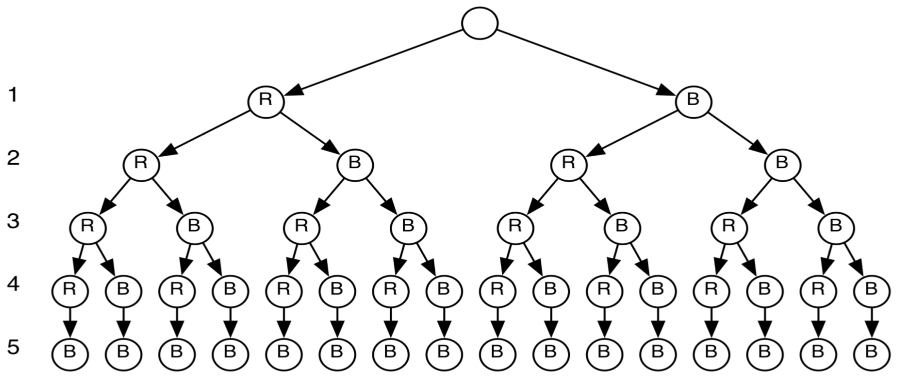
Let's look at the CSP formulation of this problem. Let the squares be the variables and the colors be the values. All the variables have domains {R, B}.

a. If we run full constraint propagation on the initial state, what are the resulting domains of the variables?

b. Say, instead, the initial domain of variable 5 is restricted to {B}, with the other domains as before. If we now run full constraint propagation, what are the resulting domains of the variables?

c. If in the initial state (all variables have domains {R, B}), we assign variable 1 to R and do forward checking, what are the resulting domains of the other variables?

d. Assume that during backtracking we first attempt assigning variables to R and then to B. Assume, also, that we examine the variables in numerical order, starting with 1. Also, let the domain of variable 5 be *{*B*}*, the other domains are *{*R,B*}*. In the following tree, which shows the space of assignments to the 5 variables we care about, indicate how pure backtracking(BT) would proceed by placing a checkmark next to any assignment that would be attempted during the search and crossing out the nodes where a constraint test would fail. Leave unmarked those nodes that would never be explored.



e. If we use backtracking with forward checking (BT-FC) in this same situation, give a list of all the assignments attempted, in sequence. Use the notation variable=color for assignments, for example, 1=R.

f. If we use backtracking with forward checking (BT-FC) but with dynamic variable ordering, using the most-constrained-variable strategy, give a list of all the variable assignments attempted, in sequence. If there is a tie between variables, use the lowest-numbered one first. Use the notation variable = color for assignments, for example, 1=R.

4. **Dynamic Programming: Finite Horizon Problems**

Alexei plays a game that starts with a deck with b "black" cards and r "red" cards. Alexie knows b and r. At each time period he draws a random card and decides between the following two options:

1) Without looking at the card, "predict" that it is black, in which case he wins the game if the prediction is correct and loses if the prediction is incorrect.
2) "Discard" the card, after looking at its color, and continue the game with one card less.

If the deck has only black cards he wins the game, while if the deck has only red cards he loses the game. Alexei wants to find a policy that maximizes his probability of a win.

a) Formulate Alexei's problem into the format of a finite-horizon basic problem. Identify states, controls, disturbances, etc.

b) Write the DP algorithm. Use induction to show that the optimal probability of a win starting with b black cards and r red cards is b/b+r.

c) Characterize the optimal policies.

5. **Dynamic Programming: Infinite Horizon Problems**

A workshop manager has just bought an expensive new machine, and at each day he has two options: maintain the machine at cost M, or not maintain it. However, in the latter case, he runs the risk of a breakdown, which costs B and occurs with probability p_j, where j is the number of consecutive days after the preceding breakdown (if any) that the machine has not been maintained (e.g., on the first day with no maintenance the probability of breakdown is p_1, on the second successive day of no maintenance the probability of breakdown is p_2, etc). Assume that p_j is monotonically nondecreasing in j, and that there exists an interger m such that $p_m B > M$.

a) Formulate this as an infinite horizon discounted cost problem with states 0,1,…,m, and write the corresponding Bellman's equation.

b) Characterize as best as you can the optimal policy.