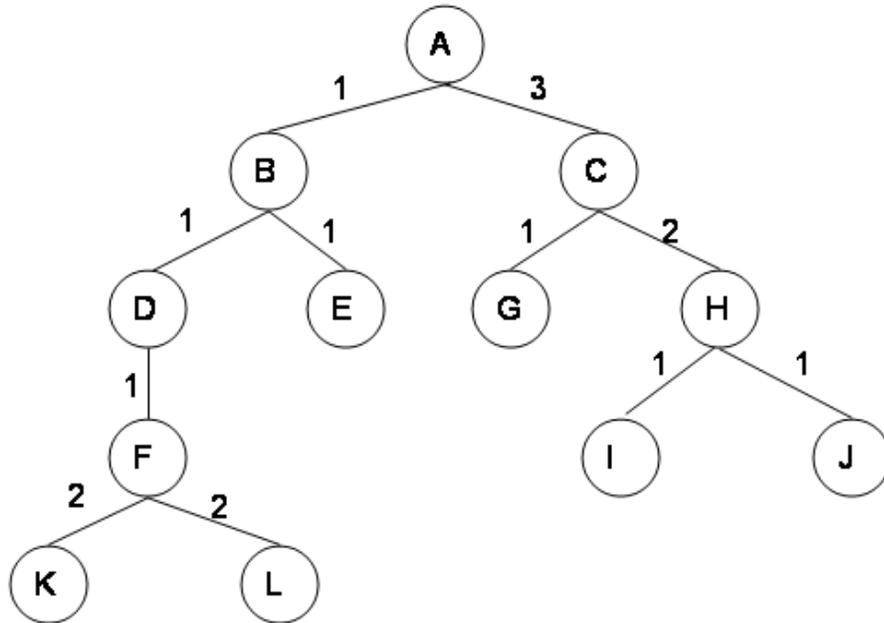


1 Tree Search (12 points)

Consider the tree shown below. The numbers on the arcs are the arc lengths.

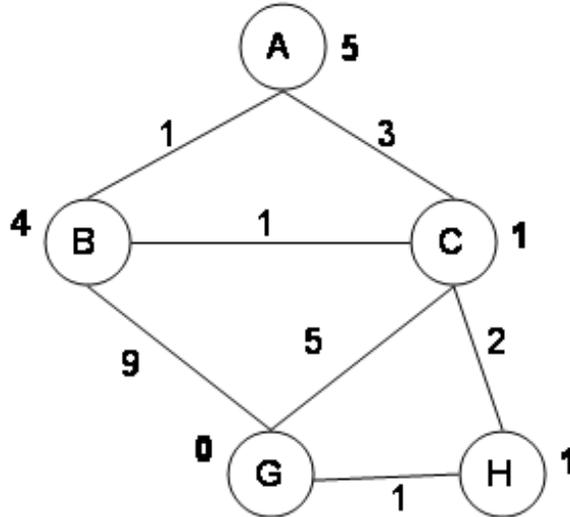


Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state *G*. No visited or expanded lists are used. What order would the states be expanded by each type of search? Stop when you expand *G*. Write only the sequence of states expanded by each search.

Search Type	List of states
Breadth First	
Depth First	
Progressive Deepening Search	
Uniform Cost Search	

2 Graph Search (10 points)

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.



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)

3 Heuristics and A* (8 points)

1. Is the heuristic given in Problem 2 admissible? Explain.
2. Is the heuristic given in Problem 2 consistent? Explain.
3. 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.

5 CSP (16 points)

Let's look at the problem of scheduling programs on a set of computers as a constraint satisfaction problem.

We have a set of programs (jobs) J_i to schedule on a set of computers (machines) M_j . Each job has a maximum running time R_i . We will assume that jobs (on any machines) can only be started at some pre-specified times T_k . Also, there's a T_{max} time by which all the jobs must be finished running; that is, start time + running time is less than or equal to max time. For now, we assume that any machine can execute any job.

Let's assume that we attack the problem by using the jobs as variables and using values that are each a pair (M_j, T_k) . Here is a simple example.

- Running time of J_1 is $R_1 = 2$
 - Running time of J_2 is $R_2 = 4$
 - Running time of J_3 is $R_2 = 3$
 - Running time of J_4 is $R_4 = 3$
 - Starting times $T_k = \{1, 2, 3, 4, 5\}$
 - Two available machines M_1 and M_2 .
 - The max time is $T_{max} = 7$.
 - An assignment would look like $J_1 = (M_2, 2)$, that is, run job J_1 on machine M_2 starting at time 2.
1. What are the constraints for this type of CSP problem? Write a boolean expression (using logical connectives and arithmetic operations) that must be satisfied by the assignments to each pair of variables. In particular:
 - J_i with value (M_j, T_k)
 - J_m with value (M_n, T_p)

2. Write down a complete valid solution to the example problem above.
 - $J_1 =$
 - $J_2 =$
 - $J_3 =$
 - $J_4 =$

3. Which variable would be chosen first if we did BT-FC with dynamic ordering of variables (most constrained)? Why?

4. If we do constraint propagation in the initial state of the example problem, what domain values (if any) are eliminated? Explain.

5. If we set $J_2 = (M_1, 1)$, what domain values are still legal after forward checking?
 - $J_1 \in$

 - $J_2 \in$

 - $J_3 \in$

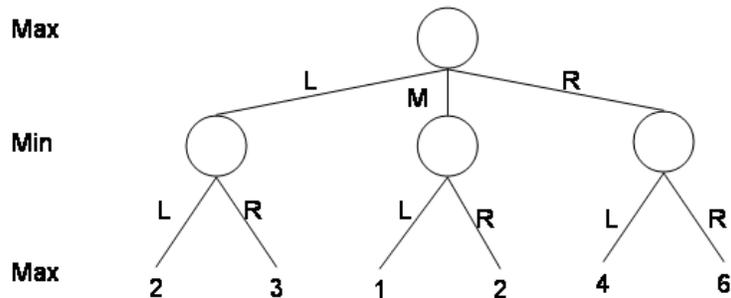
 - $J_4 \in$

6. We could have formulated this problem using the machines M_j as the variables. What would the values be in this formulation, assuming you have N machines and have K jobs to schedule?

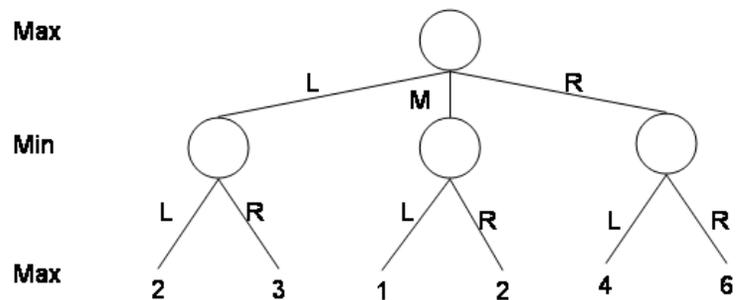
7. What are some disadvantages of this formulation (using machines as variables)?

6 Game Search (10 points)

Consider the game tree shown below. The top node is a max node. The labels on the arcs are the moves. The numbers in the bottom layer are the values of the different outcomes of the game to the max player.

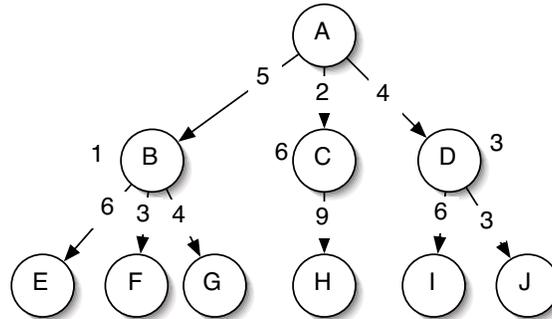


1. What is the value of the game to the max player?
2. What first move should the max player make?
3. Assuming the max player makes that move, what is the best next move for the min player, assuming that this is the entire game tree?
4. Using alpha-beta pruning, consider the nodes from **right to left**, which nodes are cut off? Circle the nodes that are not examined.



1 Tree Search (10 points)

Consider the tree shown below. The numbers on the arcs are the arc lengths; the numbers near states B, C, and D are the heuristic estimates; all other states have a heuristic estimate of 0.

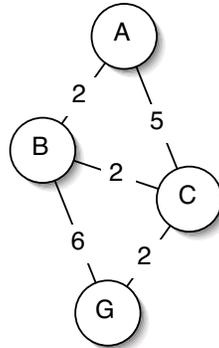


Assume that the children of a node are expanded in alphabetical order when no other order is specified by the search, and that the goal is state *J*. No visited or expanded lists are used. What order would the states be expanded by each type of search. Write only the sequence of states expanded by each search.

Search Type	List of states
Breadth First	
Depth First	
Progressive Deepening Search	
Best-First Search	
A* Search	

2 Graph Search (8 points)

Consider the graph shown below. Note that the arcs are undirected. Let A be the start state and G be the goal state.



Simulate uniform cost search with a strict expanded list on this graph. At each step, show the state of the node that's being expanded, the length of that path, and the current value of the expanded list (as a list of states).

State Expanded	Length of Path	Expanded List
A	0	(A)

3 A* Algorithm (12 points)

1. Let's consider three elements in the design of the A* algorithm:

- The heuristic, where the choices are:
 - **arbitrary** heuristic
 - **admissible** heuristic
 - **consistent** heuristic
- History:
 - **none**
 - **visited** list
 - **strict** expanded list
 - **non-strict** expanded list
- Pathmax
 - **Use** pathmax
 - **Don't use** pathmax

In the table below, indicate all the combinations that *guarantee* that A* will find an optimal path. Not all rows have to be filled. If multiple values works for any of Heuristic, History and Pathmax, independent of the other choices, you can write the multiple values in one row. So

Heuristic	History	Pathmax
A,B	C	D,E

can be used to represent all of: A,C,D; A,C,E; B,C,D; and B,C,E.

Heuristic	History	Pathmax

2. In the network of problem 2, assume you are given the following heuristic values:

$$A = 5; B = 4; C = 0; G = 0$$

Is this heuristic:

- Admissible? Yes No
- Consistent? Yes No

Justify your answer very briefly.

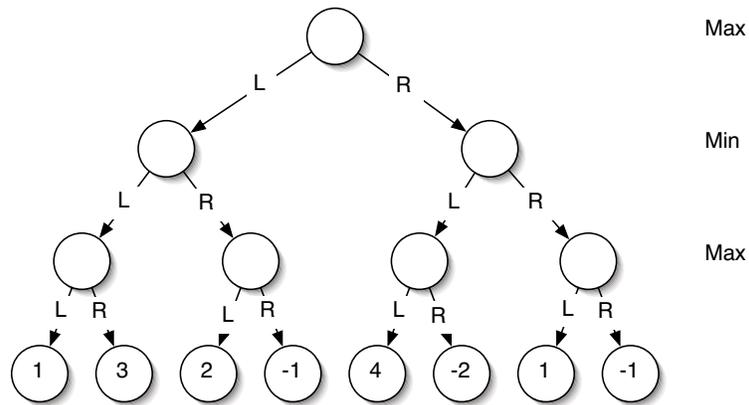
3. With the heuristic above will A* using a strict expanded list find the optimal path?

Yes No

Justify your answer very briefly.

4 Game Search (5 points)

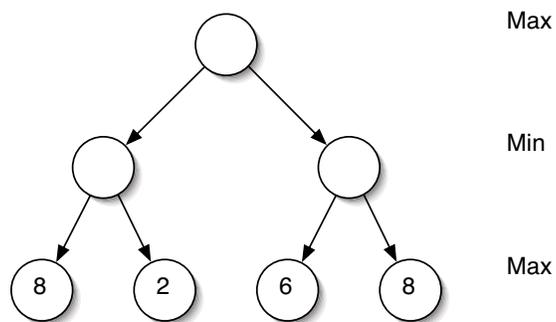
Consider the game tree shown below. Assume the top node is a max node. The labels on the arcs are the moves. The numbers in the bottom layer are the values of the different outcomes of the game to the max player.



1. What is the value of the game to the max player?
2. What first move should the max player make?
3. Assuming the max player makes that move, what is the best next move for the min player, assuming that this is the entire game tree?

5 Alpha-Beta Pruning (5 points)

In the following game tree, are there any alpha-beta cutoffs?



- Consider the nodes from left to right, which nodes are cutoff? Circle the nodes that are not examined and label them with L.
- Consider the nodes from right to left, which nodes are cutoff? Circle the nodes that are not examined and label them with R.

6 CSP Methods (15 points)

Let's consider some combinations of CSP methods. For each of the combinations described below say very briefly whether:

1. It would be **well-defined** to combine them, in the sense that none of the implementation assumptions of the methods as we defined them are violated in the combination.
2. It could be **useful**, that is, one would expect improved performance (over using only the first method mentioned), at least in some problems. Improved performance could be either from being better able to solve problems or improved efficiency (indicate which).

In each case, circle Yes or No for each of Well-Defined? and Useful? and give a very brief explanation of your answers.

Warning: Please pay careful attention to the definition of the methods being combined, we are referring to the original definition of the methods – in isolation. Almost any idea can be made to work with any other idea with sufficient creativity - but that's not what we are looking for in this problem.

- Full constraint propagation (CP) followed by pure backtracking (BT).

1. Well-Defined? Yes No
2. Useful? Yes No

- Full constraint propagation (CP) combined with forward checking (FC).

1. Well-Defined? Yes No
2. Useful? Yes No

- Pure backtracking (BT) combined with dynamic variable (most constrained) and value ordering (least constraining).

1. Well-Defined? Yes No

2. Useful? Yes No

- Min-conflict-hill-climb (MC) combined with dynamic variable (most constrained) and value ordering (least constraining).

1. Well-Defined? Yes No

2. Useful? Yes No

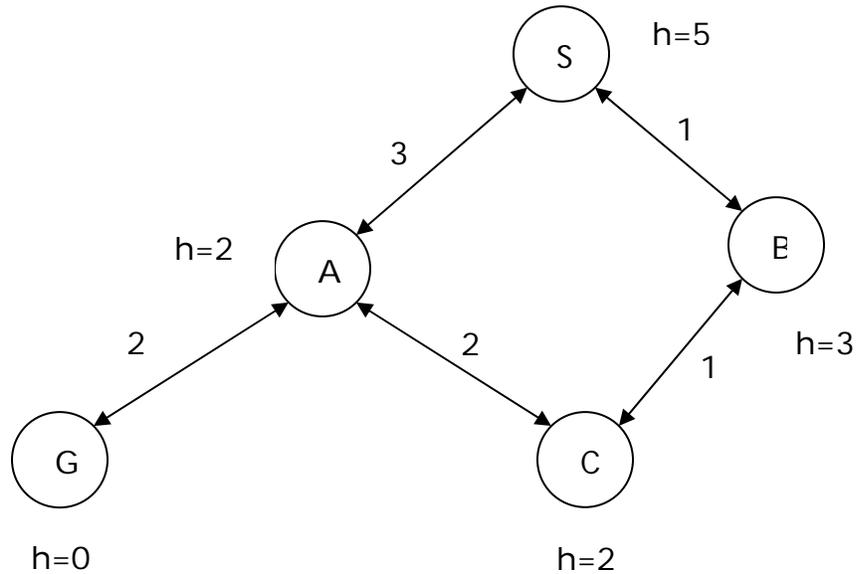
- Pure backtracking (BT) combined with full constraint propagation (CP) after each tentative assignment.

1. Well-Defined? Yes No

2. Useful? Yes No

Problem 1 – Search

Below is a graph to be searched (starting at S and ending at G). Link/edge costs are shown as well as heuristic estimates at the states. You may not need all the information for every search.



Draw the complete search tree for this graph. Label each node in the tree with the cost of the path to that node and the heuristic cost at that node. When you need to refer to a node, use the name of the corresponding state and the length of the path to that node.

For each of the searches below, just give a list of node names (state name, length of path) drawn from the tree above. Break ties using alphabetical order.

1. Perform a depth-first search using a visited list. Assume children of a state are ordered in alphabetical order. Show the sequence of nodes that are expanded by the search.
2. Perform a best-first (greedy search) without a visited or expanded list. Show the sequence of nodes that are expanded by the search.
3. Perform a Uniform Cost Search without a visited or expanded list. Show the sequence of nodes that are expanded by the search.
4. Perform an A* search (no pathmax) without an expanded list. Show the sequence of nodes that are expanded by the search.

Is the heuristic in this example

1. admissible?

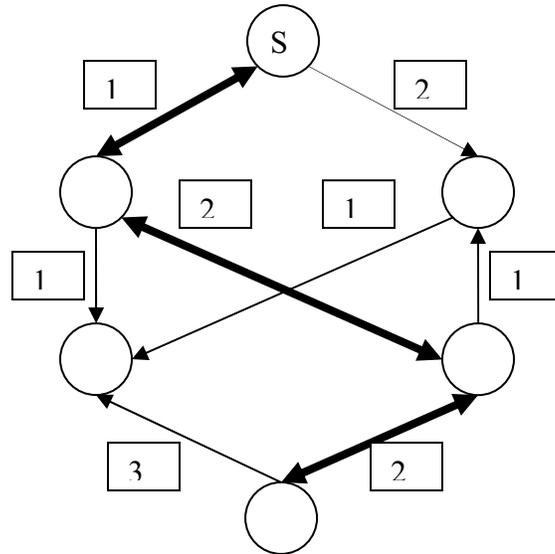
2. consistent?

Justify your answer, briefly.

For each of the following situations, pick the search that is most appropriate (be specific about visited and expanded list). Give a one sentence reason why you picked it. If you write a paragraph, we will not read it.

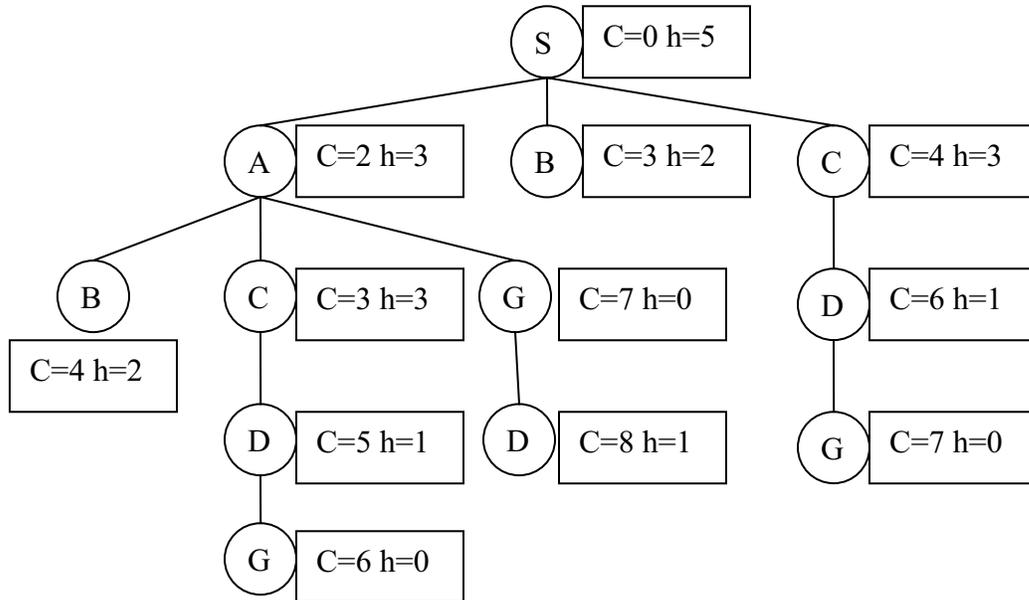
1. We have a very large search space with a large branching factor and with possibly infinite paths. We have no heuristic. We want to find paths to the goal with minimum numbers of state.
2. We have a space with a manageable number of states but lots of cycles in the state graph. We have links of varying costs but no heuristic and we want to find shortest paths.
3. Our search space is a tree of fixed depth and all the goals are the leaves of the tree. We have a heuristic and we want to find any goal as quickly as possible.
4. We have a space with a manageable number of states but lots of cycles in the state graph. We have links of varying costs and an admissible heuristic and we want to find shortest paths.

Problem 1: Search (25 points)



A. Construct the search tree for the graph above, indicate the path length to each node. The numbers shown above are link lengths. Pay careful attention to the arrows; some are bi-directional (shown thick) while some are uni-directional.

B. Using the following search tree (different from Part A), perform the searches indicated below (always from S to G). Each node shows both the total path cost to the node as well as the heuristic value for the corresponding state.



For each of the searches below, write the sequence of nodes **expanded** by the search. Specify a node by writing the name of the state and the length of the path (shown as C=x above), e.g. S0, B3, etc. Break ties using alphabetical order.

1. Depth First Search (no visited list)

2. Breadth First Search (with visited list)

3. Uniform Cost Search (with strict expanded list)

4. A* (without expanded list)

C. Choose the most efficient search method that meets the criteria indicated below.

Explain your choice.

1. You are given a state graph with link costs. The running time of the algorithm should be a function of the number of states in the graph and the algorithm should guarantee that the path with shortest path cost is found.

2. You are given a state graph with link costs and consistent heuristic values on the states. The running time of the algorithm should be a function of the number of states in the graph and the algorithm should guarantee that the path with shortest path cost is found.

You are given a state graph with no link costs or heuristic values. The algorithm should find paths to a goal with the least number of states and the space requirements should depend on the depth of the first goal found and not be exponential in that depth.

Problem 5 – CSP

Assume we have four variables (A, B, C, D) and two values (1, 2). We write variable/value assignments as A1, B2, etc. Assume the only legal values are as listed below:

- A-B: A1-B1, A2-B1, A2-B2
- A-C: A1-C2, A2-C1
- A-D: A2-D2
- B-C: B1-C2, B2-C1
- B-D: B2-D2
- C-D: C1-D1, C1-D2

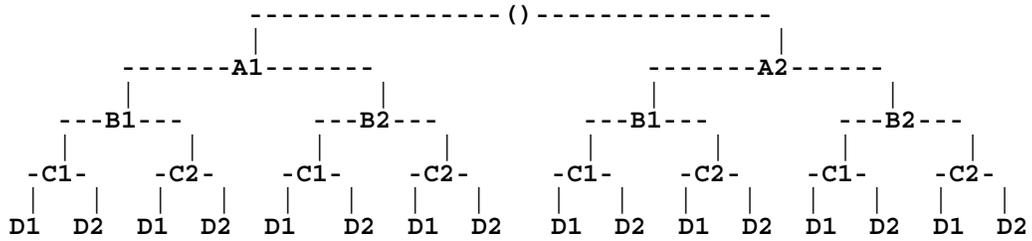
An entry in the matrix below indicates a consistent assignment. This is simply another way of presenting the same information in the list above.

	A1	A2	B1	B2	C1	C2	D1	D2
A1			X			X		
A2			X	X	X			X
B1	X	X				X		
B2		X			X			X
C1		X		X			X	X
C2	X		X					
D1					X			
D2		X		X	X			

Assume you do full constraint propagation in this problem. Show the legal values for each variable after propagation:

- A :
- B :
- C :
- D :

Here's the search tree (as in the PS):



Assume that you do the backtracking with forward checking. Show the assignments in order as they are generated during the search.

What is the first solution found in the search?

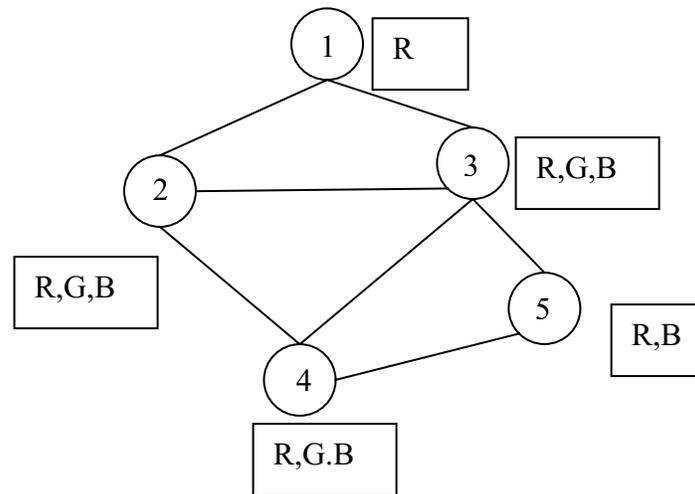
The constraints – repeated for easy reference:

- A-B: A1-B1, A2-B1, A2-B2
- A-C: A1-C2, A2-C1
- A-D: A2-D2
- B-C: B1-C2, B2-C1
- B-D: B2-D2
- C-D: C1-D1, C1-D2

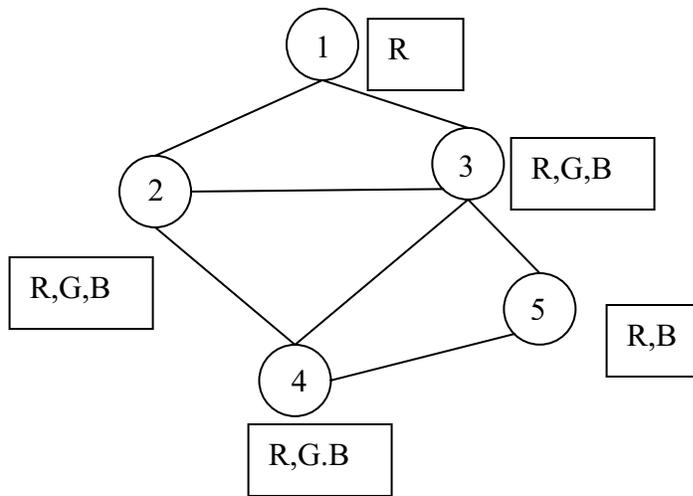
	A1	A2	B1	B2	C1	C2	D1	D2
A1			X			X		
A2			X	X	X			X
B1	X	X				X		
B2		X			X			X
C1		X		X			X	X
C2	X		X					
D1					X			
D2		X		X	X			

Problem 5: CSP (15 points)

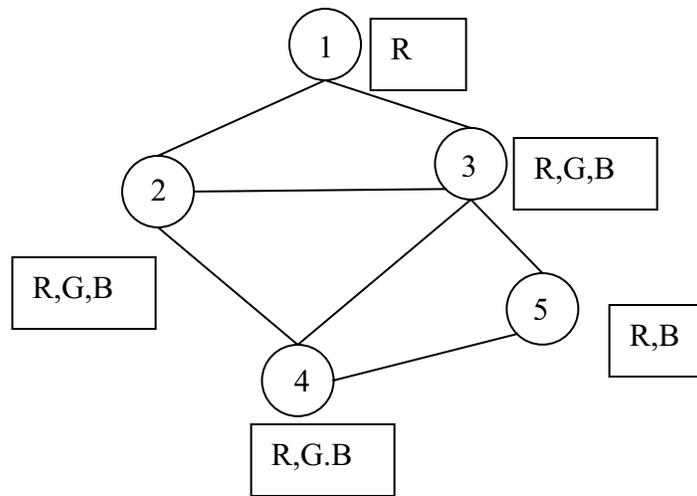
Consider the following constraint graph for a graph coloring problem (the constraints indicate that connected nodes cannot have the same color). The domains are shown in the boxes next to each variable node.



A. What are the variable domains after a full constraint propagation?



B. Show the sequence of variable assignments during a pure backtracking search (do not assume that the propagation above has been done), assume that the variables are examined in numerical order and the values are assigned in the order shown next to each node. Show assignments by writing the variable number and the value, e.g. 1R. **Don't write more than 10 assignments, even if it would take more to find a consistent answer.**



C. Show the sequence of variable assignments during backtracking with forward checking, assume that the variables are examined in numerical order and the values are assigned in the order shown next to each node. Show assignments by writing the variable number and the value, e.g. 1R.