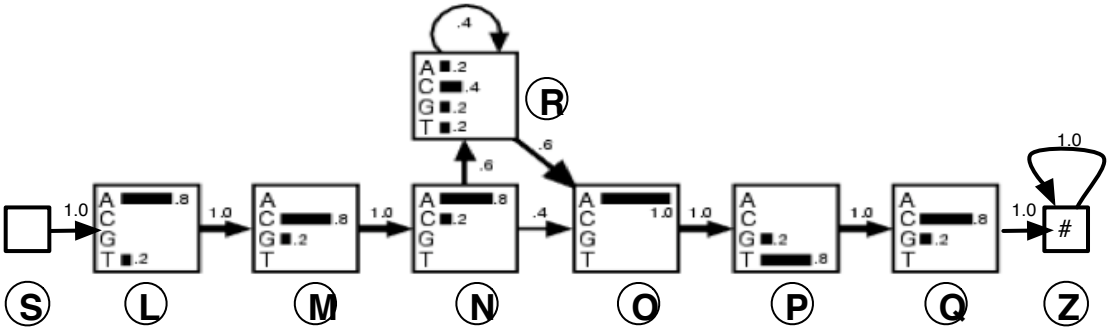
# 7 Temporal Models (20 points)

1. For the Rain-Umbrella HMM model (from class and Figure 15.2 in AIMA) with 2 time periods, show how **variable elimination** operates to answer the following query $P(R_1|U_1 = T, U_2 = T)$. You do not need to do numerical calculations.

2. Explain the relationship between the variable elimination computation and the **forward/backward algorithm**. Be specific, referring to the factors obtained during variable elimination.

# 5 Hidden Markov Models (20 points)

The following picture specifies a Hidden Markov Model for a DNA sequence "pattern", of length at least 6. There are two special states, S which is the start state and Z which is the end state. We always start in state S, that is, the state distribution at time 0 has all the probability on state S, there are no observations in state S. Once we transition into state Z, all the observations are of an "end of string" symbol (which is not relevant in this problem). The other states show the distribution over the observations possible in that state (one of the four bases of DNA - A,C,G,T). The transition probabilities between the states are given on the arcs between the states.




1. (10 pts) What is the distribution over the states after seeing "ACAAA", i.e. $P(X_5|E_{1:5} = ACAAA)$. Compute the probabilities and write them in the 5th table column. Write the probabilities as sums and products of entries in the diagram, do **not** multiply or add the numbers. So, an entry in the table could be $0.3 \times 0.4 \times 0.2 + 0.5 \times 0.2 \times 0.2$. The table is big enough to keep track of intermediate results, but you do not need to fill in the whole table. If you do, you might want to label important entries with letters, e.g. a and b, so you can reuse them. In this range of time, S and Z have 0 probability, so they're not shown.

| State | $t = 1$ | $t = 2$ | $t = 3$ | $t = 4$ | $t = 5$ |
|---|---|---|---|---|---|
| L | | | | | |
| M | | | | | |
| N | | | | | |
| O | | | | | |
| P | | | | | |
| Q | | | | | |
| R | | | | | |

2. (1 pt) Which algorithm should be used for computing the distribution above?

3. (4 pts) How would you expect the distribution you computed above (for $X_5$) to change after seeing "ACAAAATC", that is, $P(X_5|E_{1:8} = ACAAAATC)$? You do not need to compute the new distribution numerically; indicate the qualitative changes to the state probabilities.

4. (1 pt) Which algorithm should be used for computing this new distribution?

5. (4 pts) Explain how this algorithm would arrive at the expected change in the distribution.

# 8 Search Algorithms (29 points)

1. What are to pros (if any) and cons (if any) to using an expanded list with Uniform Cost Search (as opposed to not keeping track of visited or expanded states)? Explain; consider both time and space.

2. What are the pros (if any) and cons (if any) to using A* versus Uniform Cost Search? Explain; consider both time and space.

3. What are the pros (if any) and cons (if any) to using Breadth First versus Progressive Deepening? Explain; consider both time and space.

4. If you take an admissible heuristic $h(s)$ with A* and modify it so that it returns $2h(s)$, what could be the impact on A*? Indicate possible positive and negative impacts.

5. Imagine that you are developing an algorithm to plan the motion of a mobile robot moving among obstacles. You have a map that indicates feasible intermediate locations between the start and the goal. Assume your map gives the coordinates of each location as well as which locations are reachable from which other locations. Each location has a position and also a "clearance", indicating how tight the passage is. Lower clearance will require the robot to slow down to go through there. So, we want to minimize a combination of the length of the path and the sum of (1/clearance) for every location along the path, for example, $\sum LinkLength + \alpha \sum \frac{1}{clearance}$, with $\alpha$ used to control the relative importance of the two measures. Give a (non-zero) admissible heuristic for this problem and discuss (briefly) its strengths and weaknesses.

6. You are given a graph corresponding to the structure of the complete World Wide Web. A node corresponds to a Web page and there are directed links connecting the nodes. If you want to find the shortest sequence of links from a start page to an end page, what search algorithm should you use? Explain.

7. When using the BT-FC algorithm for CSP problems you need to test any new assignment to a variable at the leaf of the search tree for consistency with previous assignments along the path. Is this True or False? Explain.

8. Briefly, explain why BT-FC is often more efficient than simple backtracking.

9. Circle TRUE or FALSE next to each statements below:

   (a) Alpha-Beta and MIN-MAX always compute the same score.
   **True** **False**

   (b) In the best case, on a fixed tree, Alpha-Beta cuts the running time of MIN-MAX in half.
   **True** **False**

   (c) Alpha-Beta computes the same score independent of the ordering of the leaves of the game tree.
   **True** **False**

   (d) The running time of Alpha-Beta is independent of the ordering of the leaves of the game tree.
   **True** **False**

   (e) A good chess-playing program will never need more than 13-ply look-ahead.
   **True** **False**

# 4 Constraints (20 points)

1. (8 pts) We have three variables: X, Y, and Z, each with the domain = $\{1, 2, 3, 4, 5\}$. The constraint between each pair of variables is that the sum of the variable assignments be odd. Assume you use BT-FC with the variable and value orderings are as given above. For each variable, how many tentative assignments would you need to try before discovering that it is impossible to find a satisfying solution to this problem.

   - Number of tentative assignments to X =
   - Number of tentative assignments to Y =
   - Number of tentative assignments to Z =

2. (3 pts) **True** or **False**: We could improve the efficiency of solving CSPs if we could develop a good heuristic function, so that we could use $A^*$ instead of depth-first search.

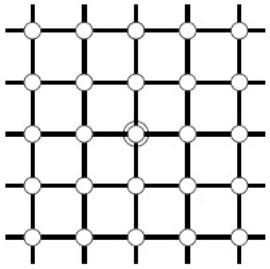
3. (3 pts) **True** or **False**: Full constraint propagation may not always find a unique solution to a CSP, but it will always tell you when a solution does not exist.

4. (3 pts) **True** or **False**: When doing backtracking with forward-checking, we never need to check the consistency of a new assignment with previous assignments.

5. (3 pts) **True** or **False**: Randomized hill-climbing methods such as GSAT can be viewed as a depth-first search in the space of partial assignments, in which the descendant of a search node is chosen randomly.

## 5 Search (35 points)

Consider the **unbounded**, i.e, extending infinitely, regular 2D grid state space shown below; the cost of each link is 1. The start state is the origin (in the center of the grid) and the goal state is at $(x, y)$. Assume, for simplicity, that $x \geq 0, y \geq 0$.



1. (1 pt) What is the branching factor $b$ in this state space?
2. (3 pts) How many **distinct states** are there at depth $k$ (for $k > 0$)? Circle all the true choices.
   (a) $4^k$
   (b) $4k$
   (c) $4k^2$
3. (3 pts) What is the length of the shortest path to the goal? And, is there unique such path?

4. (3 pts) Let $d$ be the length of the shortest path to the goal. Breadth-First search **without** an expanded or visited list expands (in the worst case) a number of nodes that grows as
   (a) $4^d$
   (b) $d$
   (c) $d^2$

   nodes before terminating. Circle all the true choices.
5. (3 pts) Let $d$ be the length of the shortest path to the goal. Breadth-First search **with** an expanded list expands (in the worst case) a number of nodes that grows as
   (a) $4^d$
   (b) $d$
   (c) $d^2$

   nodes before terminating. Circle all the true choices.

6. (3 pts) **True** or **False**: $h = |u - x| + |v - y|$ is an admissible heuristic for a state at $(u, v)$.

7. (3 pts) **True** or **False**: A* search with a strict expanded list using $h$ expands a number of nodes before terminating that grows linearly with $x + y$.

8. (3 pts) **True** or **False**: $h$ is admissible if some links are removed from the grid.

9. (3 pts) **True** or **False**: $h$ is admissible if some links are added between non-adjacent states.

10. (10 pts) Consider the following four search methods: depth-first (DF), breadth-first (BF), progressive deepening (PD), and uniform-cost (UC). Each of these methods can optionally be paired with an expanded list (EL). For each of these 8 methods (4 basic methods with and without EL), indicate those that **guarantee** the following properties on a grid state-space like above but **where the links have unequal costs**. You can give answers of the form "all except BF" or "all that use EL" or a list of methods "DF+EL, PD+EL" or "None".

    - Find a path to the goal
    - Find path with fewest states
    - Find minimal cost path
    - Visit each state at most once
    - Expand each state at most once

# 6.034 Quiz 1, Spring 2005

## 1 Search Algorithms (16 points)

### 1.1 Games

The standard alpha-beta algorithm performs a depth-first exploration (to a pre-specified depth) of the game tree.

1. Can alpha-beta be generalized to do a breadth-first exploration of the game tree and still get the optimal answer? Explain how or why not. If it can be generalized, indicate any advantages or disadvantages of using breadth-first search in this application.

2. Can alpha-beta be generalized to do a progressive-deepening exploration of the game tree and still get the optimal answer? Explain how or why not. If it can be generalized, indicate any advantages or disadvantages of using progressive-deepening search in this application.

## 1.2 Algorithms

1. You are faced with a path search problem with a very large branching factor, but where the answers always involve a relative short sequence of actions (whose exact length is unknown). All the actions have the same cost. What search algorithm would you use to find the optimal answer? Indicate under what conditions, if any, a visited or expanded list would be a good idea.

2. You are faced with a path search problem with a very large branching factor, but where the answers always involve a relative short sequence of actions (whose exact length is unknown). These actions, however, have widely varying costs. What search algorithm would you use to find the optimal answer? Indicate under what conditions, if any, a visited or expanded list would be a good idea.

## 2 Constraints (16 points)

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 |
---------
  |   |
```

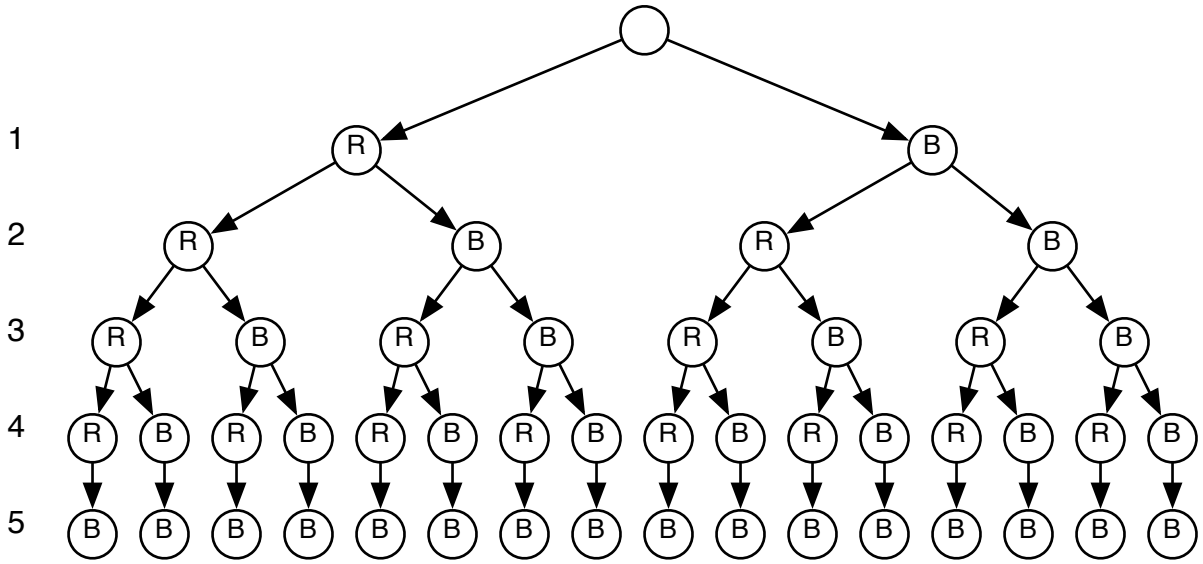
Let's look at the CSP formulation of this problem. Let the squares be the variables and the colors be the values. All the variables have domains { R, B }.

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

2. 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?

3. 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?

4. 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 check mark 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.




5. 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.

6. 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.

## 3 Constraint satisfaction (24 points)

You are trying to schedule observations on the space telescope. We have $m$ scientists who have each submitted a list of $n$ telescope observations they would like to make. An observation is specified by a target, a telescope instrument, and a time slot. Each scientist is working on a different project so the targets in each scientist's observations are different from those of other scientists. There are $k$ total time slots, and the telescope has three instruments, but all must be aimed at the same target at the same time.

The greedy scientists cannot all be satisfied, so we will try to find a schedule that satisfies the following constraints:

**C1**. Exactly two observations from each scientist's list will be made (the choice of the two will be part of the solution).

**C2**. At most one observation per instrument per time slot is scheduled.

**C3**. The observations scheduled for a single time slot must have the same target.

Note that for some set of requested observations, there may not be any consistent schedule, but that's fine.
Consider the following three formulations of the problem.

**A**. The variables are the $3k$ instrument/time slots.

**B**. The variables are the $m$ scientists.

**C**. The variables are the $mn$ scientists' requests.

For each formulation, specify

**1**. The value domain for the variables.

**2**. The size of the domain for the variables (in terms of $k$, $m$, and $n$).

**3**. Which of the constraints are necessarily satisfied because of the formulation.

**4**. Whether the constraints can be specified as binary constraints in this formulation. If they can, explain how. If not, provide a counterexample.

**Formulation A:** The variables are the $3k$ instrument/time slots.
1. Domain:

2. Size of domain:

3. Satisfied constraints:

4. Binary constraints?:

**Formulation B:** The variables are the $m$ scientists.
1. Domain:

2. Size of domain:

3. Satisfied constraints:

4. Binary constraints?:

**Formulation C:** The variables are the $mn$ scientists' requests.
1. Domain:

2. Size of domain:

3. Satisfied constraints:

4. Binary constraints?:

## 4 Search Problem formulation (23 points)

Consider a Mars rover that has to drive around the surface, collect rock samples, and return to the lander. We want to construct a plan for its exploration.

- It has batteries. The batteries can be charged by stopping and unfurling the solar collectors (pretend it's always daylight). One hour of solar collection results in one unit of battery charge. The batteries can hold a total of 10 units of charge.
- It can drive. It has a map at 10-meter resolution indicating how many units of battery charge and how much time (in hours) will be required to reach a suitable rock in each square.
- It can pick up a rock. This requires one unit of battery charge. The robot has a map at 10-meter resolution that indicates the type of rock expected in that location and the expected weight of rocks in that location. Assume only one type of rock and one size can be found in each square.

The objective for the rover is to get one of each of 10 types of rocks, within three days, while minimizing a combination of their total weight and the distance traveled. You are given a tradeoff parameter $\alpha$ that converts units of weight to units of distance. The rover starts at the lander with a full battery and must return to the lander.

Here is a list of variables that might be used to describe the rover's world:

- types of rocks already collected
- current rover location (square on map)
- current lander location (square on map)
- weight of rocks at current location (square on map)
- cost to traverse the current location (square on map)
- time since last charged
- time since departure from lander
- current day
- current battery charge level
- total battery capacity
- distance to lander
- total weight of currently collected rocks

1. Use a set of the variables above to describe the rover's state. Do not include extraneous information.

2. Specify the goal test.

3. Specify the actions. Indicate how they modify the state and any preconditions for being used.

4. Specify a function that determines the cost of each action.

5. This can be treated as a path search problem. We would like to find a heuristic. Say whether each of these possible heuristics would be useful in finding the optimal path or, if not, what's wrong with them. Let $l$ be the number of rocks already collected.

   **H1**: The sum of the distances (in the map) from the rover to the $10 - l$ closest locations for the missing types of rocks.

   **H2**: The length of the shortest tour through the $10 - l$ closest locations for the missing types of rocks.

   **H3**: The distance back to the lander.

## 5 Search traces (21 points)

Consider the graph shown in the figure below. We can search it with a variety of different algorithms, resulting in different search trees. Each of the trees (labeled G1 though G7) was generated by searching this graph, but with a different algorithm. Assume that children of a node are visited in alphabetical order. Each tree shows all the nodes that have been visited. Numbers next to nodes indicate the relevant "score" used by the algorithm for those nodes.

For each tree, indicate whether it was generated with

1. Depth first search
2. Breadth first search
3. Uniform cost search
4. A* search
5. Best-first (greedy) search

In all cases a strict expanded list was used. Furthermore, if you choose an algorithm that uses a heuristic function, say whether we used

**H1**: heuristic 1 = $\{h(A) = 3, h(B) = 6, h(C) = 4, h(D) = 3\}$

**H2**: heuristic 2 = $\{h(A) = 3, h(B) = 3, h(C) = 0, h(D) = 2\}$

Also, for all algorithms, say whether the result was an optimal path (measured by sum of link costs), and if not, why not. Be specific.

Write your answers in the space provided below (not on the figure).

**G1**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

**G2**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

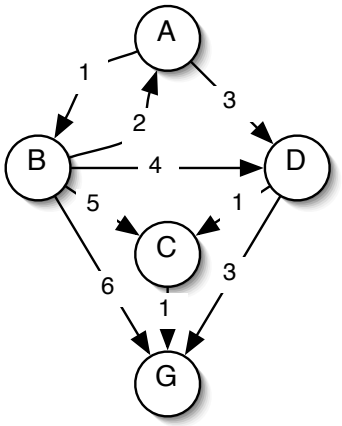
**G3**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

**G4**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

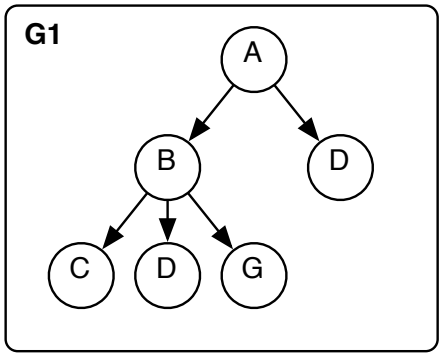
**G5**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

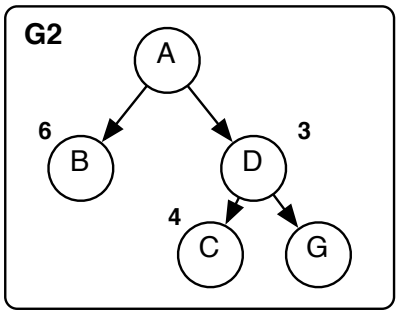
**G6**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

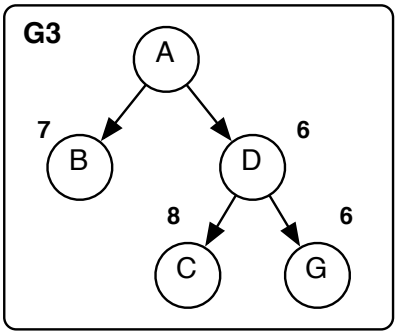
**G7**: 1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

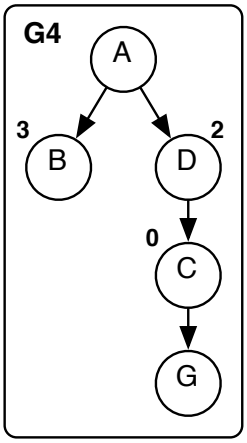


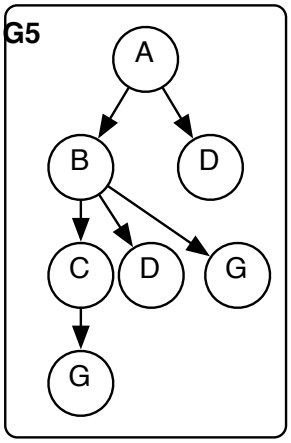

| | H1 | H2 |
|---|---|---|
| A | 3 | 3 |
| B | 6 | 3 |
| C | 4 | 0 |
| D | 3 | 2 |

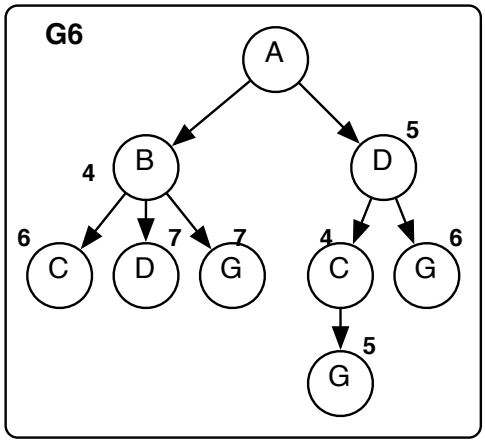


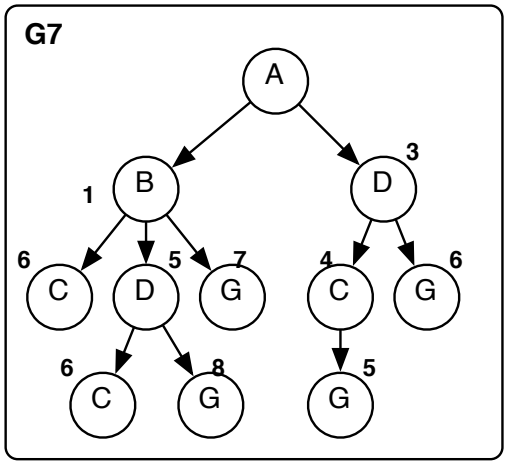



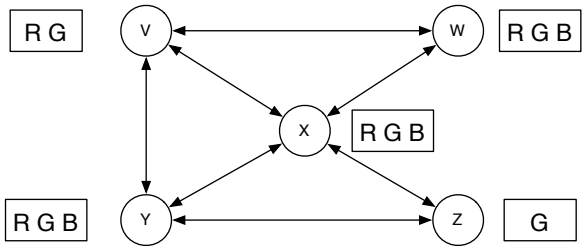



# 4 Constraint Satisfaction (20 points)

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.




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

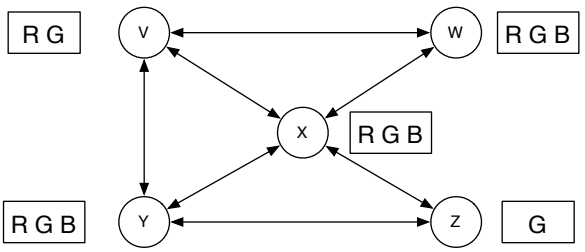
| Variable | Domain |
|---|---|
| V | |
| W | |
| X | |
| Y | |
| Z | |

2. What can you conclude about the existence of a solution based on this result? Explain.

3. 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 alphabetical order and the values are assigned in the order shown next to each node. Show assignments by writing the variable name and the value in the table below. Don't write more than 12 assignments, even if it would take more to find a consistent answer.

**Show only assignments that satisfy all the constraints up to and including that level of the backtracking search.**

| Step | Variable | Value | | Step | Variable | Value |
|---|---|---|---|---|---|---|
| 1 | | | ——— | 7 | | |
| 2 | | | ——— | 8 | | |
| 3 | | | ——— | 9 | | |
| 4 | | | ——— | 10 | | |
| 5 | | | ——— | 11 | | |
| 6 | | | ——— | 12 | | |




4. Show the sequence of variable assignments during backtracking with forward checking, assume that the variables are examined in alphabetical order and the values are assigned in the order shown next to each node. Show assignments by writing the variable name and the value in the table below. Don't write more than 12 assignments, even if it would take more to find a consistent answer.

**Show assignment before forward checking is performed.**

| Step | Variable | Value | | Step | Variable | Value |
|---|---|---|---|---|---|---|
| 1 | | | ——— | 7 | | |
| 2 | | | ——— | 8 | | |
| 3 | | | ——— | 9 | | |
| 4 | | | ——— | 10 | | |
| 5 | | | ——— | 11 | | |
| 6 | | | ——— | 12 | | |

## 5 Non-Binary Constraints I (20 pts)

We have limited our investigation of constraint satisfaction problems to those involving binary constraints. However, many constraints in the real world are not naturally binary. A simple example is finding satisfying assignments for a propositional formula in conjunctive normal form (CNF). An example of such a formula is:

$$(A \vee \neg B \vee C) \wedge (A \vee B \vee \neg C) \wedge (A \vee B \vee C)$$

1. Assuming that we pick the propositional variable in a CNF formula as the variables of a CSP, in general, what are the constraints?

2. Describe how this formulation can be transformed into a binary CSP formulation.

3. Illustrate your binary formulation of CNF satisfiability on the example sentence above. Make sure that you identify the variables, domains and constraints.

4. The transformation on a non-binary CSP problem into a binary one is an example of a general method. Outline briefly the key idea of this general method?

## 6 Non-Binary Constraints II (10 pts)

This problem is not limited to CNF satisfiability.

An alternative approach to dealing with non-binary constraints is to adapt our programs for solving CSP so as to be able to deal with non-binary constraints. For each of the following three methods for solving CSPs, outline what would need to be changed to handle constraints involving 3 variables. Think carefully, in some cases the changes are very minor. There may be more than one way to extend the methods, write down one.

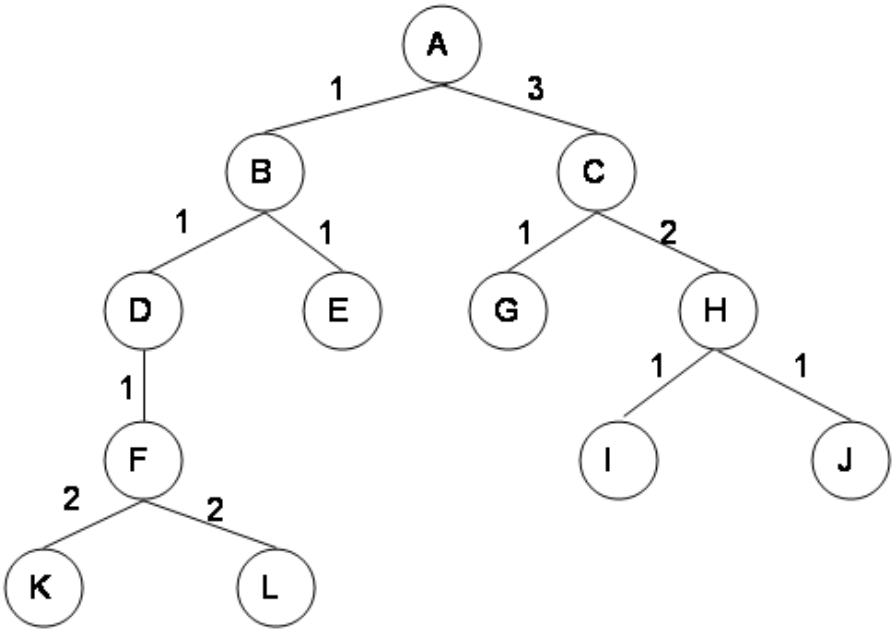
1. Min-Conflict Hill Climbing

2. Backtracking

3. Backtracking with Forward Checking

# 1 Tree Search (12 points)

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

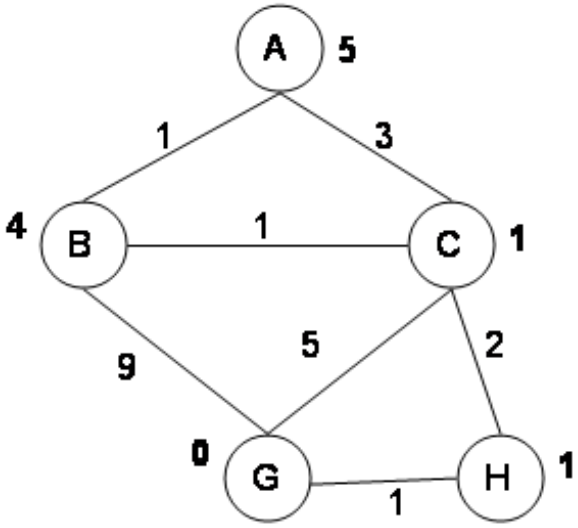



Assume that the nodes are expanded in alphabetical order when no other order is specified by the search, and that the goal is state $G$. No visited or expanded lists are used. What order would the states be expanded by each type of search? Stop when you expand G. Write only the sequence of states expanded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

# 4 Search problem formulation (10 points)

A Mars rover has to leave the lander, collect rock samples from three places (in any order) and return to the lander.

Assume that it has a navigation module that can take it directly from any place of interest to any other place of interest. So it has primitive actions *go-to-lander*, *go-to-rock-1*, *go-to-rock-2*, and *go-to-rock-3*.

We know the time it takes to traverse between each pair of special locations. Our goal is to find a sequence of actions that will perform this task in the shortest amount of time.

1. Formulate this problem as a search problem by specifying the state space, initial state, path-cost function, and goal test. Try to be sure that the state space is detailed enough to support solving the problem, but not redundant.

2. Say what search technique would be most appropriate, and why.

3. One possible heuristic evaluation function for a state would be the distance back to the lander from the location of the state; this is clearly admissible. What would be a more powerful, but still admissible, heuristic for this problem? (Don't worry about whether it's consistent or not.)

# 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?

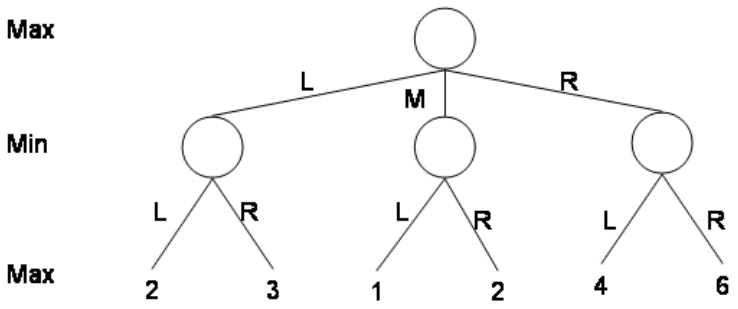
   - $J_1 \in$
   - $J_2 \in$
   - $J_3 \in$
   - $J_4 \in$

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.

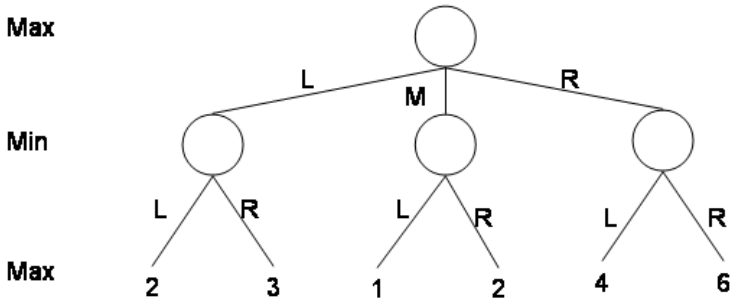



1. What is the value of the game to the max player?

2. What first move should the max player make?

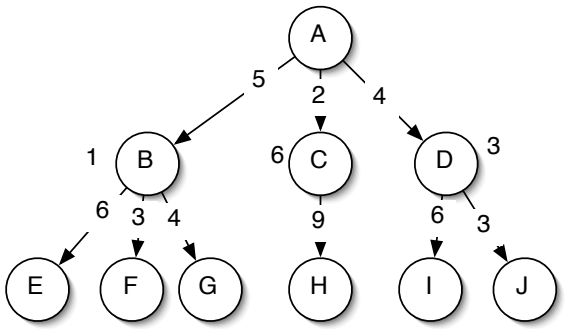
3. Assuming the max player makes that move, what is the best next move for the min player, assuming that this is the entire game tree?

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.

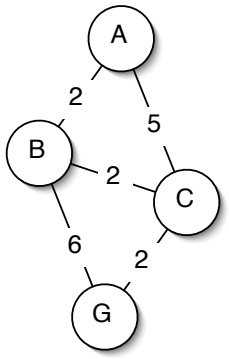



Assume that the children of a node are expanded in alphabetical order when no other order is specified by the search, and that the goal is state $J$. No visited or expanded lists are used. What order would the states be expanded by each type of search. Write only the sequence of states expanded by each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

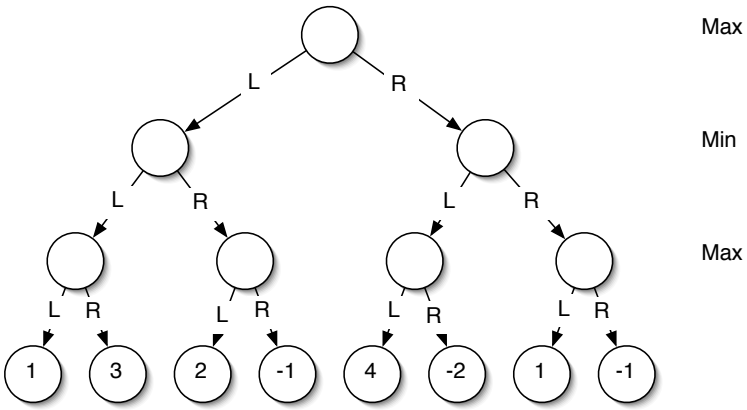
3. With the heuristic above will A* using a strict expanded list find the optimal path?

Yes No

Justify your answer very briefly.

# 4 Game Search (5 points)

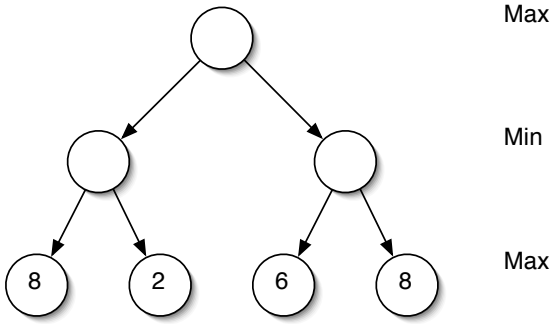
Consider the game tree shown below. Assume the top node is a max node. The labels on the arcs are the moves. The numbers in the bottom layer are the values of the different outcomes of the game to the max player.




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 refering 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).

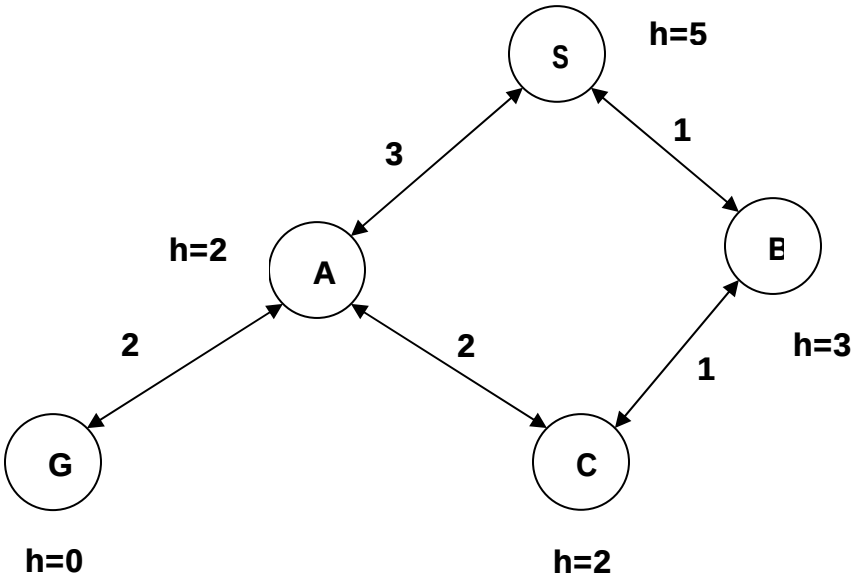
    1. Well-Defined? Yes No
    2. Useful? Yes No

- 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.**

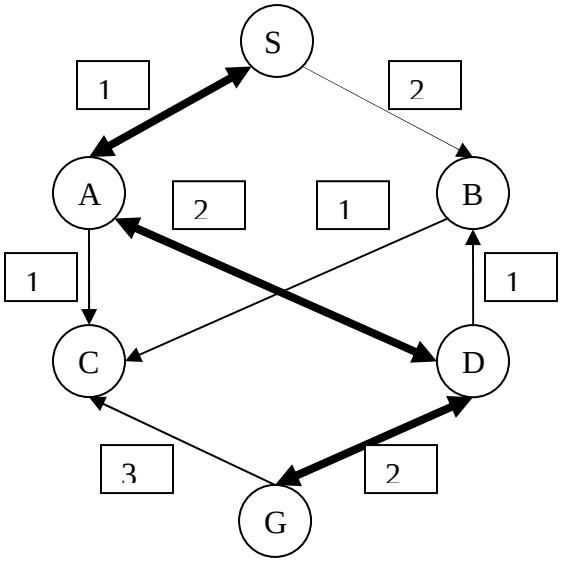
1. **We have a very large search space with a large branching factor and with possibly infinite paths. We have no heuristic. We want to find paths to the goal with minimum numbers of state.**

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.**

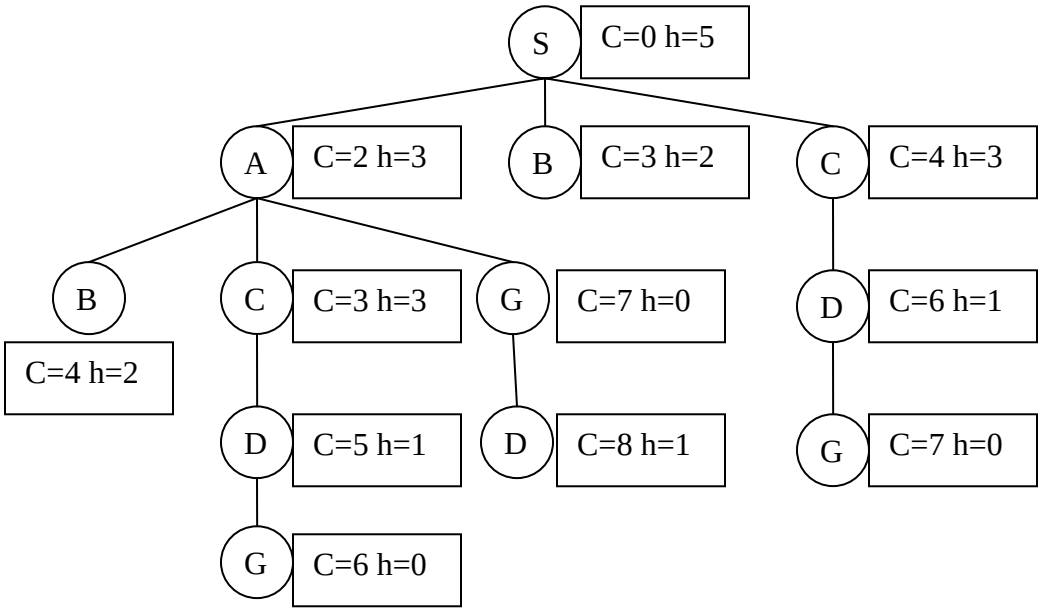
4. **We have a space with a manageable number of states but lots of cycles in the state graph. We have links of varying costs and an admissible heuristic and we want to find shortest paths.**

# 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**

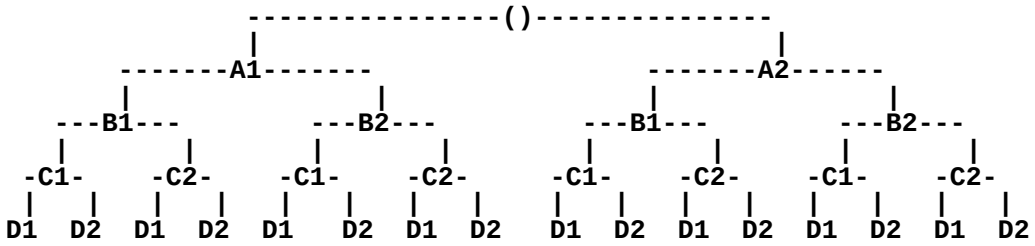
**An entry in the matrix below indicates a consistent assignment. This is simply another way of presenting the same information in the list above.**

| | 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?**

---

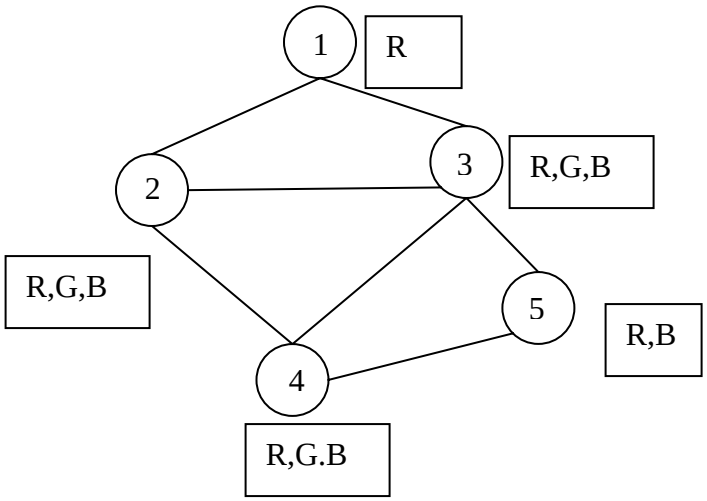
**The constraints – repeated for easy reference:**

- **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**

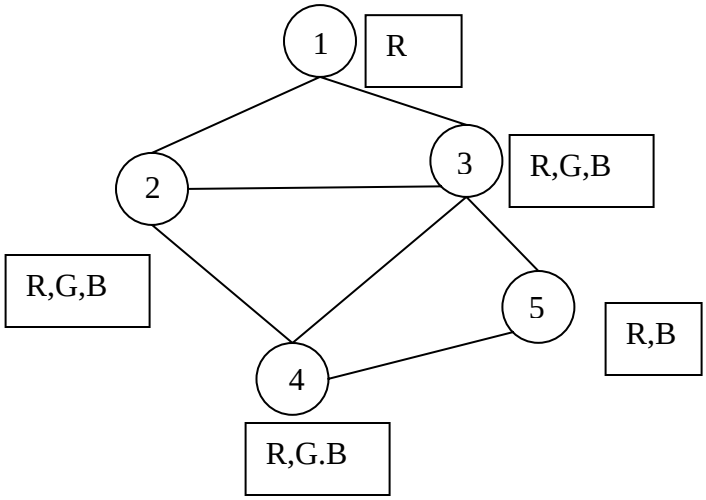
| | A1 | A2 | B1 | B2 | C1 | C2 | D1 | D2 |
|---|---|---|---|---|---|---|---|---|
| **A1** | | | X | | | X | | |
| **A2** | | | X | X | X | | | X |
| **B1** | X | X | | | | X | | |
| **B2** | | X | | | X | | | X |
| **C1** | | X | | X | | | X | X |
| **C2** | X | | X | | | | | |
| **D1** | | | | | X | | | |
| **D2** | | X | | X | X | | | |

# Problem 5: CSP (15 points)

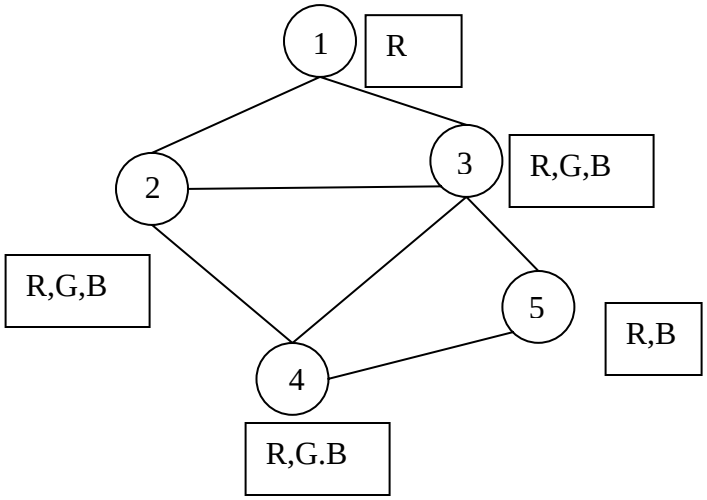
Consider the following constraint graph for a graph coloring problem (the constraints indicate that connected nodes cannot have the same color). The domains are shown in the boxes next to each variable node.




**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.