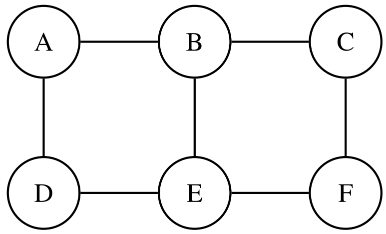
Q1. Worst-Case Backtracking

Consider solving the following CSP with standard backtracking search where we enforce arc consistency of all arcs before every variable assignment. Assume every variable in the CSP has a domain size $d > 1$.



(a) For each of the variable orderings, mark the variables for which backtracking search (with arc consistency checking) could end up considering more than one different value during the search.

(i) Ordering: A, B, C, D, E, F

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

(ii) Ordering: B, D, F, E, C, A

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

(b) Now assume that an adversary gets to observe which variable ordering you are using, and after doing so, chooses to add one additional binary constraint between any pair of variables in the CSP to maximize the number of backtracking variables in the worst case. For each of the following variable orderings, select which additional binary constraint the adversary should add. Then, mark the variables for which backtracking search (with arc consistency checking) could end up considering more than one different value when solving the modified CSP.

(i) Ordering: A, B, C, D, E, F
The adversary should add the additional binary constraint:

○ AC ○ AE ○ AF ○ BD
○ BF ○ CD ○ CE ○ DF

When solving the modified CSP with this ordering, backtracking might occur at:

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

(ii) Ordering: B, D, F, E, C, A
The adversary should add the additional binary constraint:

○ AC ○ AE ○ AF ○ BD
○ BF ○ CD ○ CE ○ DF

When solving the modified CSP with this ordering, backtracking might occur at:

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F

Q2. Satisfying Search

Consider a search problem $(S, A, Succ, s_0, G)$, where all actions have cost 1. S is the set of states, $A(s)$ is the set of legal actions from a state s, $Succ(s, a)$ is the state reached after taking action a in state s, s_0 is the start state, and $G(s)$ is true if and only if s is a goal state.

Suppose we have a search problem where we know that the solution cost is exactly k, but we do not know the actual solution. The search problems has $|S|$ states and a branching factor of b.

(a) **(i)** Since the costs are all 1, we decide to run breadth-first tree search. Give the tightest bound on the worst-case running time of breadth-first tree search in terms of $|S|$, b, and k.

The running time is $O($ ______________________________________ $)$

(ii) Unfortunately, we get an out of memory error when we try to use breadth first search. Which of the following algorithms is the best one to use instead?

- ◯ Depth First Search
- ◯ Depth First Search limited to depth k
- ◯ Iterative Deepening
- ◯ Uniform Cost Search

Instead of running a search algorithm to find the solution, we can phrase this as a CSP:

Variables: $X_0, X_1, X_2, \cdots X_k$

Domain of each variable: S, the set of all possible states

Constraints:

1. X_0 is the start state, that is, $X_0 = s_0$.
2. X_k must be a goal state, that is, $G(X_k)$ has to be true.
3. For every $0 \leq i < k$, (X_i, X_{i+1}) is an edge in the search graph, that is, there exists an action $a \in A(X_i)$ such that $X_{i+1} = Succ(X_i, a)$.

With these constraints, when we get a solution $(X_0 = s_0, X_1 = s_1, \cdots X_k = s_k)$, the solution to our original search problem is the path $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_k$.

(b) This is a tree-structured CSP. Illustrate this by drawing the constraint graph for $k = 3$ and providing a linearization order. (For $k = 3$, the states should be named X_0, X_1, X_2, and X_3.)

Constraint Graph:

Linearization Order: ____

(c) We can solve this CSP using the tree-structured CSP algorithm. You can make the following assumptions:

1. For any state s, computing $G(s)$ takes $O(1)$ time.
2. Checking consistency of *a single arc* $F \rightarrow G$ takes $O(fg)$ time, where f is the number of remaining values that F can take on and g is the number of remaining values that G can take on.

Remember that the search problem has a solution cost of exactly k, $|S|$ states, and a branching factor of b.

(i) Give the tightest bound on the time taken to enforce unary constraints, in terms of $|S|$, b, and k.

The running time to enforce unary constraints is $O($ ________________________________ $)$

(ii) Give the tightest bound on the time taken to run the backward pass, in terms of $|S|$, b, and k.

The running time for the backward pass is $O($ ________________________________ $)$

(iii) Give the tightest bound on the time taken to run the forward pass, in terms of $|S|$, b, and k.

The running time for the forward pass is $O($ ________________________________ $)$

(d) Suppose $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_k$ is a solution to the search problem. Mark all of the following options that are *guaranteed* to be true after enforcing unary constraints and running arc consistency.

☐ The remaining values of X_i will be s_i and possibly other values.
☐ The remaining values of X_i will be s_i and nothing else.
☐ A solution can be found by setting each X_i to any of the remaining states in its domain.
☐ A solution can be found by executing the forward pass of the tree-structured CSP algorithm.
☐ None of the above

(e) Suppose you have a heuristic $h(s)$. You decide to add more constraints to your CSP (with the hope that it speeds up the solver by eliminating many states quickly). Mark all of the following options that are valid constraints that can be added to the CSP, under the assumption that $h(s)$ is (a) any function (b) admissible and (c) consistent. *Recall that the cost of every action is* 1.

	Any $h(s)$	$h(s)$ is admissible	$h(s)$ is consistent
For every $0 \le i \le k$, $h(X_i) \le i$	☐	☐	☐
For every $0 \le i \le k$, $h(X_i) \le k - i$	☐	☐	☐
For every $0 \le i < k$, $h(X_{i+1}) \le h(X_i) - 1$	☐	☐	☐
For every $0 \le i < k$, $h(X_{i+1}) \ge h(X_i) - 1$	☐	☐	☐

☐ None of the above

(f) Now suppose we only know that the solution will have $\le k$ moves. We do not need to find the optimal solution - we only need to find some solution of cost $\le k$. Mark all of the following options such that if you make single change described in that line it will correctly modify the CSP to find some solution of cost $\le k$. *Remember, the CSP can only have unary and binary constraints.*

☐ Remove the constraints "(X_i, X_{i+1}) is an edge in the search graph". Instead, add the constraints "(X_i, X_{i+1}) is an edge in the search graph, OR $X_i = X_{i+1}$".
☐ Remove the constraints "(X_i, X_{i+1}) is an edge in the search graph". Instead, add the constraints "(X_i, X_{i+1}) is an edge in the search graph, AND $X_i = X_{i+1}$".
☐ Remove the constraint "X_k is a goal state." Instead, add the constraint "For every $0 \le i \le k$, X_i is a goal state".
Remove the constraint "X_k is a goal state." Instead, add the constraint "There is some i, $0 \le i \le k$, such that X_i is a goal state".
☐ None of the above