Course Name: Analysis and Design of Algorithms

Topics to be covered

- Hamiltonian Cycles

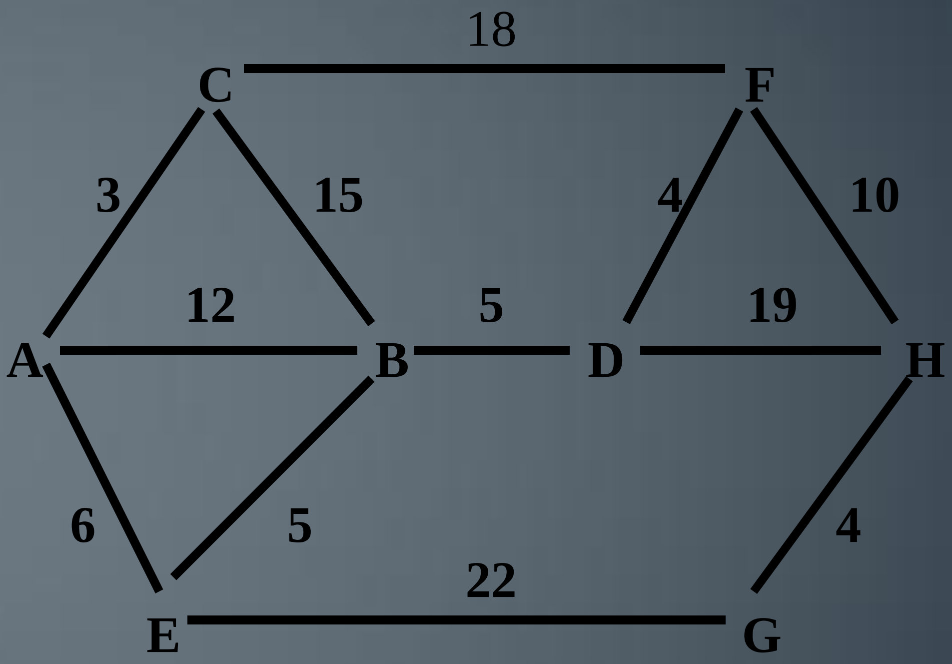
Hamiltonian Circuits Problem

- Hamiltonian circuit (tour) of a graph is a path that starts at a given vertex, visits each vertex in the graph exactly once, and ends at the starting vertex.

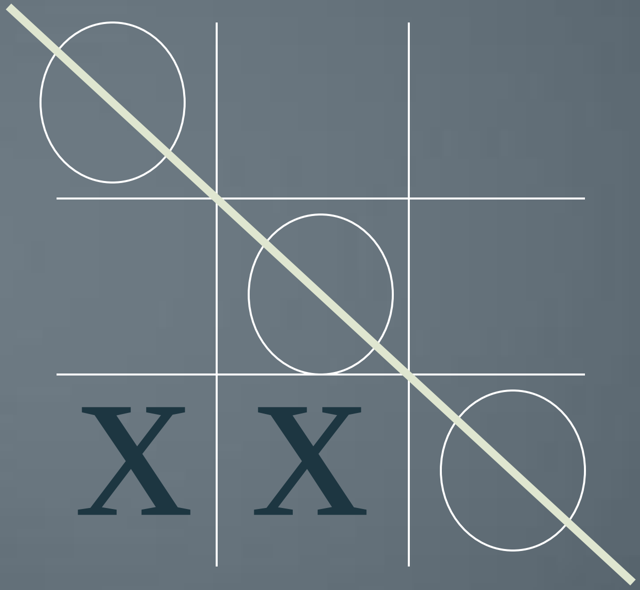
State Space Tree

- Put the starting vertex at level 0 in the tree
- At level 1, create a child node for the root node for each remaining vertex that is adjacent to the first vertex.
- At each node in level 2, create a child node for each of the adjacent vertices that are not in the path from the root to this vertex, and so on.

Example



Backtracking Algorithms

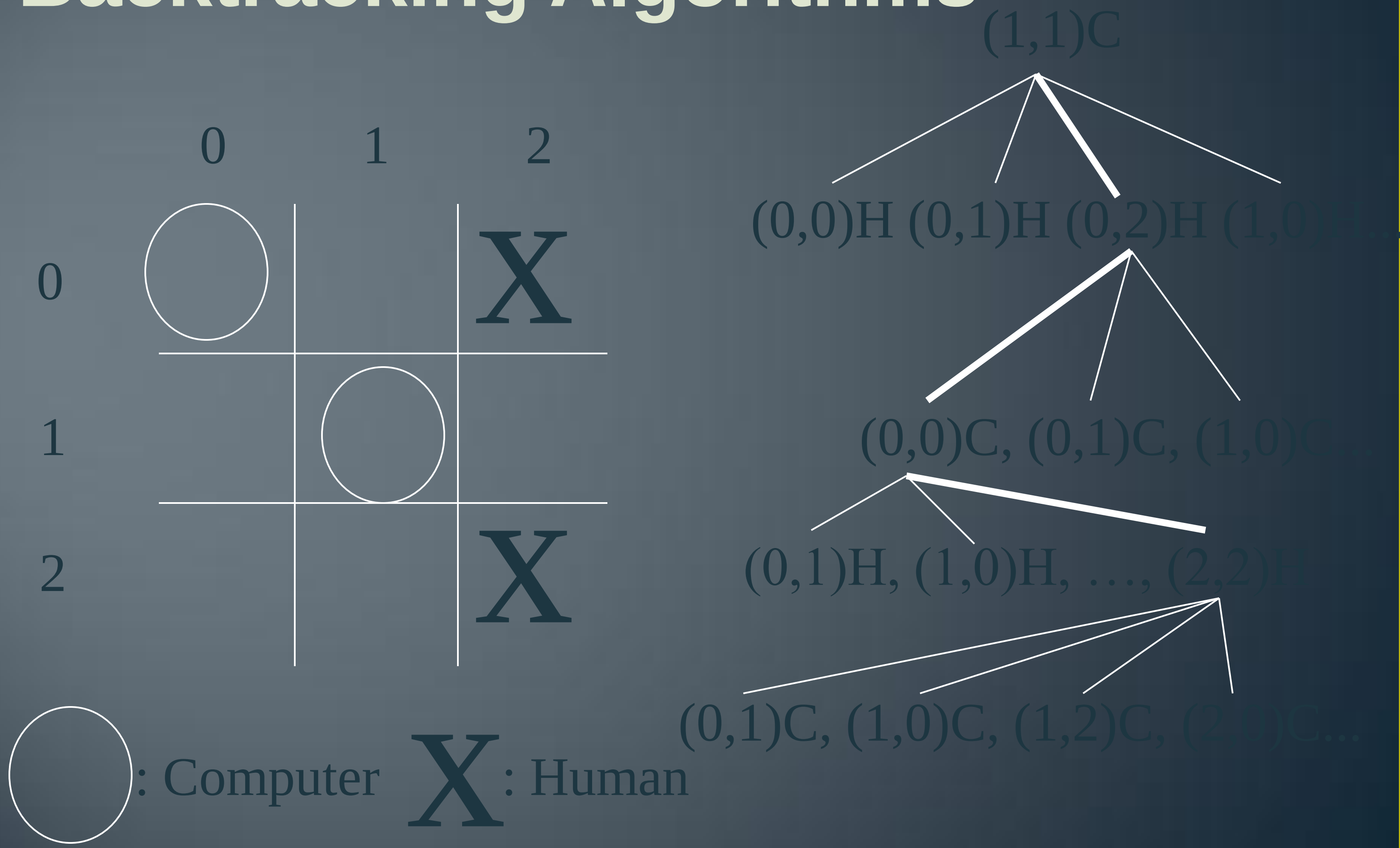


The tic-tac-toe game

How can a computer play the game?

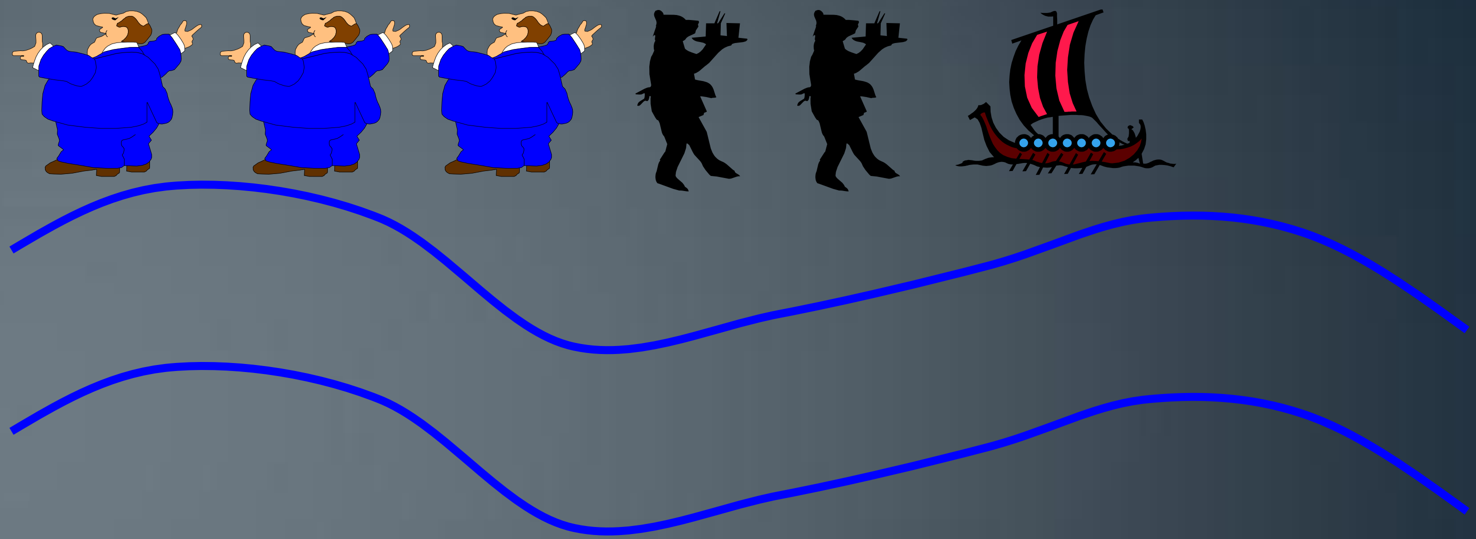
Remember Deep Blue?

Backtracking Algorithms



The tic-tac-toe game

Backtracking Algorithms



3 missionaries and 2 cannibals want to cross the river

Condition:

1. A boat can take one or two (must include a missionary)
2. At any time, on either bank, the number of missionaries must not be less than the number of cannibals.

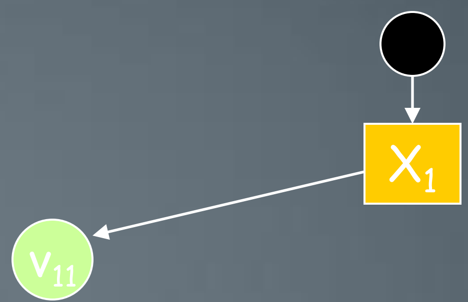
Backtracking Search

Essentially a simplified depth-first algorithm using recursion

Backtracking Search
(3 variables)

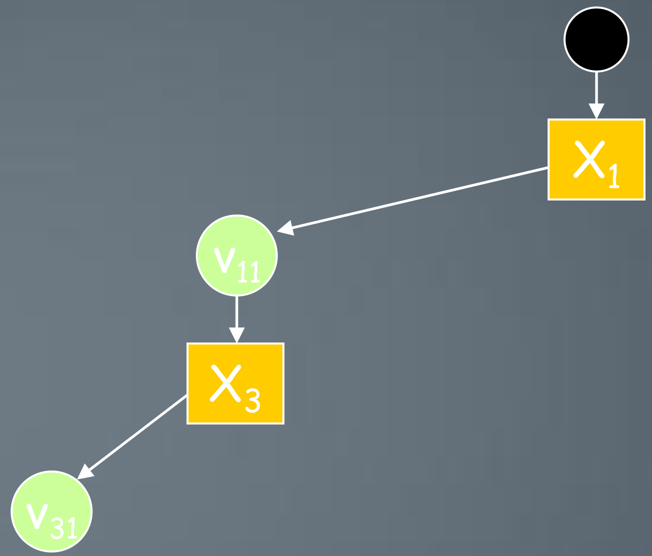
Assignment = {}

Backtracking Search
(3 variables)



Assignment = $\{(X_1, v_{11})\}$

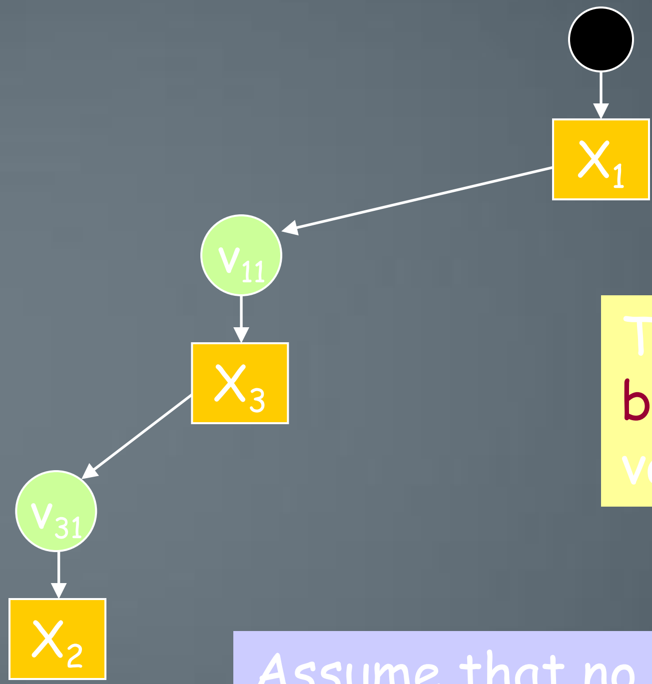
Backtracking Search
(3 variables)



Assignment = $\{(X_1,v_{11}), (X_3,v_{31})\}$

Backtracking Search

(3 variables)

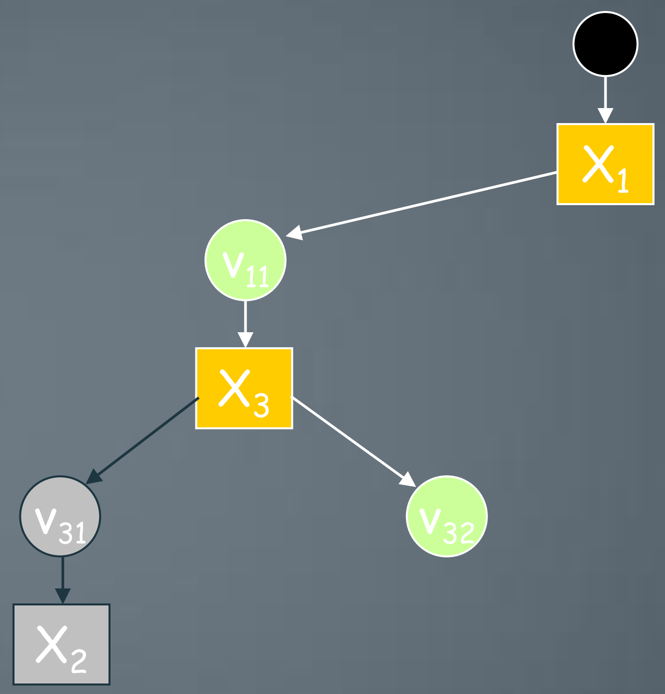


Then, the search algorithm **backtracks** to the previous variable and tries another value

Assume that no value of X_2 leads to a valid assignment

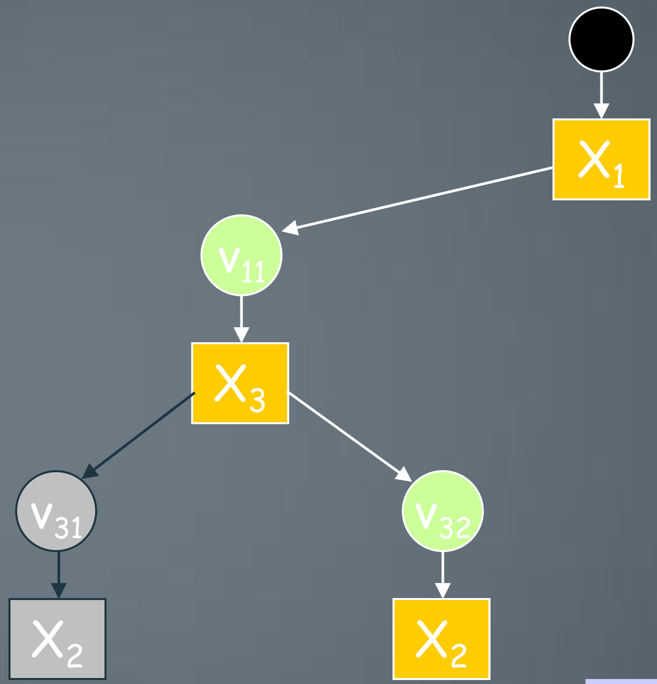
Assignment = $\{(X_1,v_{11}), (X_3,v_{31})\}$

Backtracking Search
(3 variables)



Assignment = $\{(X_1, v_{11}), (X_3, v_{32})\}$

Backtracking Search
(3 variables)

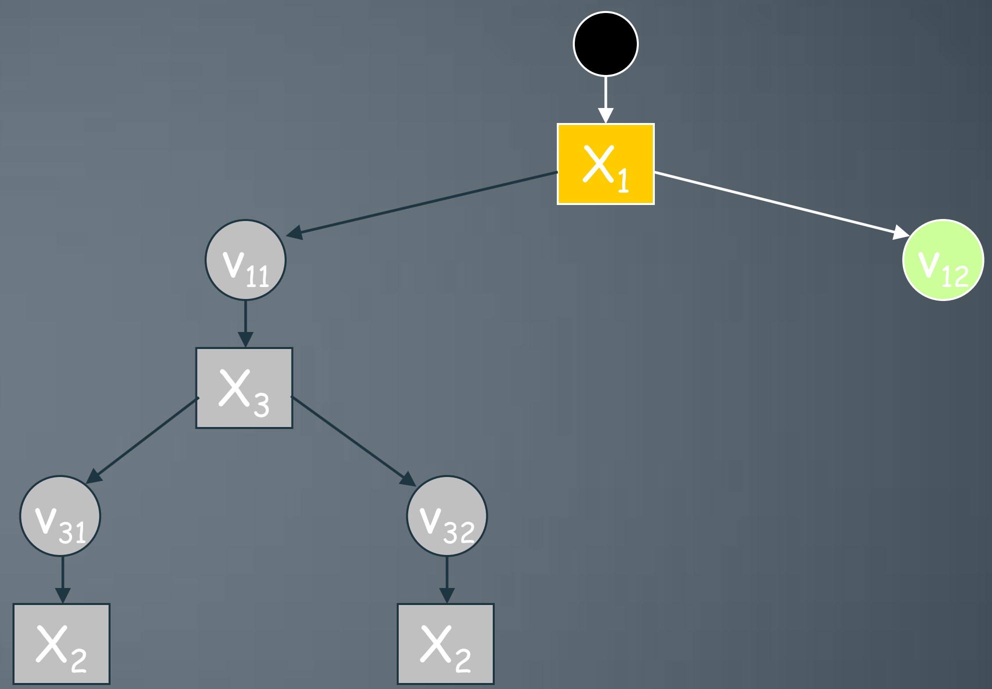


The search algorithm backtracks to the previous variable (X_3) and tries another value. But assume that X_3 has only two possible values. The algorithm backtracks to X_1

Assume again that no value of X_2 leads to a valid assignment

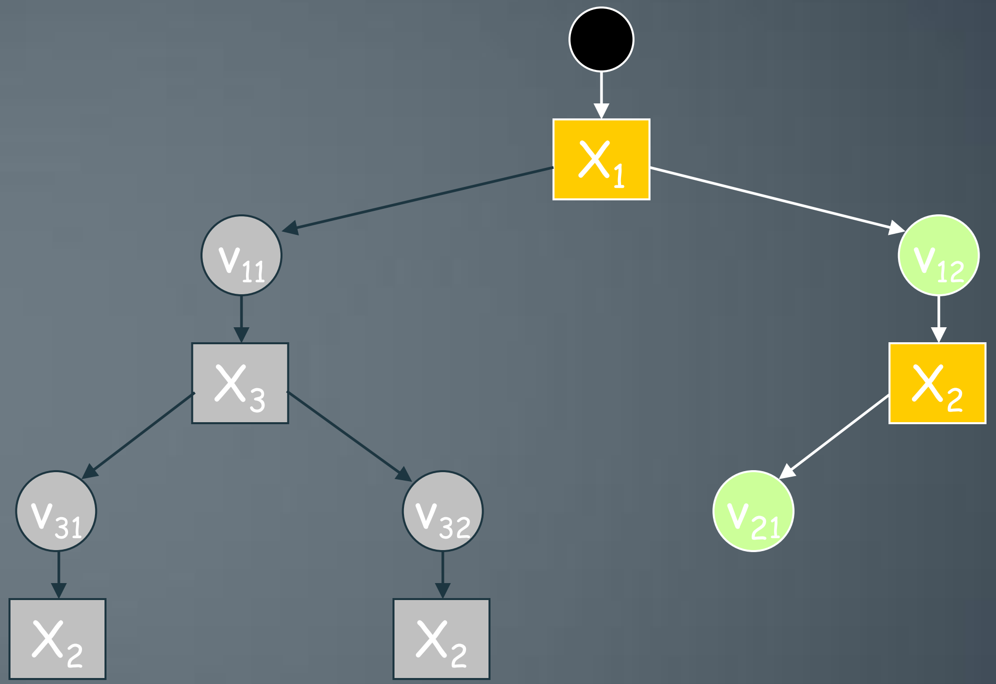
Assignment = $\{(X_1,v_{11}), (X_3,v_{32})\}$

Backtracking Search
(3 variables)



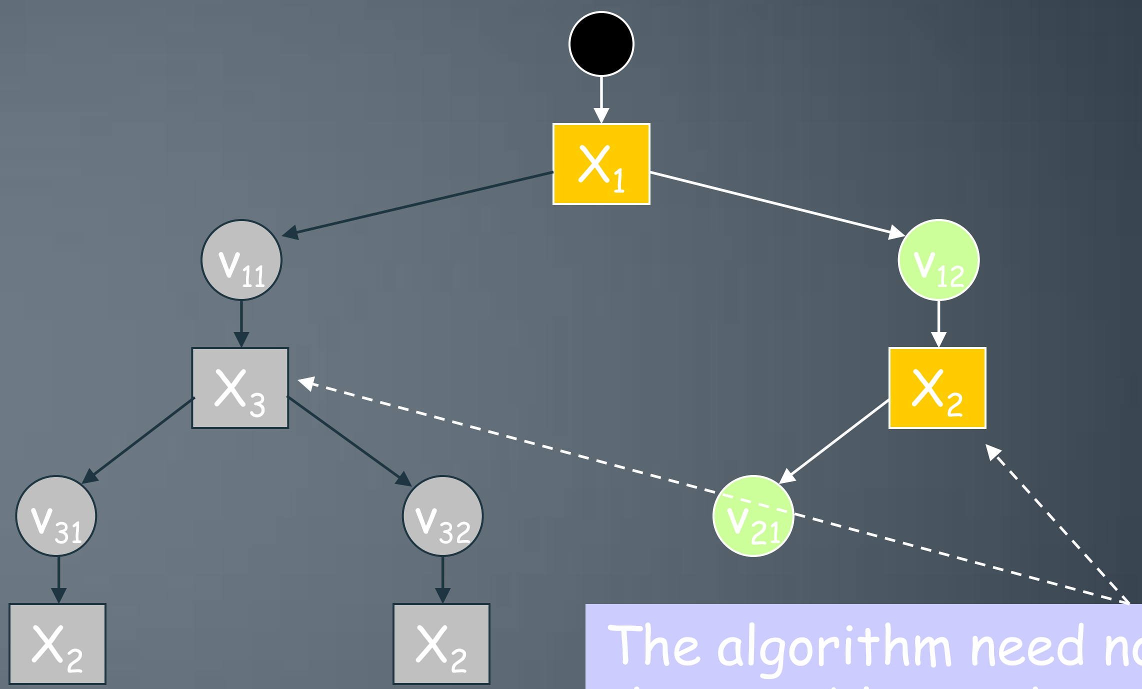
Assignment = $\{(X_1, v_{12})\}$

Backtracking Search
(3 variables)



Assignment = $\{(X_1, v_{12}), (X_2, v_{21})\}$

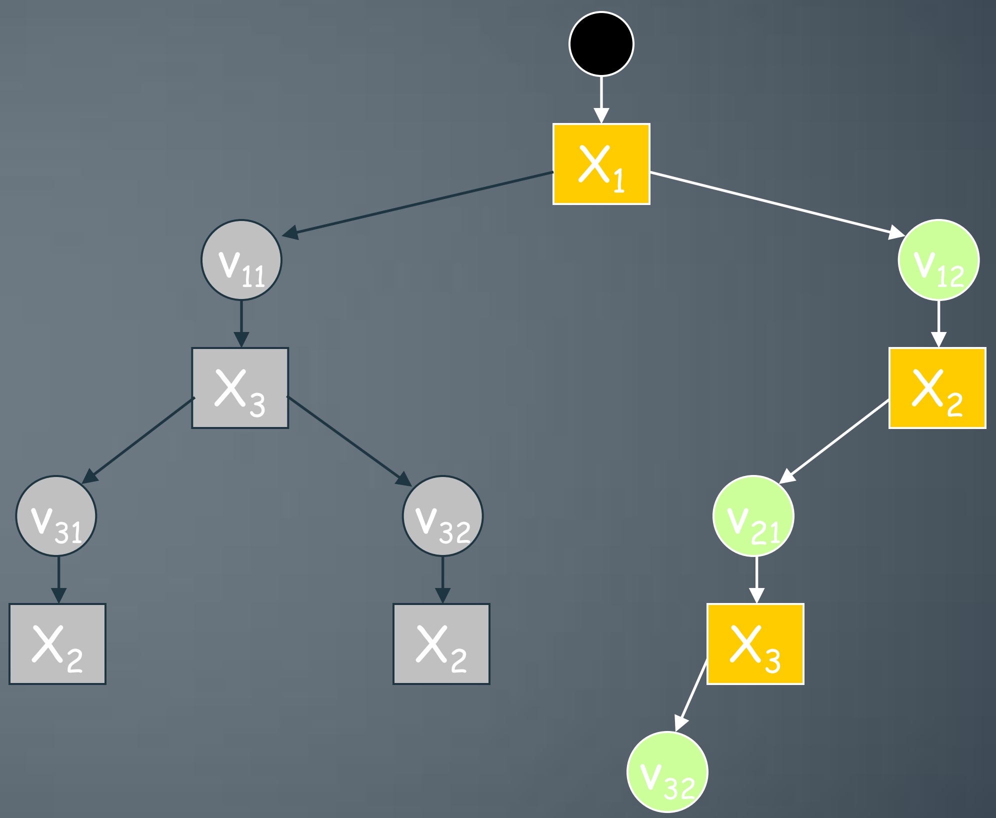
Backtracking Search
(3 variables)



The algorithm need not consider the variables in the same order in this sub-tree as in the other

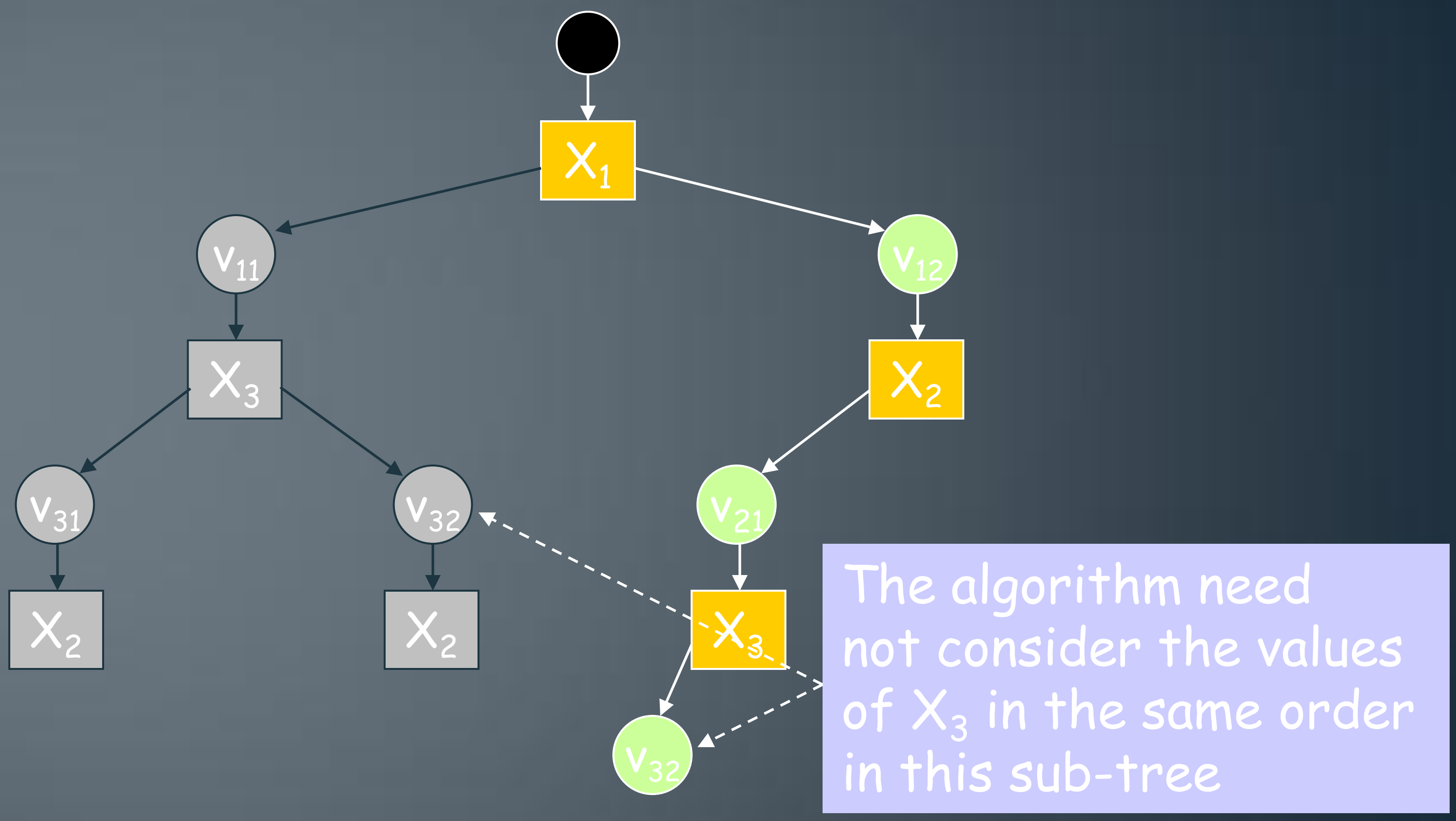
Assignment = $\{(X_1,v_{12}), (X_2,v_{21})\}$

Backtracking Search
(3 variables)



Assignment = $\{(X_1,v_{12}), (X_2,v_{21}), (X_3,v_{32})\}$

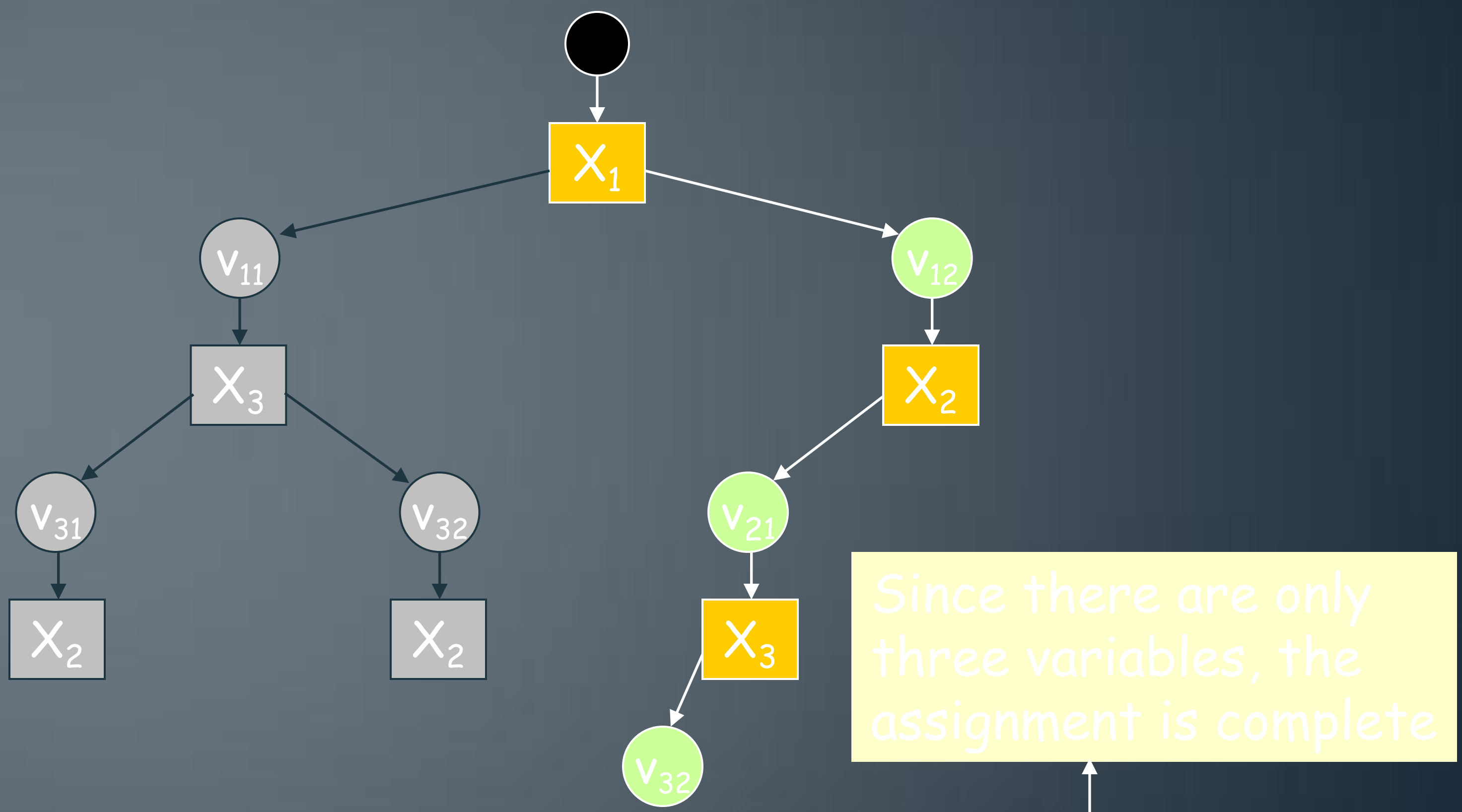
Backtracking Search
(3 variables)



Assignment = $\{(X_1,v_{12}), (X_2,v_{21}), (X_3,v_{32})\}$

Backtracking Search

(3 variables)



Assignment = $\{(X_1, v_{12}), (X_2, v_{21}), (X_3, v_{32})\}$

Backtracking Algorithm

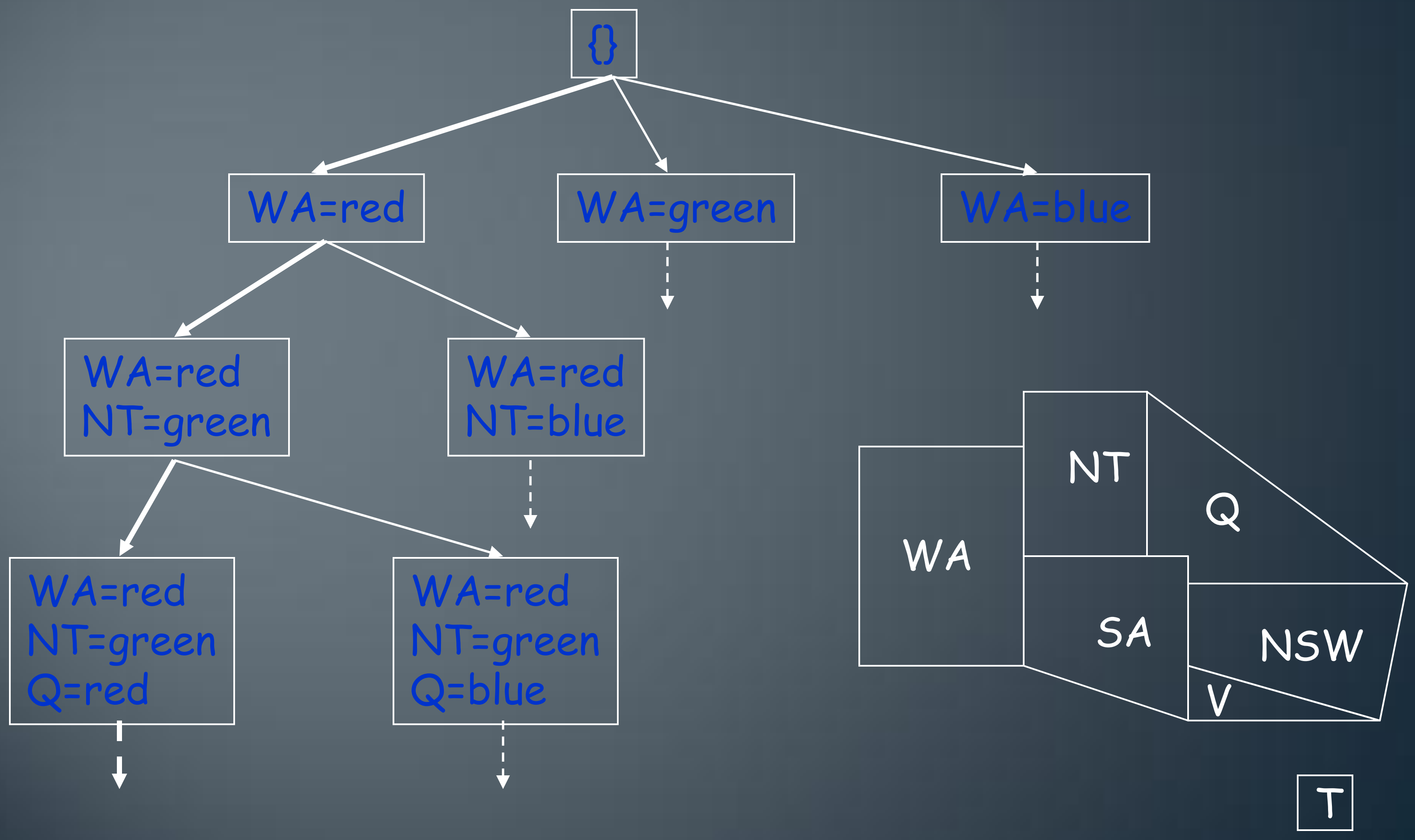
[This recursive algorithm keeps too much data in memory. An iterative version could save memory (left as an exercise)]

CSP-BACKTRACKING(A)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
    a. Add (X←v) to A
    b. If A is valid then
        i. result ← CSP-BACKTRACKING(A)
        ii. If result ≠ failure then return result
5. Return failure

Call CSP-BACKTRACKING({})

Map Coloring



Chapter Summary

- Backtracking is an algorithm design technique for solving problems in which the number of choices grows at least exponentially with their instant size.
- This approach makes it possible to solve many large instances of NP-hard problems in an acceptable amount of time.
- The technique constructs a pruned state space tree.
- Backtracking constructs its state-space tree in the depth-first search fashion in the majority of its applications.