

Design and Analysis of Algorithms

L42: Backtracking Algorithms Approach

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

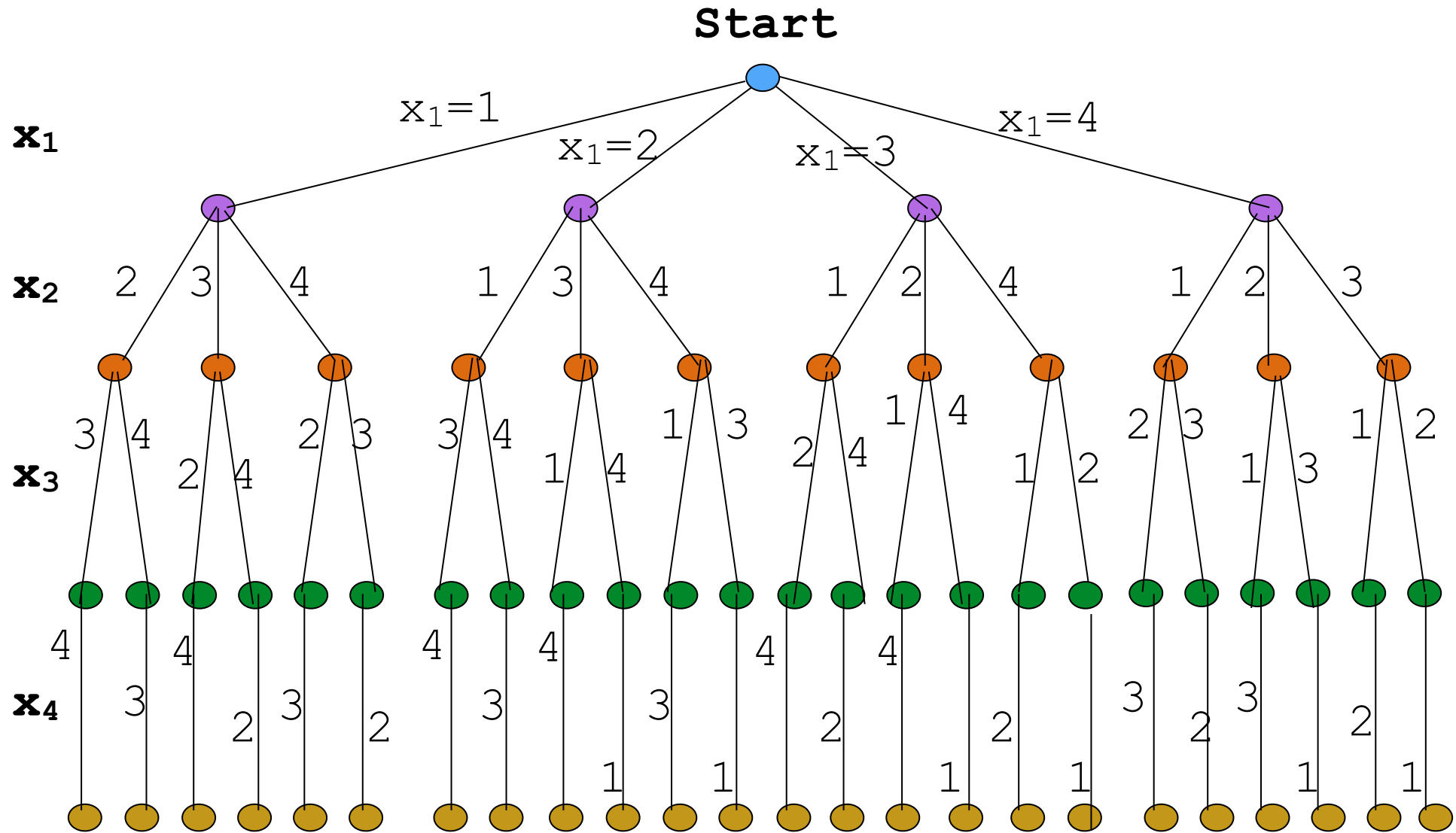
Resources

- Text book 2: Horowitz
 - Sec 7.1, 7.2, 7.3, 7.4, 7.5, 8.2, 11.1
- Text book 1: Levitin
 - Sec 12.1, 12.2
- RI: Introduction to Algorithms
 - Cormen et al.
- https://en.wikipedia.org/wiki/Dynamic_programming
- <https://www.codechef.com/wiki/tutorial-dynamic-programming>
- <https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-I/tutorial/>

Overview of Backtracking

- Basic approach of backtracking
 - Determine problem solutions by systematically searching the solution space
- Approach to search the solution space
 - Construct a tree of solution space
 - Node of this tree corresponds to a tuple variable assigned a possible feasible value
 - Edge of tree corresponds to tuple variables (x_i, x_{i+1}) where x_i is assigned a possible value
 - Leaf node satisfying the constraints (criterion function) represents a solution
- Two kind of trees
 - Static trees, Dynamic trees

State Space: 4 Queens



Size of 4-queens state space: $4! = 24$

DAA/Dynamic Programming

RPR/

4

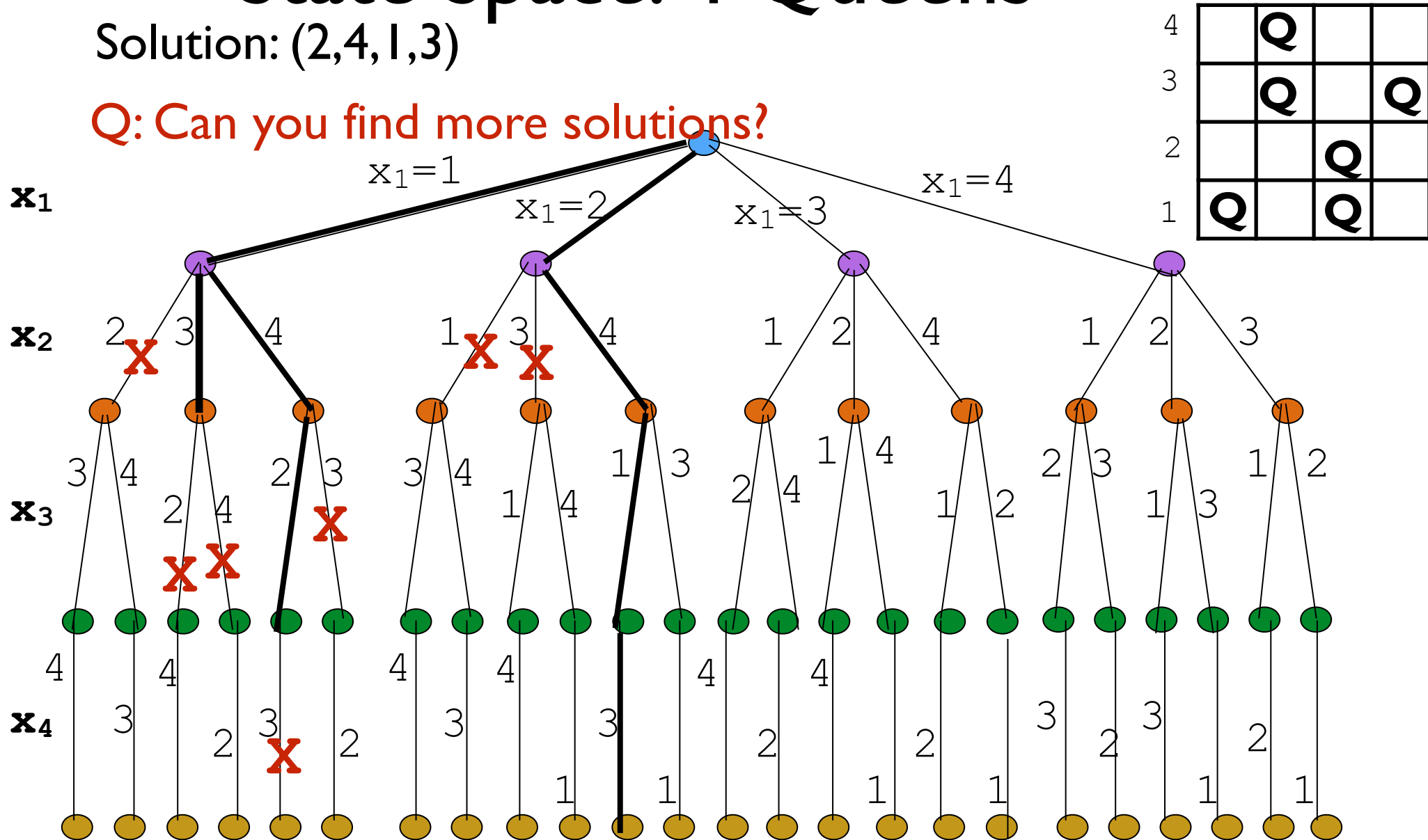
Solution: 4-Queens

4		Q₂		
3				Q₄
2	Q₁			
1			Q₃	
	a	b	c	d

State Space: 4 Queens

Solution: (2,4,1,3)

Q: Can you find more solutions?



Size of 4-queens state space: $4! = 24$

DAA/Dynamic Programming

RPR/

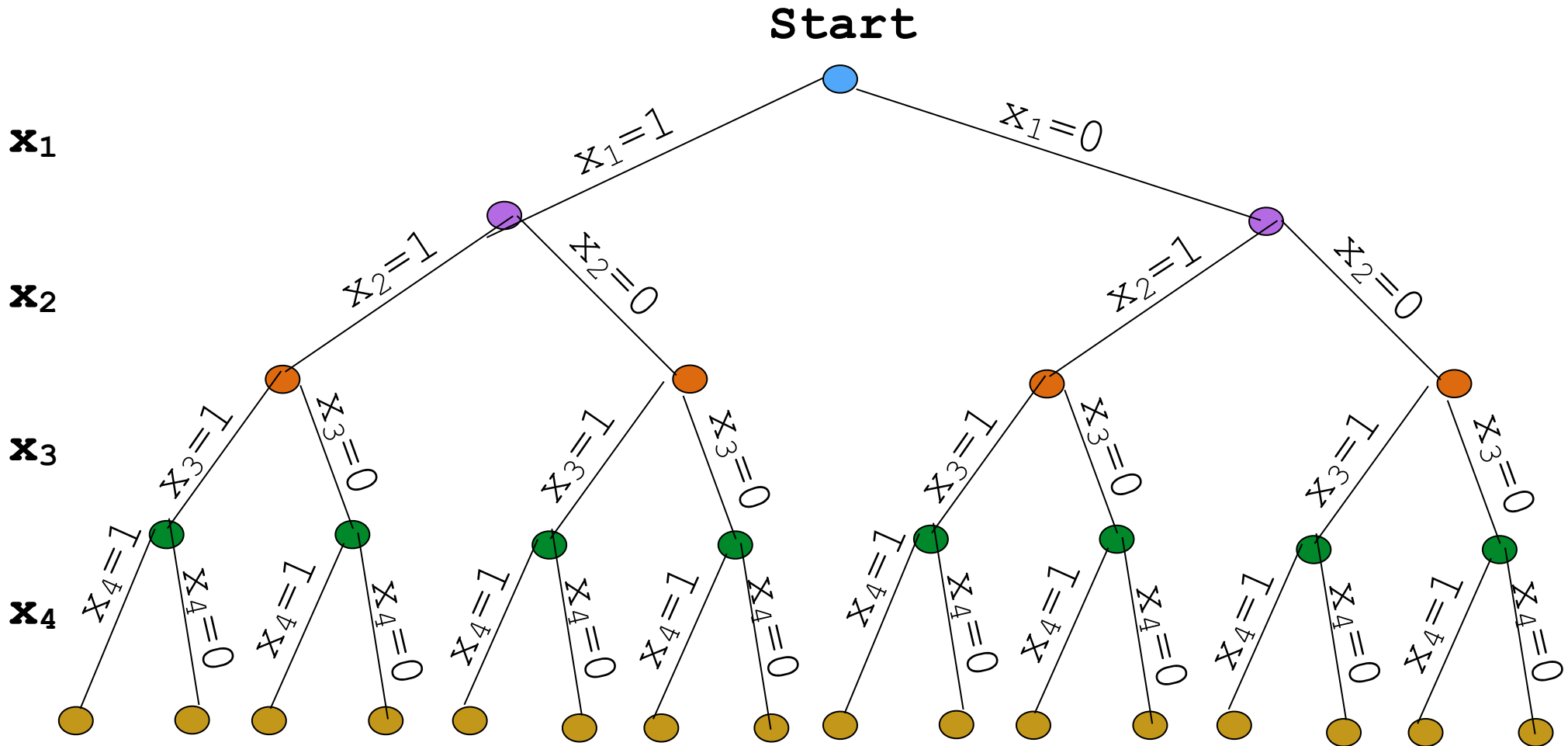
6

State Space: 4 Queens

- Solving 4-Queens problems
 - Build a complete possible tree
 - Called a static approach
 - Explore (traverse) the tree for possible solutions.
 - Prune the tree when come to a node where one can not traverse further
 - Backtrack to explore next path.
 - When leaf node is reached, a solution is found.
- Note: Building a static tree is independent of problem instance.
 - Tree is built for all possible solutions.

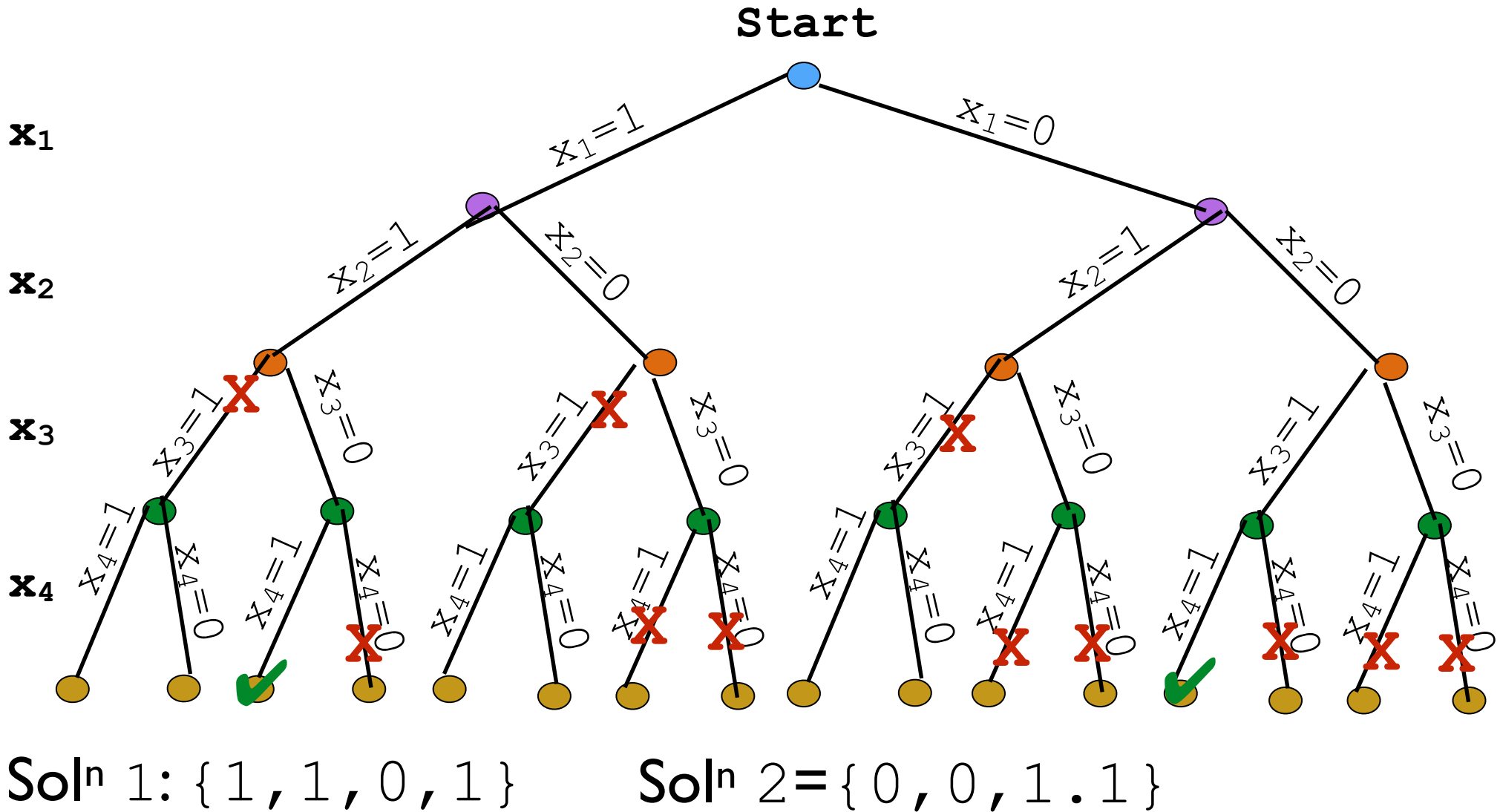
State Space: Sub of Subset problem

- **Ex:** $S = \{11, 13, 24, 7\}$, and $m = 31$



Solution Space: Subset sum problem

- Ex: $S = \{11, 13, 24, 7\}$, and $m = 31$



State Space: Subset Sum

- Solving sum of subset problems
 - Build a complete possible tree
 - Again a static approach
 - Explore (traverse) the tree for possible solutions.
 - Prune the tree on reaching a node where can not traverse further
 - Backtrack to explore next path.
 - When reach the leaf node, it is not necessary that a solution is found.
- Note: Building a static tree is independent of problem instance.
 - Tree is built for all possible solutions.

State Space Trees: Terminology

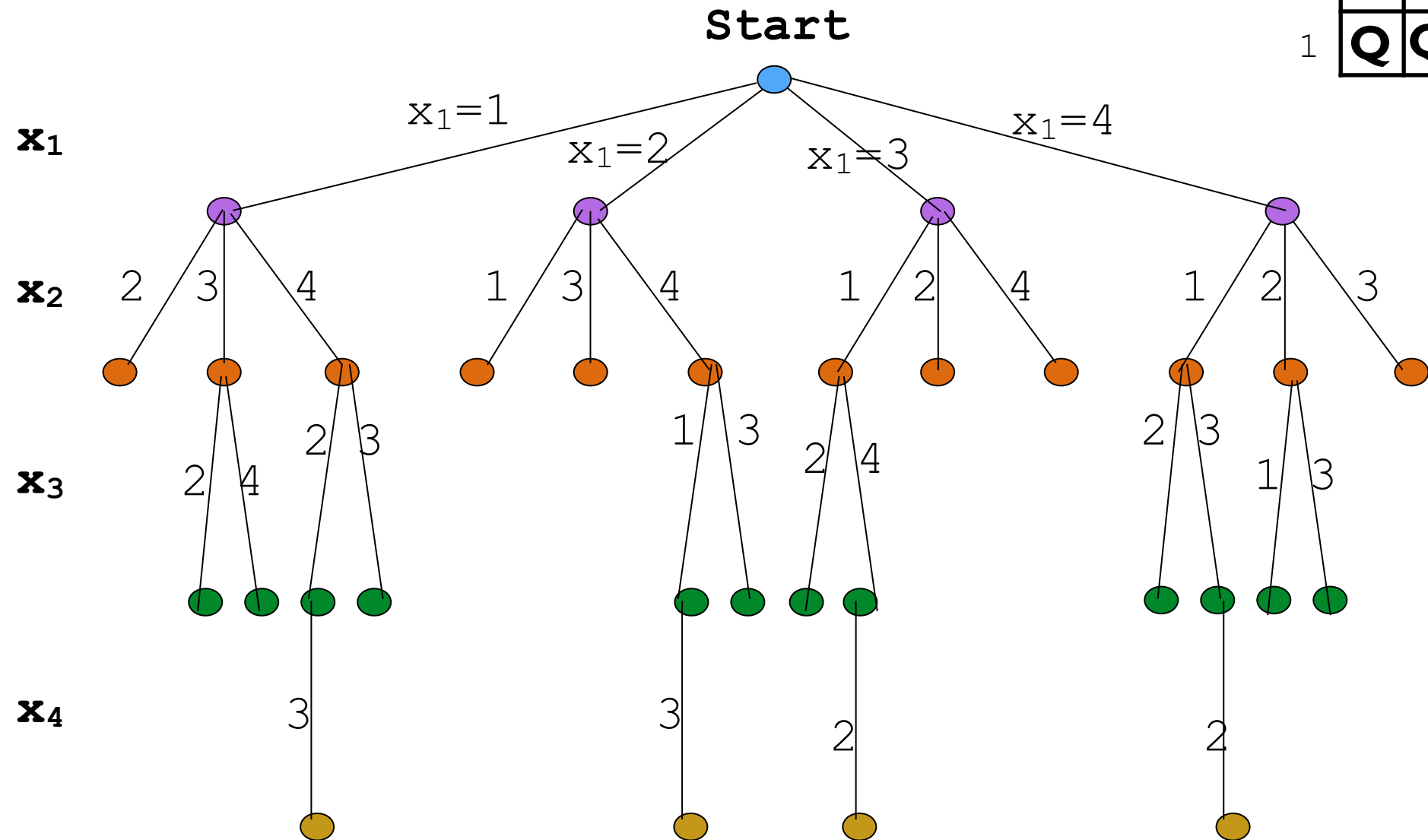
- Terminology
 - Each node in the tree defines a problem state
 - All paths from root to other nodes define state space of problem
 - Solution states are those problem states s for which the path from root to s defines a tuple in the solution space.
 - Answer states are those solution states s for which the path from root to s defines a tuple that is a member of the set of solutions.
 - Tree organization is referred to Space Tree.

State Space Trees

- Dynamic Trees:
 - Tree organization is determined dynamically as the state space is being searched.
 - Tree organizations that are dependent on problem instance are called dynamic trees.
- Two ways to generate problem states
 - Backtracking, and Branch-n-Bound
 - Both begin with the root and generate other nodes
 - A node whose all children have not been generated is called a live node
 - A dead node is a generated node which is not to be expanded further, or whose all children are generated.

Backtracking Tree: 4 Queens

4				
3				Q
2		Q	Q	
1	Q	Q	Q	



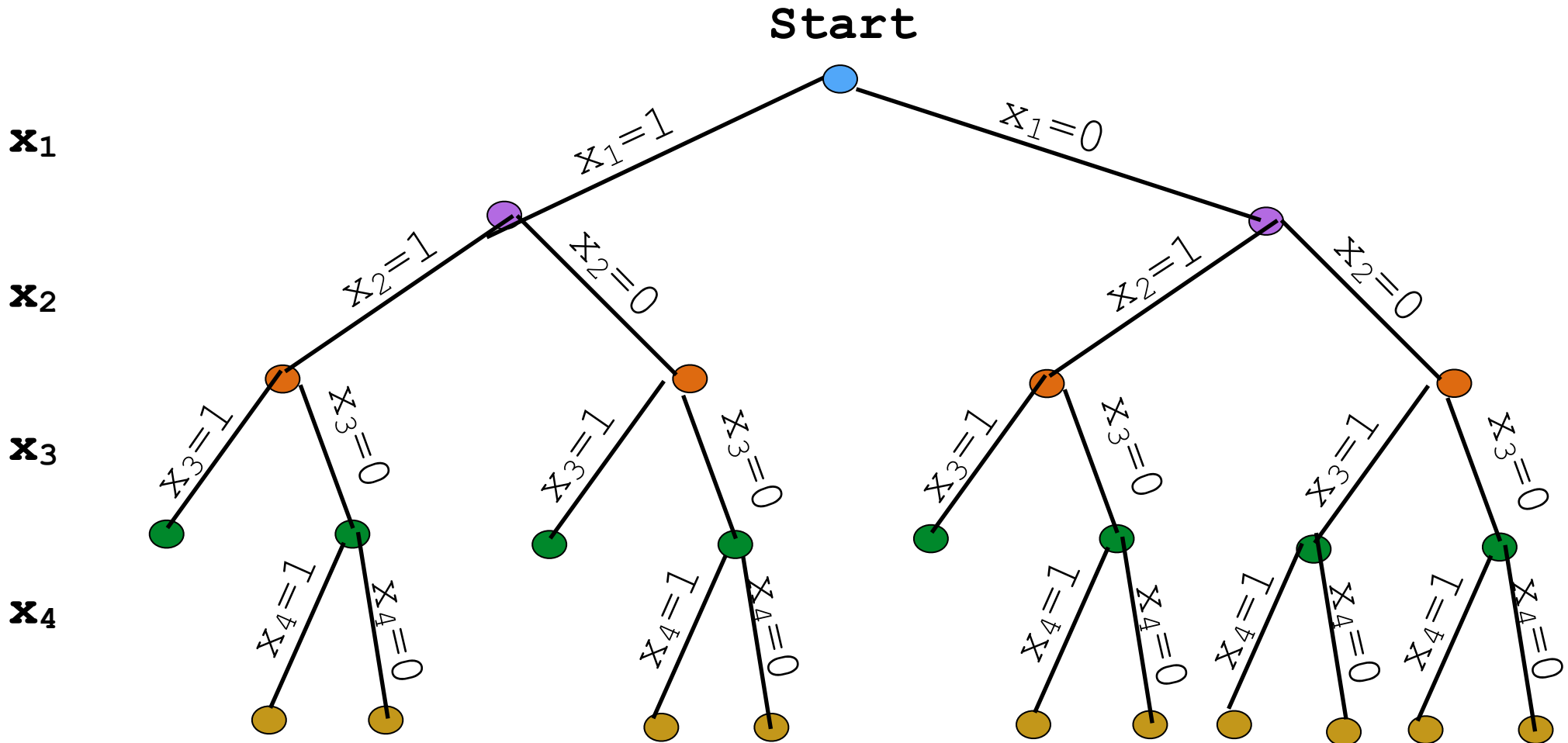
DAA/Dynamic Programming

RPR/

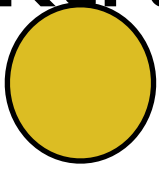
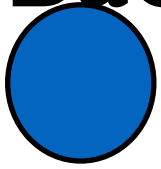
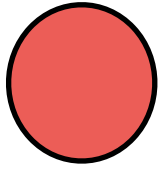
13

Backtracking Tree: Subset Sum

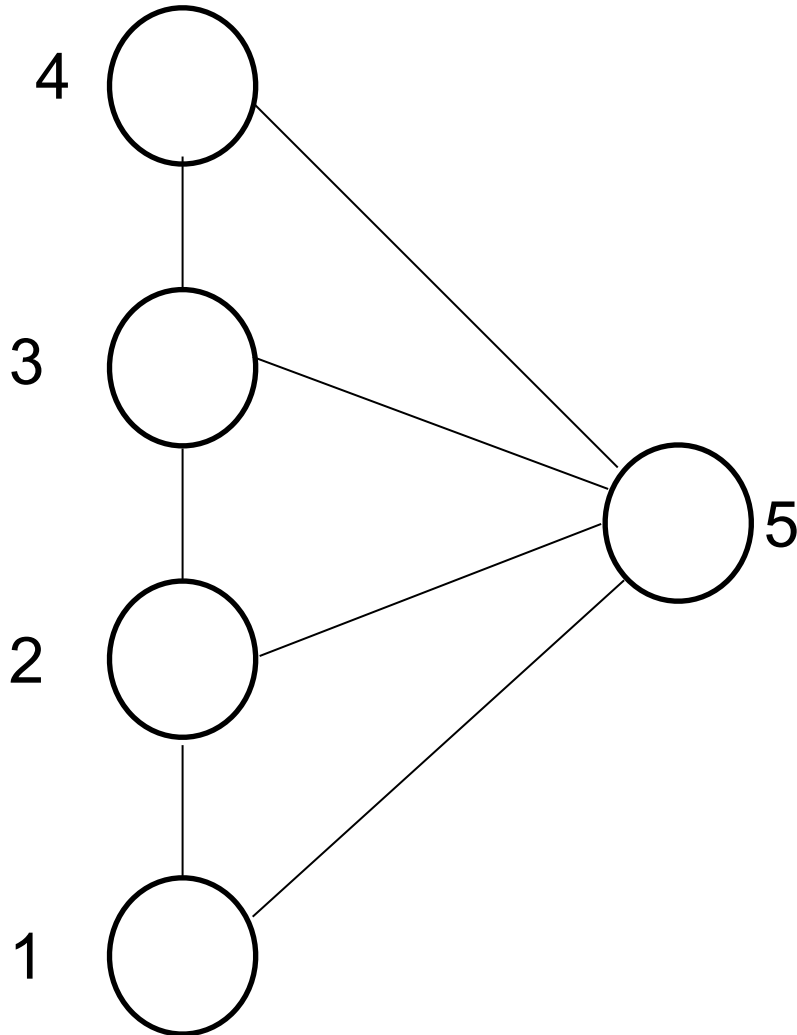
- Ex: $S = \{11, 13, 24, 7\}$, and $m = 31$



Backtracking: 3-color problem



- Exercise:
 - Construct the state space tree for the graph shown for 3-color problem.



Algo: Backtracking

- Notations:
 - (x_1, x_2, \dots, x_i) be the path from root to a node in state space tree.
 - $T(x_1, x_2, \dots, x_i)$ be the set of all possible values for x_{i+1} such that $(x_1, x_2, \dots, x_{i+1})$ is also a path to problem state
 - B_{i+1} is boundary function predicate i.e. if $B_{i+1}(x_1, x_2, \dots, x_{i+1})$ is false for path (x_1, \dots, x_{i+1}) , then path can't be extended to reach an answer state
 - $T(x_1, x_2, \dots, x_n) = \emptyset$
 - Candidates for position $i+1$ of solution vector (x_1, \dots, x_n)
 - Those values generated by T and satisfies B_{i+1}

Algo: Backtracking

Algo Backtrack (int n)

int k=1

while (k) {

if there remains an untried $x[k]$ such that

$x[k]$ is in $T(x[1], \dots, x[k-1])$, and

$B(x[1], \dots, x[k])$ is true, then {

if $(x[1], \dots, x[k])$ is a path to an answer node,

output $x[1:k]$

k++

}

k--

Summary

- Basic approach of backtracking
- Static tree
- Dynamic tree