UNIT 5: Backtracking

N-Queen’s problem

Subset sum problem

Backtracking

- coined by American mathematician **D. H. Lehmer** in the 1950s
- The pioneer string-processing language **SNOBOL** (1962) is the first to provide a built-in general backtracking facility
- Often make it possible to solve at least some large instances of difficult combinatorial problems.
- Is an improvement over exhaustive search (exhaustive-search technique generates all candidate solutions and then identifies the one (or the ones) with a desired property)

Backtracking

- A general algorithmic technique for solving some **computational problems** (**constraint satisfaction problems**) recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time

1. Decision Problem (search for a feasible solution)
2. Optimization Problem (search for the best solution)
3. Enumeration Problem (find all feasible solutions)

Backtracking Examples:

Backtracking can be used to solve puzzles or problems like:

- Puzzles such as eight queens **puzzle, crosswords, verbal arithmetic, Sudoku, and Peg Solitaire**.
- Combinatorial optimization problems such as **parsing** and the **knapsack problem**.
- Logic programming languages such as **Icon, Planner and Prolog**, which use backtracking internally to generate answers

Backtracking vs Recursion

- In recursion, the function calls itself until it reaches a base case.
- Backtracking is when the algorithm makes an opportunistic decision, which may come up to be wrong. If the decision was wrong then the backtracking algorithm restores the state before the decision. Thus recursion is used to explore all the possibilities until we get the best result for the problem

Note:

Backtracking can be implemented recursively or non-recursively

State - space tree

- A tree where nodes reflect specific choices made for a solution's components.
- Root represents an initial state before the search for a solution begins.
- Nodes of the first level in the tree represent the choices made for the first component of a solution,
- Nodes of the second level represent the choices for the second component, and so on

In backtracking a node is terminated as soon as it can be guaranteed that no solution to the problem can be obtained by considering choices that correspond to the node's descendants.

Backtracking Strategy/Idea:

construct solutions one component at a time and evaluate such partially constructed candidates as follows:

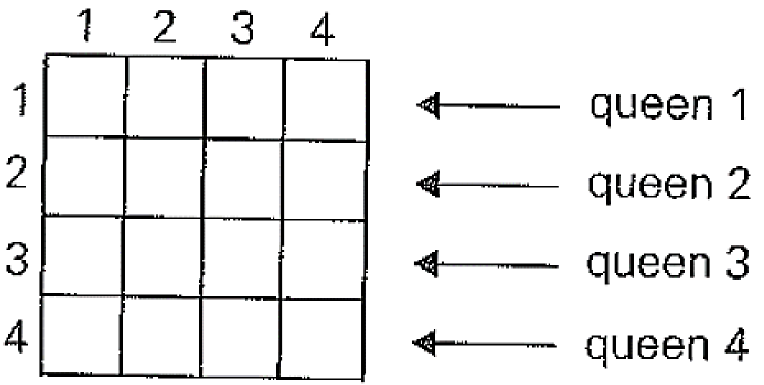
- If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component.
- If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered. In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option

N – Queen’s problem

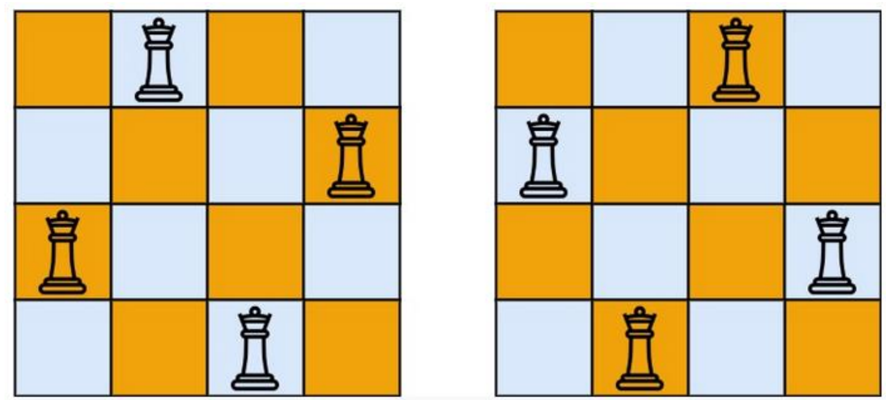
The problem is to place n queens on an n-by-n chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal

Note:
For n = 1, the problem has a trivial solution
For n = 2 and n = 3, the problem has no solution

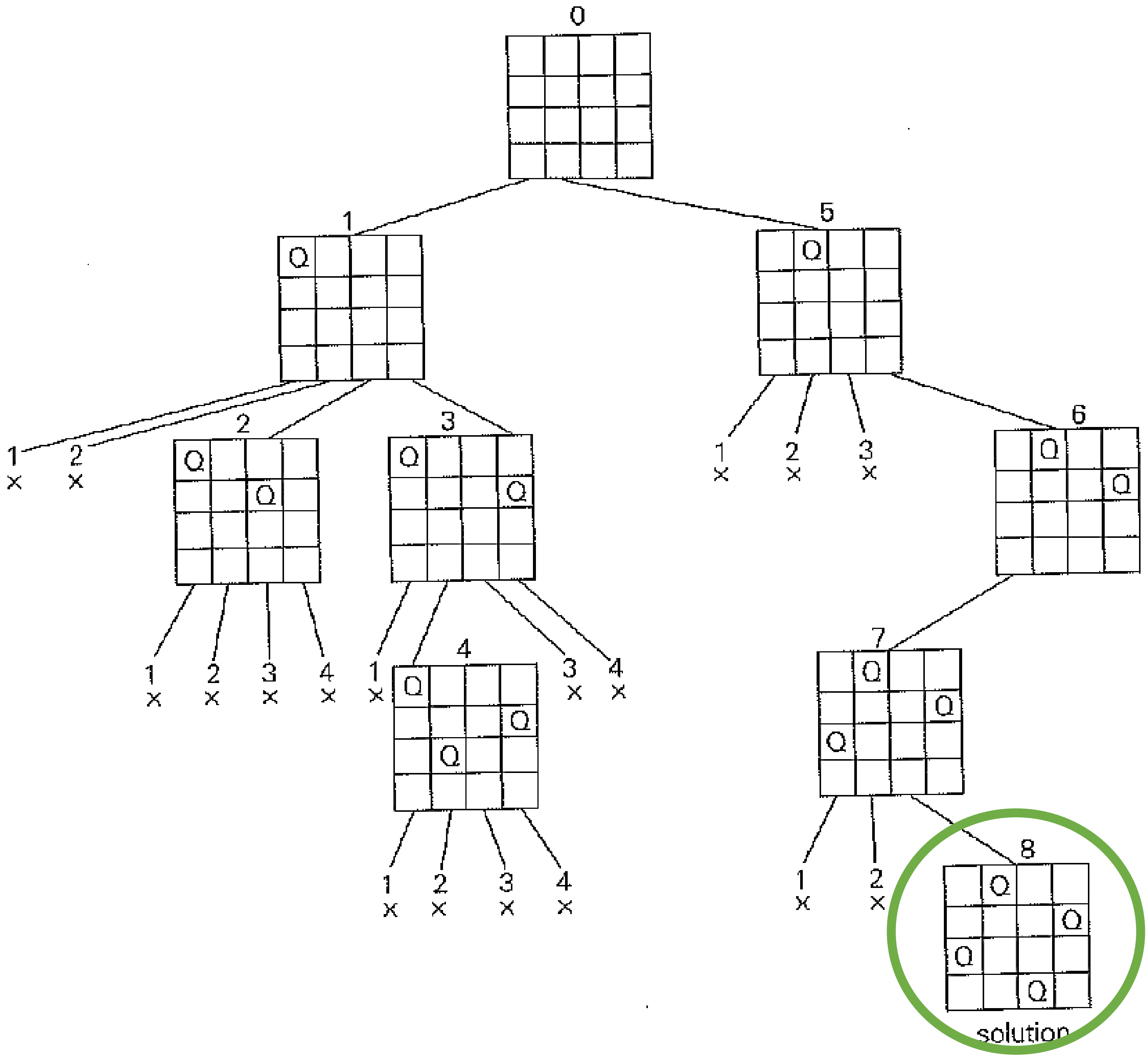
N – Queen's problem (n = 4)



Board for the four-queens problem



Since each of the four queens has to be placed in its own row, assign a column for each queen on the board presented



0
1
Q
5
Q
1
×
2
×
2
Q
Q
3
Q
Q
1
×
2
×
3
×
6
Q
Q
1
×
2
×
3
×
4
×
1
×
4
Q
Q
Q
3
×
4
×
7
Q
Q
Q
1
×
2
×
3
×
4
×
1
×
2
×
8
Q
Q
Q
Q
solution

N – Queen’s visualization

https://www.cs.usfca.edu/~galles/visualization/RecQueens.html

Subset sum problem

Find a subset of a given set S= {s1, ... , sn} of n positive integers whose sum is equal to a given positive integer d.

Example:

For S = {1, 2, 5, 6, 8} and d = 9,

there are two solutions: {1, 2, 6} and {1, 8}

Solution:

Assumptions:

- set contains non-negative values
- input set is unique (no duplicates present)

Exhaustive search approach: generates all possible subsets (size of such a power set is 2^N)

Recursive approach:

Dynamic programming approach:

Backtracking approach:

Solution:

Backtracking approach:

- sort the set's elements in increasing order:

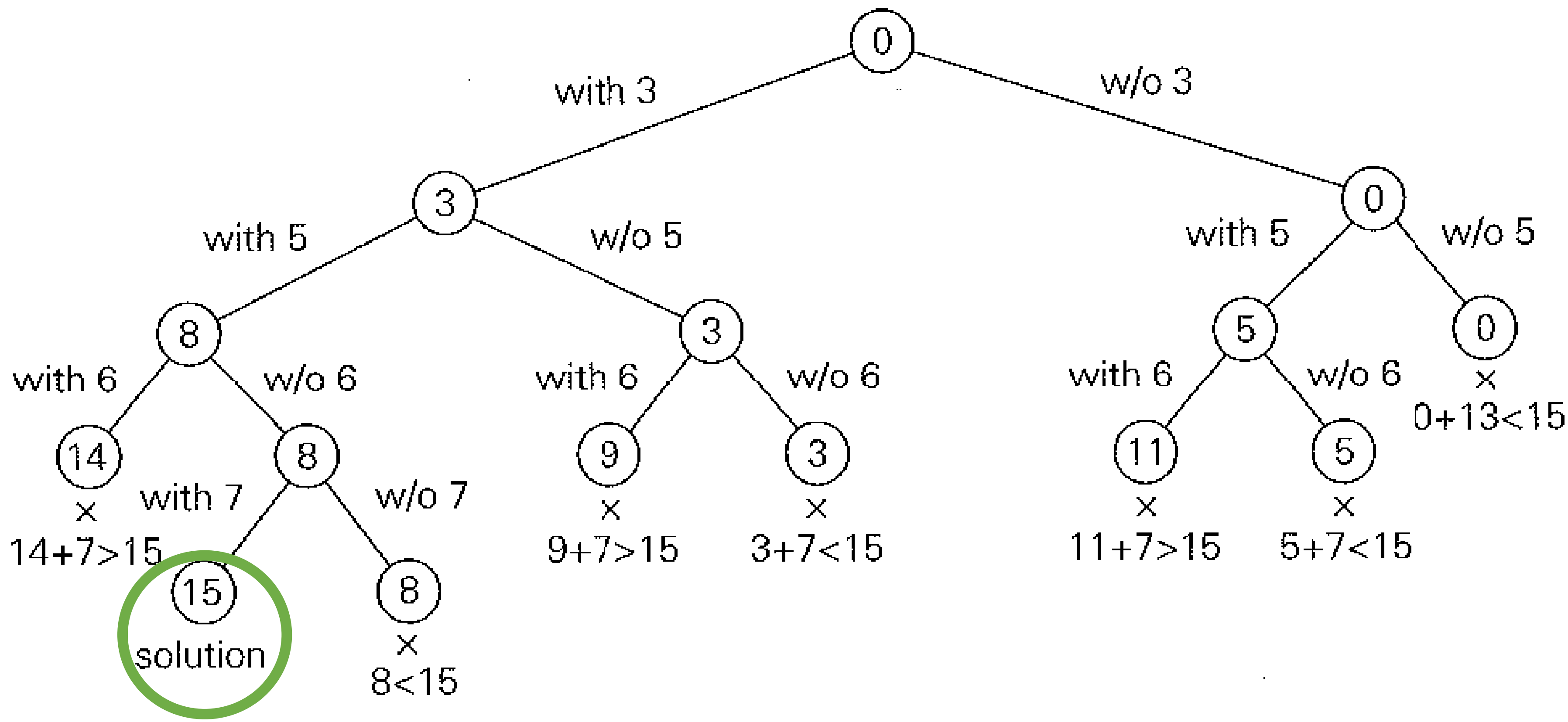
  s1 ≤ s2 ≤ s3 ≤ … ≤ sn

- Construct state space tree:
  - root represents initial state with no decisions made about the given elements.
  - left and right children at i^{th} level represent, respectively, inclusion and exclusion of i^{th} element in a set being sought
  - terminate the node as non promising if either of the two inequalities holds: (s’ is sum of numbers)

$$s' + s_{i+1} > d \text{ (the sum } s' \text{ is too large)}$$

$$s' + \sum_{j=i+1}^{n} s_j < d \text{ (the sum } s' \text{ is too small)}$$

S = {3, 5, 6, 7} and d = 15



Let's check our understanding

- Find a subset of a given set S= {5, 3, 1, 4} whose sum is equal to a 8.

- Apply backtracking to solve the following instance of the subset-sum problem:

  S = {1, 3, 4, 5] and d = 11.