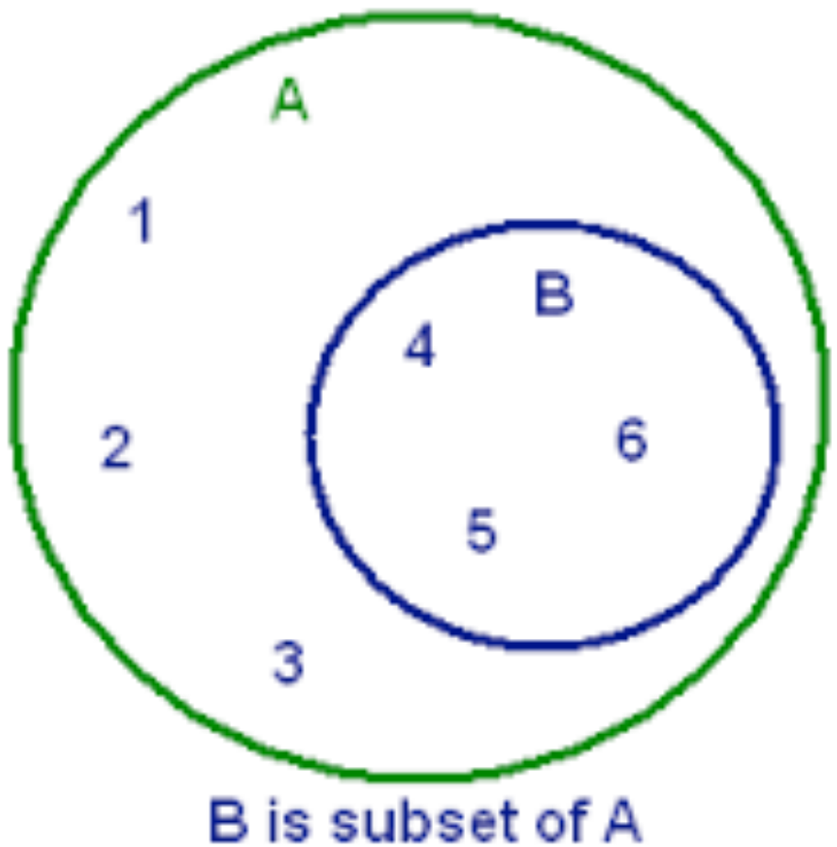
Backtracking

CS 146 - Spring 2017

Question

- Describe an algorithm for enumerating (ie listing) all subsets of a set.

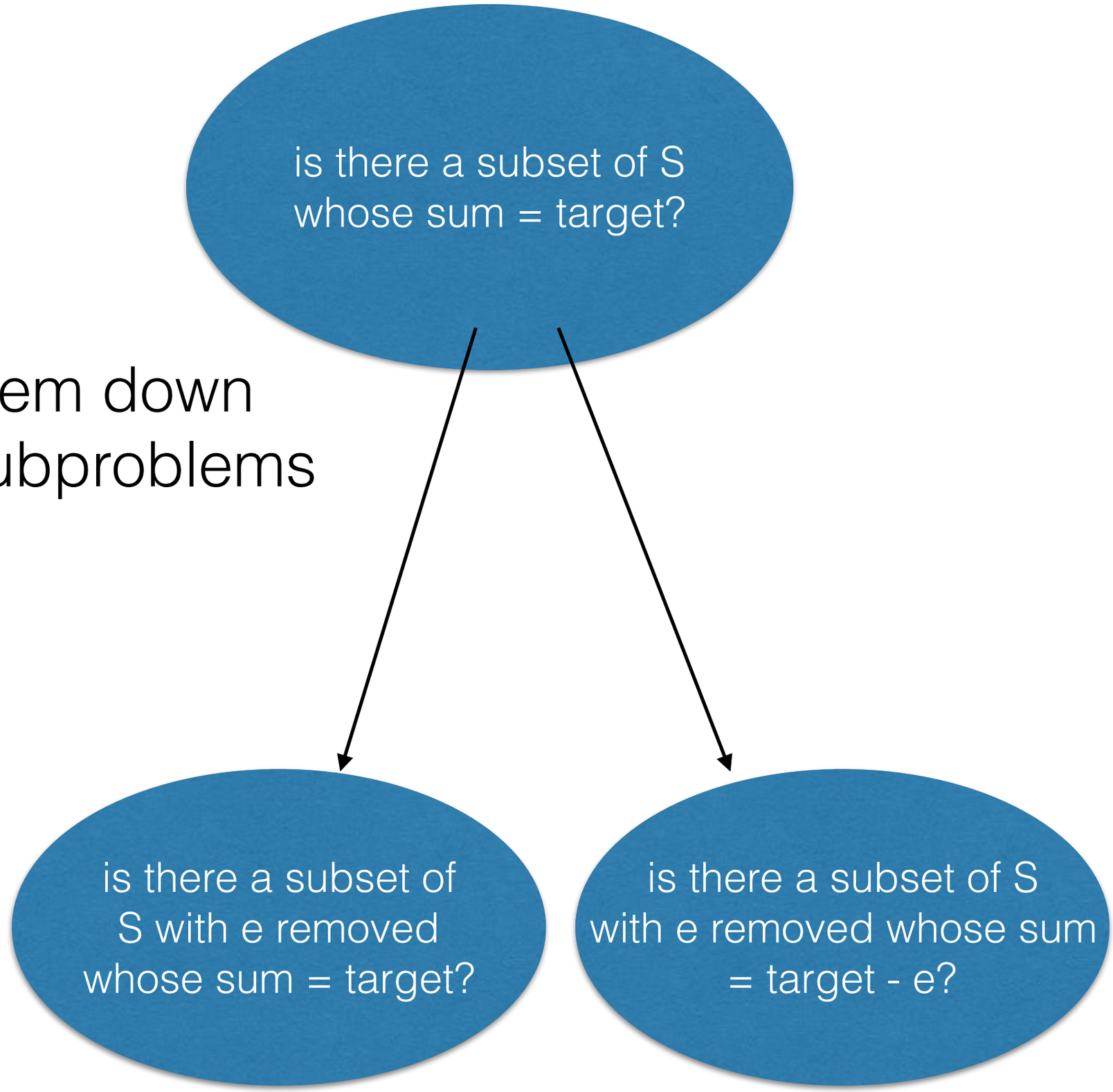


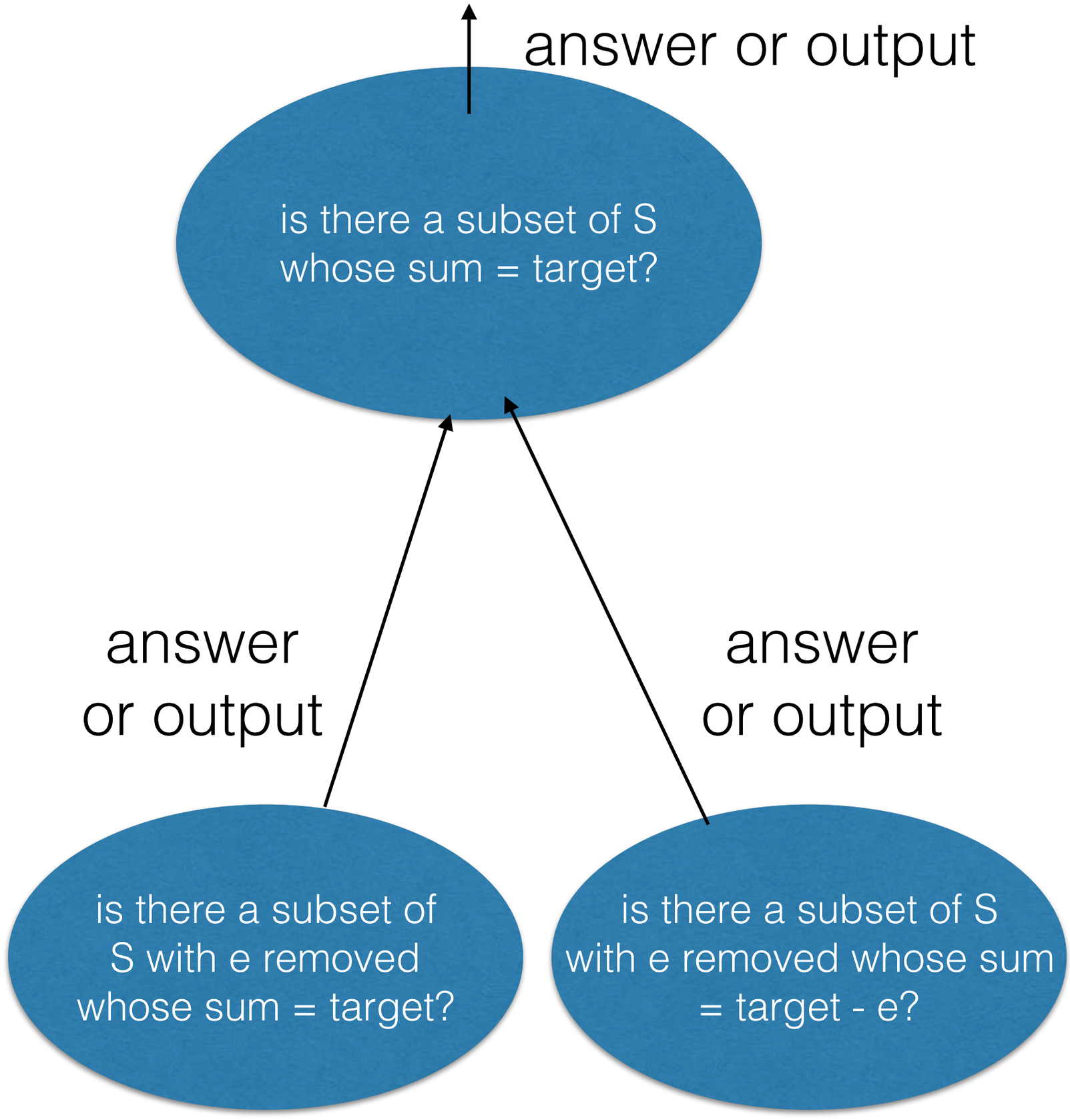
Question 2

- given a list of numbers, is there a (possibly empty) subset of these numbers that sum to n?

- find an algorithm to solve this problem.

break problem down
into smaller subproblems

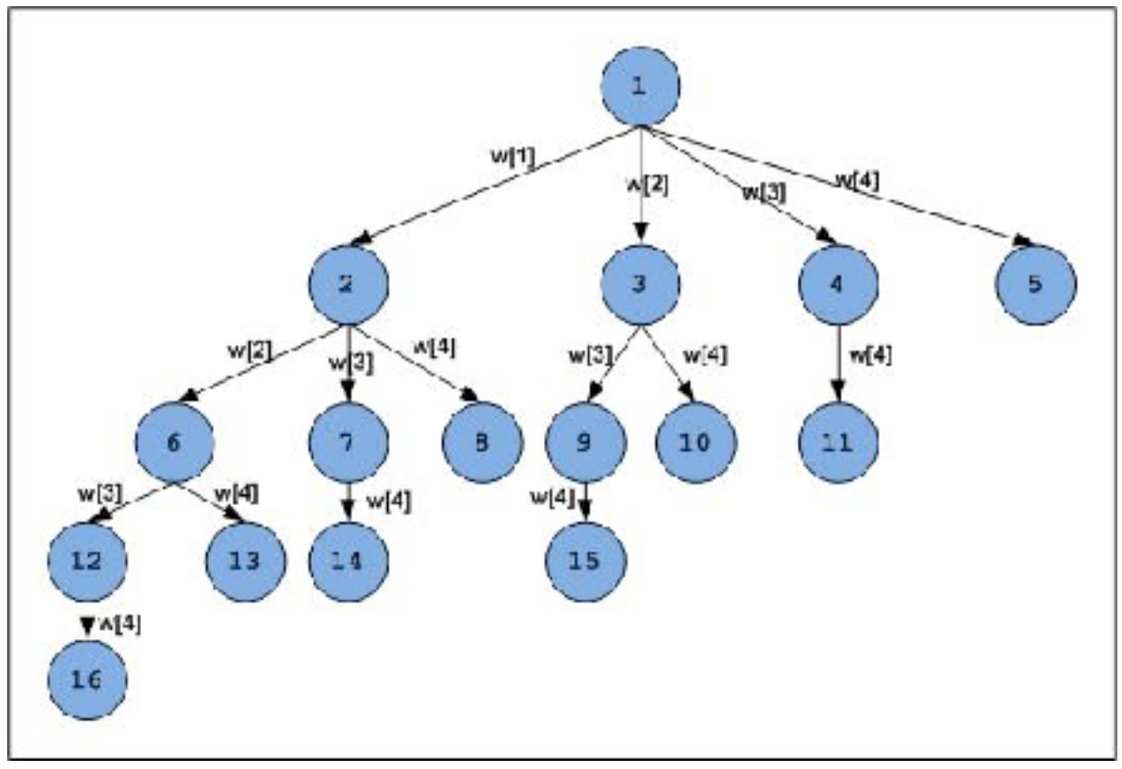



answer or output
is there a subset of S whose sum = target?
answer or output
answer or output
is there a subset of S with e removed whose sum = target?
is there a subset of S with e removed whose sum = target - e?

Backtracking: the idea

- Applies to problems whose solutions are made up of a sequence of choices

- Problem is solved through a controlled brute-force search



Examples of when backtracking applies

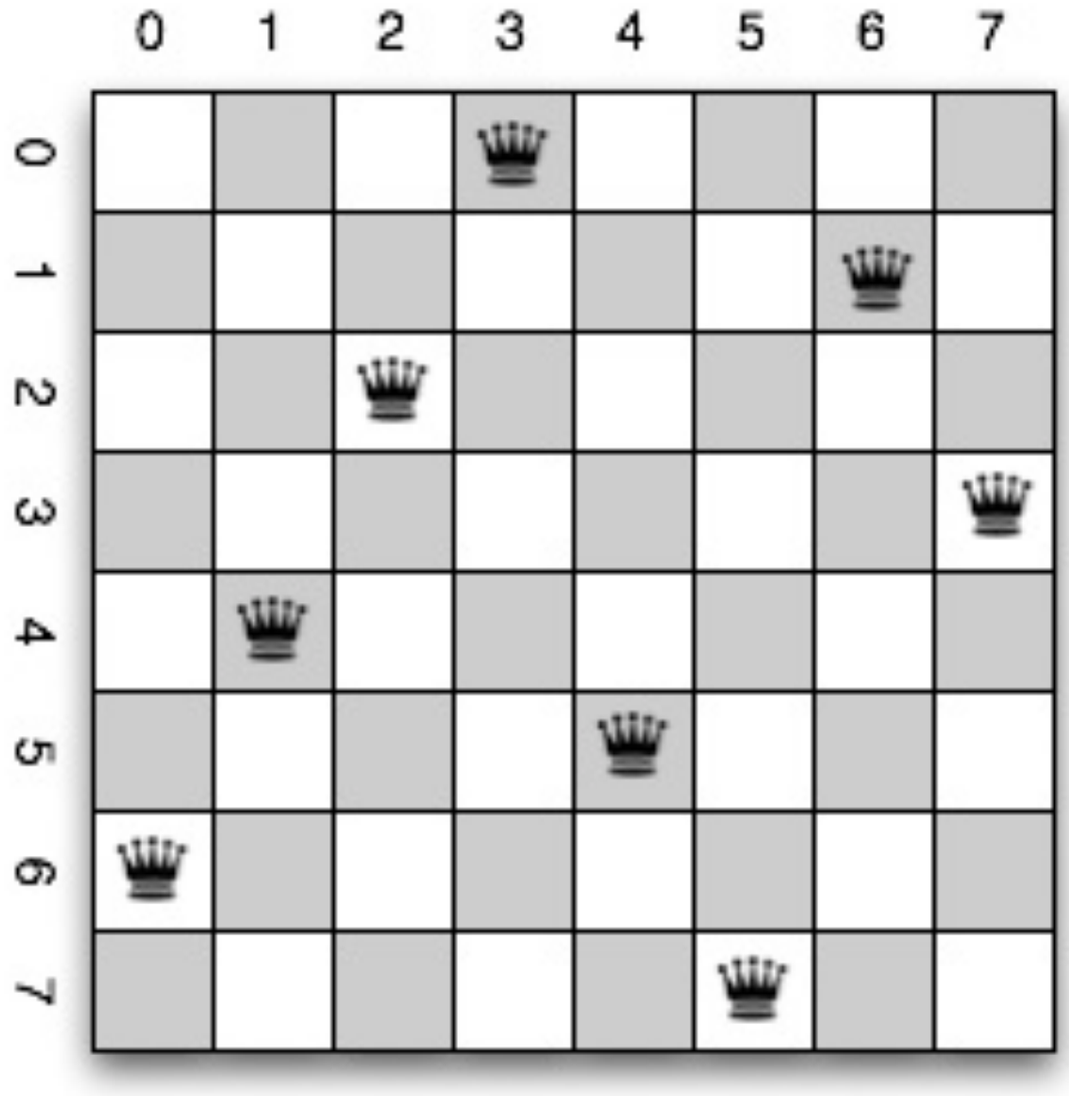
- searching through subsets of a set
- puzzles with constraints (NQueens, Sudoku)
- 2-player games (Tictactoe, connect-4, chess, Go)

Backtracking template

```
solution(problem) {
    if (problem is trivial)
        return solution to problem
    for every valid choice
        subproblem = smaller subproblem
            from problem with choice applied
        solution = solution(subproblem)
    solve problem using all found solutions
}
```

looks a lot like tree traversal

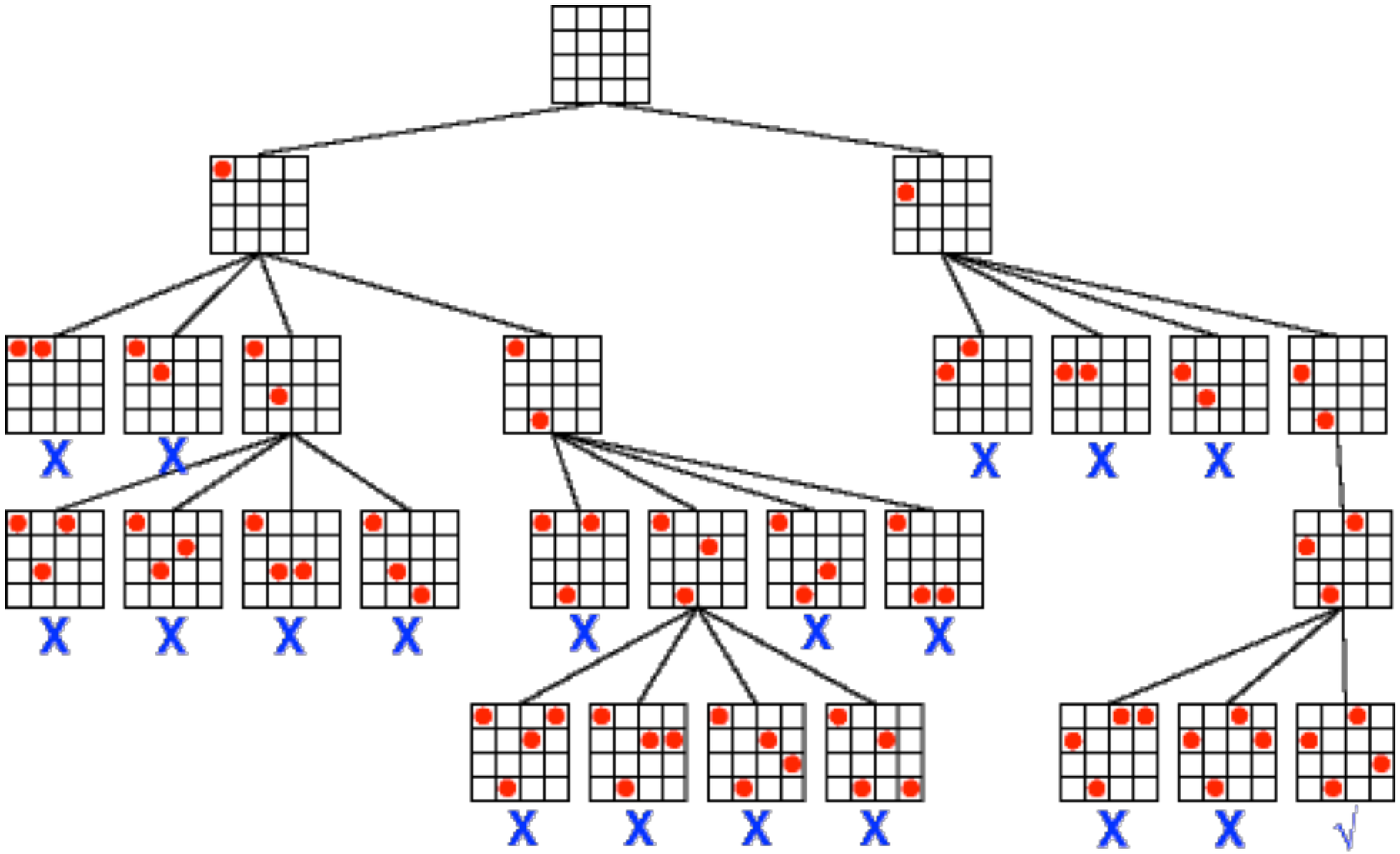
N-Queens problem



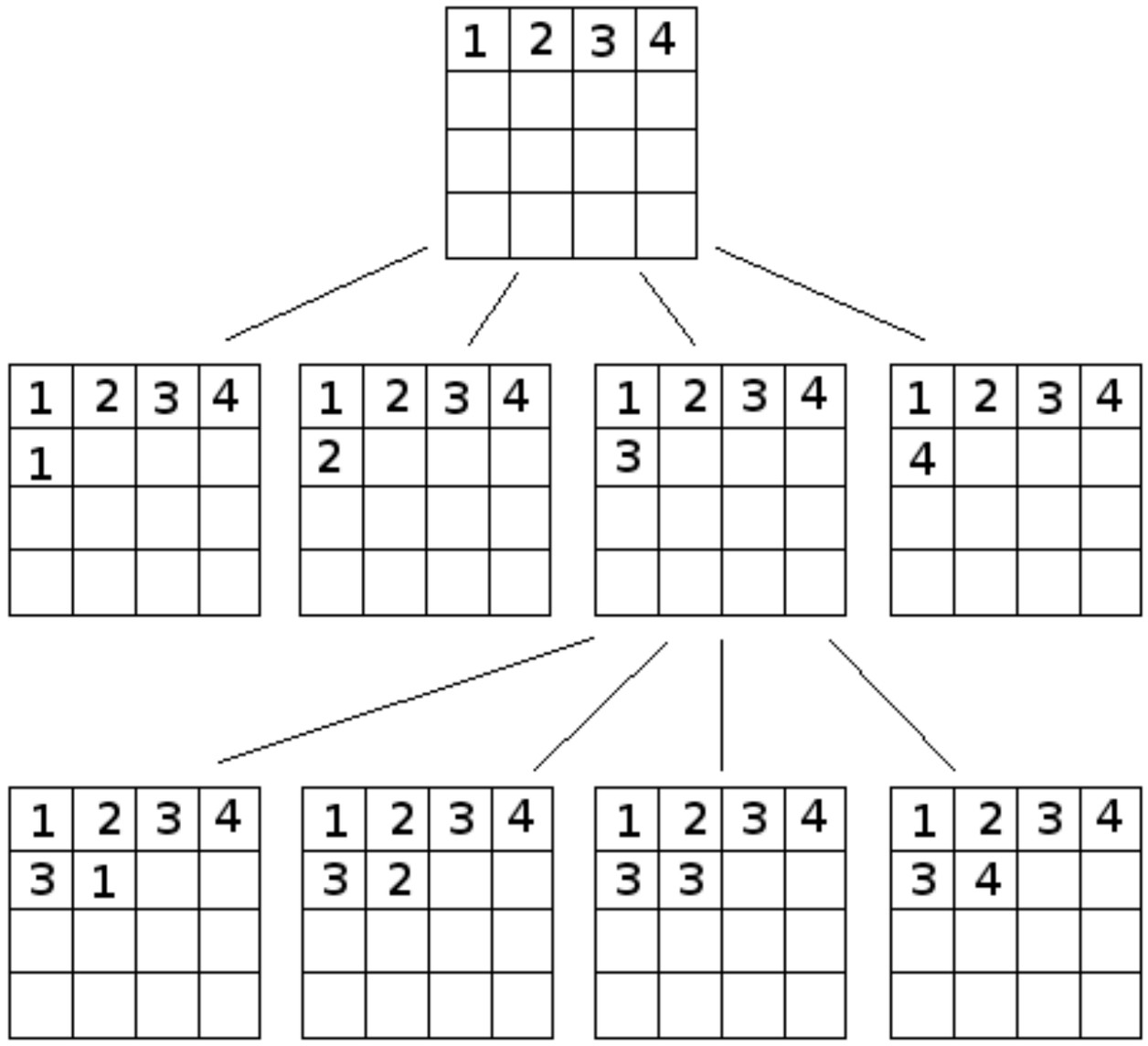
- each row, column and diagonal can contain at most 1 queen

- find all valid configurations with n queens on a n by n board

N-Queens problem



Backtracking in “sudoku”



Backtracking recap

- brute force search
- solution easily expressed using recursion
- BUT
- search space is huge
- what are some shortcuts?