Backtracking Algorithms

Backtracking can be seen as an optimized way to brute force. Brute force approaches evaluate every possibility. In backtracking you stop evaluating a possibility as soon it breaks some constraint provided in the problem, take a step back and keep trying other possible cases, see if those lead to a valid solution.

Let's take a situation. Suppose you are standing in front of three tunnels, one of which is having a bag of gold at its end, but you don't know which one. So you'll try all three. First go in tunnel , if that is not the one, then come out of it, and go into tunnel , and again if that is not the one, come out of it and go into tunnel . So basically in backtracking we attempt solving a subproblem, and if we don't reach the desired solution, then undo whatever we did for solving that subproblem, and try solving another subproblem.

Common structure might apply to many other backtracking questions, pattern includes Subsets, Permutations, and Combination Sum.

Backtracking vs DFS (Depth First Search)

The difference between DFS and backtracking is subtle, but we can summarize like this: DFS is a technique for searching a graph, while backtracking is a problem solving technique (which consists of DFS + pruning, such programs are called backtrackers). So, DFS visits each node until it finds the required value (in your case the target word), while backtracking is smarter - it doesn't even visit particular branches when it is certain that the target word would not be found there.

Popular BackTracking Programming questions

- N-Queens Problem ✅
- Subset Sum ✅
- Rat in a Maze
- Palindrome Partitions ✅
- Permutations ✅
- Combinations ✅
- Hamiltonian cycle

Time Complexities

- **Subset Sum**

```
Let's say input set is [1] then subsets are [],[1] i.e. 2^1 = 2
If [1,2] then subsets [ ], [1], [2], [1,2] i.e. 2^2 = 4
If [1,2,3] then subsets [ ], [1], [2], [1,2], [3], [1,3], [2,3], [1,2,3] i.e. 2^3 = 8
So overall time complexity is O(2^n)

Space complexity:
Since we make recursive call for each answer above, space complexity is also O(2^n)
```

Online resources

- Leetcode Pattern 3 | Backtracking
- A general approach to backtracking

Differences between *Backtracking* and *Branch & Bound* algorithm

Backtracking

1. It is used to find all possible solutions available to the problem.
2. It traverse tree by DFS (Depth First Search).
3. It realizes that it has made a bad choice & undoes the last choice by backing up.
4. It search the state space tree until it found a solution.
5. It involves feasibility function.

Branch-and-Bound

1. It is used to solve optimization problem.
2. It may traverse the tree in any manner, DFS or BFS.
3. It realizes that it already has a better optimal solution that the pre-solution leads to so it abandons that pre-solution.
4. It completely searches the state space tree to get optimal solution.
5. It involves bounding function.