Intermediate Software Engineering

Intermediate-Software-Engineering-Summer-2022 (@ Intermediate)
Personal Member ID#: 52751

Backtracking UMPIRE Cheat Sheet

Definition

Backtracking is similar to Dynamic Programming in that it solves a problem by efficiently performing an exhaustive search over the entire set of possible options. Backtracking is different in that it structures the search to be able to efficiently eliminate large sub-sets of solutions that are no longer possible. For this reason, all backtracking algorithms will have a very similar overall structure for exhaustively searching the space of possible solutions, but the art & difficulty of the particular backtracking solution will be strategies that can be used to get rid of impossible solutions quickly.

U-nderstand

What are some common questions we should ask our interviewer?

- Are there memory constraints?
- What's the required time complexity?
- What kind of data will the inputs be?
- Can I assume all the inputs will be valid?
- What if the input is empty?
- What should we return if there is no solution to the problem?
- What should we return if there are multiple solutions to the problem?
- Think about how you would write a recursive function for this problem. What is the state? What is the base case? How would you modify the state and do recursive calls?

M-atch

Are there any special techniques that we can use to help make this easier?

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.

The problems that can be solved using this strategy generally satisfy the following criteria: You are explicitly asked to return a collection of all answers. You are concerned with what the actual solutions are rather than say the most optimum value of some parameter. (if it were the latter it's most likely DP or greedy).

P-lan/Pseudocode

- Can you create any **magic** helper methods that would simplify the solution?
- Talk through different approaches you can take, and their tradeoffs
- Be able to verbally describe your approach and explain how an example input would produce the desired output

Backtracking Algorithm

```
Backtrack(x)
    if x is not a solution
        return false
    if x is a new solution
        add to list of solutions
    backtrack(expand x)
```

Tips:

- Try to avoid nested loops
  - This is usually a brute force solution and is $O(n^2)$ time complexity
- The backtracking algorithm is applied to some specific types of problems. For instance, we can use it to find a feasible solution to a decision problem. It was also found to be very effective for optimization problems.

E-valuate

Time Complexity

It's not very easy to calculate the time complexity of recursive functions in backtracking. One piece of advice would be to estimate how many possible states the function has, and how much work it does in each state.