Backtracking

Recursion: Where does Backtracking come from?

Recursion is just a function **calling itself** to solve a smaller version of the same problem. We keep breaking the problem down until we hit a base case — the smallest, simplest version — and then we return.

```
def fib(n):
    if n <= 1:
        return n
    return fib(n - 1) + fib(n - 2)
```

Backtracking

Backtracking is a strategy for exploring multiple possible paths, built on top of recursion — but with a twist: it adds a mechanism for trying, failing, and undoing choices.

Backtracking = recursive calls + making choices + undoing those choices

```
         []
       /  |  \
     1    2   3
    /     |     \
  2 3    1 3   1 2
...
```

Pruning Strategies

- **Sorting-based pruning**: Sort the input to group duplicates, then skip repeated elements in the same recursion level to avoid duplicate results.
- **Constraint-based pruning**: If a path already breaks a problem constraint (like exceeding a target), return early to avoid invalid recursion.

Practical tips

1. **Start simple** — always write the basic working backtracking solution first, even if it's not optimized.
2. **Study the recursion tree** — look for repetitive branches or clearly invalid paths.
3. **Add pruning step-by-step** — insert `if` checks inside your `for` loop to skip or `return` when necessary.

Types of Problems

- **Subset problems** – Given a list of numbers, find all possible subsets (combinations of any size).
- **Permutation problems** – Find all the possible ways to reorder the numbers.
- **Combination problems** – Pick `k` numbers from `n`, under certain rules.
- **String segmentation problems** – Cut a string in all valid ways (like palindrome partitioning).
- **Board problems** – Solve constraint-based puzzles like N-Queens or Sudoku.

Template

```
def backtrack(parameters):
    # Base case: when to stop recursion — depends on the problem
    if meet_end_condition:
        save path as a valid result
        return

    for option in options:
        make a choice
        backtrack(path + [option], updated_options)
        undo the choice (if necessary)
```

- **Function arguments and return value**
- **Base case / end condition**
- **The loop**