Recursion and Backtracking

Shivansh (CF : shiv_codegen)

Goal:

- Understand recursion
- Understand applications of recursion
- Learn how to brute-force using recursion
- Assess time complexity of recursive algorithms
- Use backtracking for efficient brute-force

Recap on Functions

- A function is a block of code which runs the code inside with the parameters it is given.

- Syntax:

```
int add(int a, int b) {
    return a + b;
}
```

What is Recursion?

Recursion happens when a function calls itself on a different set of input parameters.

Used when the solution for current problem involves first solving a smaller sub-problem.

Example: factorial(N) = factorial(N-1) * N

Recursive Function

A function that calls itself is a recursive function

Example:

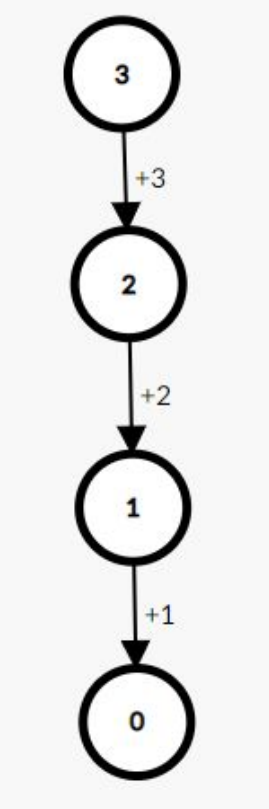
The above function will find the sum from 0 to the given parameter.

```
int sum_0_to_n(int n) {
    if (n <= 0) return 0;
    return sum_0_to_n(n-1) + n;
}
```

Recursive Tree

A recursive tree is similar to a “mind map” of the function call. Each node/vertex is the function call. Value inside the node is the parameter.

Recursive tree of previous example for n = 3 →



Recursive trees are useful to help us understand how the function acts.

Basic Structure of a Recursive Function

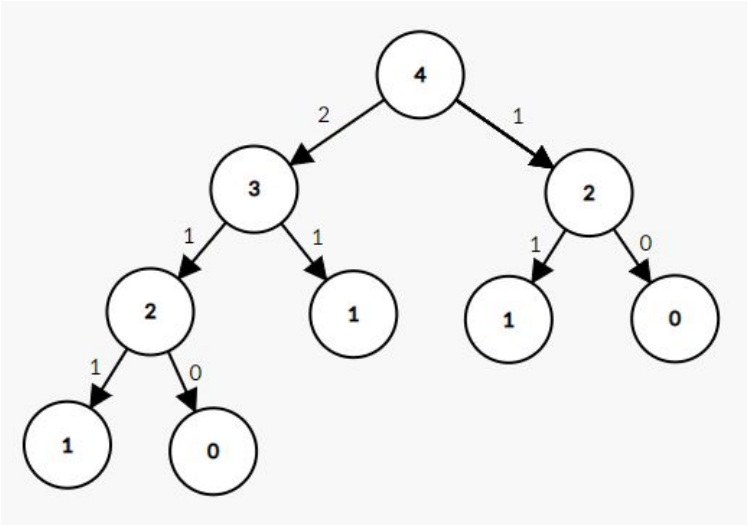
- Parameters to start the function
- Appropriate base case(s) to end the recursion
- Recursively solve the sub-problems
- Process the result and return the value

Tougher example:

Fibonacci function:

```
int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n-1) + fib(n-2);
}
```

Recursive tree:



When do we need recursion?

Recursion is usually used in complex situations where iteration is not feasible.

- Brute-force
- Backtracking
- Dynamic Programming
- Graph/Tree Problems
- Etc.

Quiz 1

1. Write a recursive function to calculate the factorial of a number
2. Write an infinite recursive function that prints the number of times it has run so far.
3. Print a number in binary recursively

Using Recursion to Brute-Force

We can use recursion to go through every possible sub-problem. Also useful when going through every combination/subset of a list.

Examples:

- Print all binary strings of a given length.
- Print all subsets of a given vector.

Time complexity of Recursive Brute-Force

- Can be calculated as the number of recursive calls multiplied by additional complexity of the function.

- Can also be thought of as sum of time complexity of each layer of the recursive tree.

Example functions:

```
void recurse(int n) {
    if (n == 0) return;
    recurse(n-1);
}
```

```
void recurse(int n) {
    if (n == 0) return;
    recurse(n-1);
    recurse(n-1);
}
```

```
void recurse(int n) {
    if (n == 0) return;
    recurse(n/2);
}
```

```
void recurse(int n) {
    if (n == 0) return;
    recurse(n/2);
    recurse(n/2);
}
```

Quiz 2

1. Print all N numbers such that each value can be from 0 to K.

2. Given N coins, print all the values you can make with some combination of coins and sum <= given K.

3. What is the time complexity of:

```
void f(int n) {
    if (n == 0) return;
    if (n % 2 == 0) f(n/2);
    if (n % 2 == 1) f(n-1);
}
```

Resources

- https://bit.ly/39lNlVT (very detailed explanation)
- https://codeforces.com/blog/entry/92031 (advanced)