Recursion
+
Dynamic Programming

Srivaths P

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:

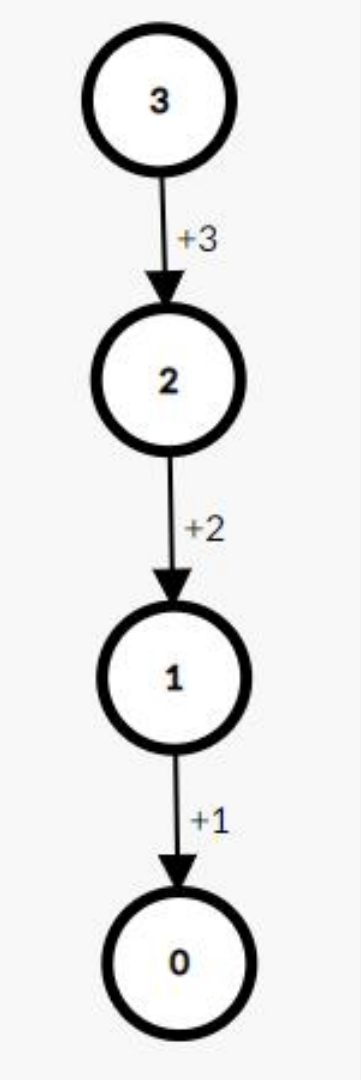
```
int sum_0_to_n(int n) {
    if (n <= 0) return 0;
    return sum_0_to_n(n-1) + n;
}
```

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

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

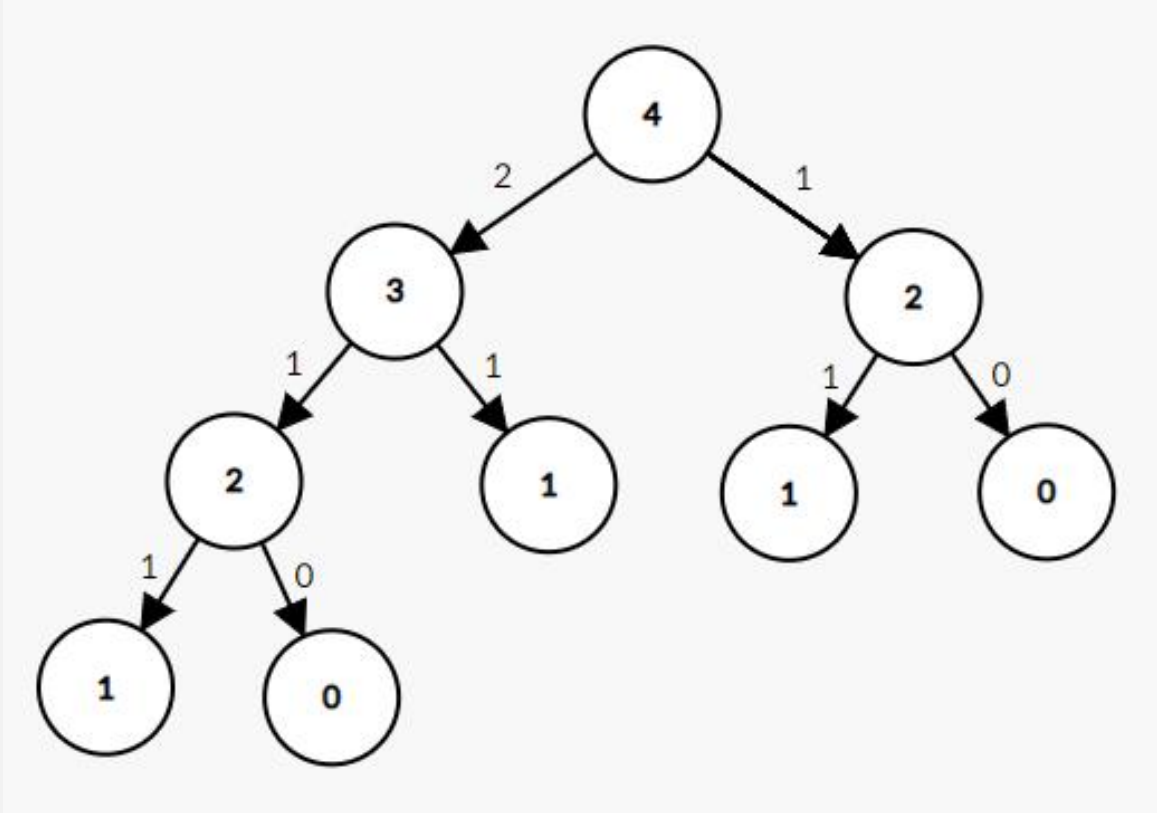
- Parameters to start the function
- Appropriate base case(s) to end the recursion
- Recursively solve the sub-problems
- Process the result of the children and return the value

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:



Time complexity of Recursive Algorithms

- 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.
- Time Complexity = O((children ^ depth) * time per node)
  Where children is the average number of children per node.

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);
}
```

Memoization / Dynamic Programming

Memoization is simply storing solution of nodes in the recursive tree that we have already computed.

We can use arrays or maps. Arrays are generally preferred due to their speed compared to maps.

The time complexity reduces to O(T.C. per node) * size of DP array.

Problems

- https://cses.fi/problemset/task/1623
- https://codeforces.com/problemset/problem/474/D
- https://cses.fi/problemset/task/1093
- https://codeforces.com/contest/1703/problem/G
- https://codeforces.com/problemset/problem/1714/D