Recursion and Backtracking

By: Andrew Qi Tang, Ethan Pronev

Recursion

When a function calls on itself directly or indirectly. (We'll focus on direct)

Idea is to solve a current problem by solving smaller problems.

Needs two parts:

1. Base Case
2. Recurrence

Basic Structure

```
void fun(){

 fun();

}
```

Recurrence

When you call the function itself again.

This is where the recursion happens.

Base Case

When do I stop?

Base Case is when you stop.

Cannot go infinitely, otherwise you’ll receive stack overflow error.

Example 1: Fibonacci Numbers

Famous sequence.

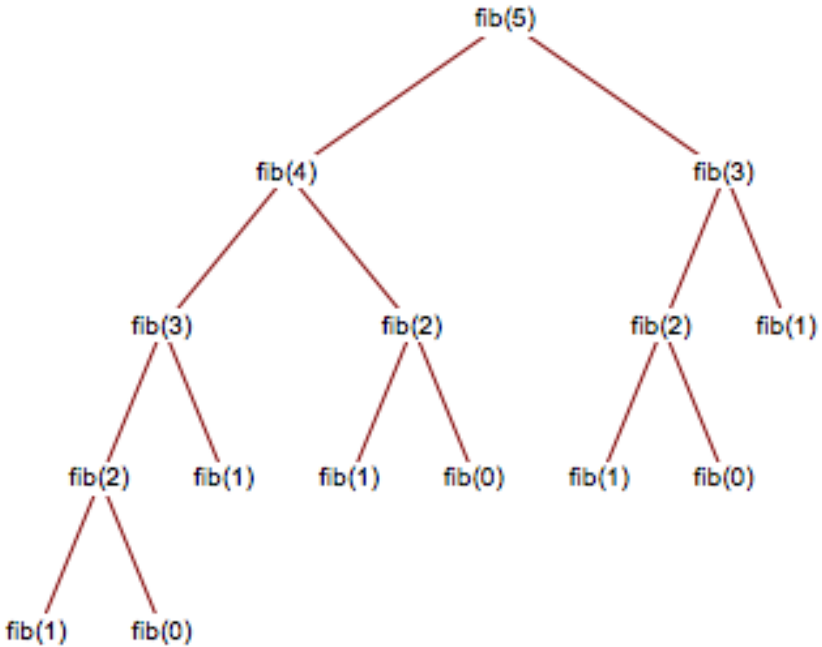
Depending on the source: 1, 1, 2, 3, 5, 8 …

The i-th term is defined as $F_i = F_{i-1} + F_{i-2}$ if i >= 2, $F_1 = 1$, $F_0 = 1$.

Example 1: Fibonacci Numbers

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

Recursion Tree for Example 1



Example 1: Fibonacci Numbers

Base Case: $F_0 = 1$ and $F_1 = 1$

Recurrence: $F_{n-1} + F_{n-2} = F_n$

Backtracking

Generates all possible ways.

Begins with an empty solution, and branch out.

Brute Force.

Can be thought of as Branch and Bound.

Normally uses recursion.

Example 2: N Queens Problem

Given an N by N chess board, count the number of ways to put N queens on it so that none of them are attacking each other.

Example 2: N Queen Problem

Idea is to try every possibility.

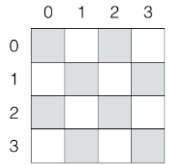
Notice that if a queen is on a row, then there will be only one queen there.

Go through the rows one by one a place queen.

Update the squares it attacks.

Base Case when there are no more rows left, we return 1.

Simulation



Example 2 Code

```
void upd(int v, int x, int y){
   for(int n = 0; n<N; n++){
       board[n][y] += v;
       board[x][n] += v;
       if(n+x < N && n+y < N){
           board[n+x][n+y] += v;
       }
       if(x-n >= 0 && n+y < N){
           board[x-n][n+y] += v;
       }
       if(x-n >= 0 && y-n >= 0){
           board[x-n][y-n] += v;
       }
       if(n+x < N && y-n >= 0){
           board[n+x][y-n] += v;
       }
   }
}
```

```
int solve(int n){
   if(n == 0){
       return 1;
   }
   int ret = 0;
   for(int i = 0; i<N; i++){
       if(!board[n][i]){
           upd(1, n, i);
           ret += solve(n-1);
           upd(-1, n, i);
       }
   }
   return ret;
}
```

https://pastebin.com/CFuFytHF

Extra Notes

Good for generating all possibilities.

Recursion is OP but slow and big.

Homework

https://dmoj.ca/problem/gfssoc3j5

https://dmoj.ca/problem/vmss7wc16c4p2

https://dmoj.ca/problem/ccc96s3

https://dmoj.ca/problem/valentines18j5s2

https://dmoj.ca/problem/cco12p1

https://dmoj.ca/problem/ccc13s3