Competitive Training Camp
Lecture 1

Christian Yongwhan Lim
Tuesday, July 15, 2025

Christian Yongwhan Lim

COLUMBIA UNIVERSITY

icpc .foundation

Education

Stanford University

Part-time Jobs

CORNELL TECH

MIT EECS

IN LVMINE TVO VIDEBIMVS LVMEN

Full-time Job

Google Research

TWO SIGMA

Workshops

Stanford ENGINEERING | Stanford Computer Forum

Carnegie Mellon University

HARVARD

UNIVERSITY OF CALIFORNIA

KAIST

NUS National University of Singapore

Coach/Judge

icpc International Collegiate Programming Contest

icpc.foundation

advancing the art and sport of competitive programming

https://www.yongwhan.io

Christian Yongwhan Lim

- **Head Coach**, Columbia ICPC Team
- **Chief Judge**, ICPC NA Mid-Central
- **Judge**, ICPC NA Qualifiers and Regionals
- **Chair**, ICPC CLI Symposium

https://www.yongwhan.io

Christian Yongwhan Lim

- **Head Coach**, Columbia ICPC Team
- **Chief Judge**, ICPC NA Mid-Central
- **Judge**, ICPC NA Qualifiers and Regionals
- **Chair**, ICPC CLI Symposium

- **Director**, ICPC Internships
- **Adjunct**, Columbia CS
- **VP of Engineering**, Arklex AI

https://www.yongwhan.io

Who are you?

- **Quick Introduction!**

Dynamic Programming

- So, what is it?

Dynamic Programming

- So, what is it?

- Why is it infamous?

Dynamic Programming (Warm-Up)

- Fibonacci sequence

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

Dynamic Programming: Top-Down (Warm-Up)

- Fibonacci sequence

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

- **TOO SLOW!** How do you optimize this a bit?

Dynamic Programming: Top-Down (Warm-Up)

- One answer: **Memoization!**

```
const int MAXN = 100;
bool found[MAXN];
int memo[MAXN];
int f(int n) {
  if (found[n]) return memo[n];
  if (n==0) return 0;
  if (n==1) return 1;
  found[n] = true;
  return memo[n] = f(n-1) + f(n-2);
}
```

Dynamic Programming: Top-Down (Warm-Up)

- You may use `map` or `unordered_map` (though slower).

```
map<int, int> memo;
int f(int n) {
  if (memo.count(n)) return memo[n];
  if (n==0) return 0;
  if (n==1) return 1;

  return memo[n] = f(n-1) + f(n-2);
}
```

Dynamic Programming: Bottom-Up (Warm-Up)

- Another answer (to speed up): **bottom-up**

```
const int MAXN = 100;
int fib[MAXN];

int f(int n) {
  fib[0] = 0;
  fib[1] = 1;
  for (int i = 2; i <= n; i++)
    fib[i] = fib[i-1] + fib[i-2];
  return fib[n];
}
```

Dynamic Programming: Bottom-Up (Warm-Up)

- To save a memory, since you are using only previous two values, you can do:

```
const int MAX = 3;
int fib[MAX];
int f(int n) {
  fib[0] = 0;
  fib[1] = 1;
  for (int i = 2; i <= n; i++)
    fib[i%MAX] = fib[(i-1)%MAX] + fib[(i-2)%MAX];
  return fib[n%MAX];
}
```

Classic Dynamic Programming (DP) Problems

- Counting all possible paths from top left to bottom right corner of a matrix (**Combinations**)
- **0-1 Knapsack**
- **Subset** Sum
- Coin Change (**CC**)

[DP] Path Counting

[DP] 0-1 Knapsack

[DP] Subset Sum

[DP] Live Coding: CSES 1636: Coin Combinations II

- Sample Code