Algorithm Design

Chapter 0

Nattee Niparnan

Introduction to Algorithm

What is an Algorithm?

- A precise instruction based on computational operation
  - Which takes some value as input and process them into output
  - Precise = no ambiguity in what to do
- We use an algorithm to solve a computational problem

What is a Computational Problem?

- A task with general description of what output is needed from the input
  - Describe what kind of admissible input we need
  - Describe the property of the desired output

- Example: GCD (Greatest Common Divider)
  - Given two positive integers (input)
  - Calculate GCD of the given input (the desired output)
    - GCD is well defined

GCD Problem

- Input: two positive integers A and B
- Output: the GCD of A and B (which is the largest integer by which both A and B can be divided)

Problem Instance

- Determining GCD is a problem
  - How many actual problems?
    - GCD of 1 and 2?
    - GCD of 234 and 42?
    - More? Obviously yes.
    - A pair of -2 and 8 is not an admissible input (because -2 is negative)
- Problem instance
  - A problem with specific values of input
  - E.g., find a GCD of 42 and 14

Algorithm Designing Goal

- Algorithm should be
  - Correct
    - For any admissible instances, it must correctly produce desired output
  - Efficient
    - Compute the output using reasonable resource (time, memory)

```
int GCD(int A,int B) {
  int ans = 1;
  for (int i = 2;i < min(A,B);i++) {
    if (A % i == 0 && B % i == 0)
      ans = i
  }
  return ans;
}
```

Calculating Fibonacci Sequence

- Fibonacci sequence
  - 0, 1, 1, 2, 3, 5, 7, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584

$$F_n = \begin{cases} 0 & : n = 0; \\ 1 & : n = 1; \\ F_{n-1} + F_{n-2} & : n > 1; \end{cases}$$

The Problem

- Input: a non-negative integer N
- Output: F_n (the n^{th} Fibonacci Number)

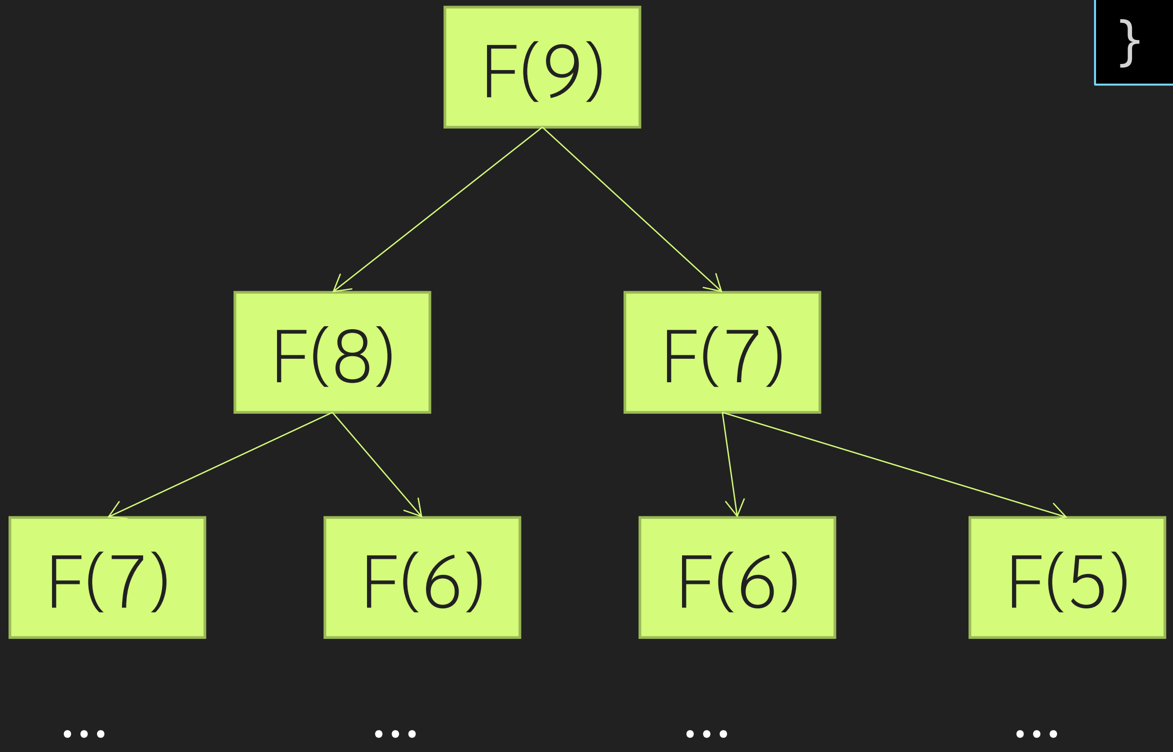
- Example instances
  - Ex. 1: N = 10
  - Ex. 2: N = 15
  - Ex. 3: N = 0

N = -4 is not an instances of this problem!!!

Approach 1

- Method: Recursive $O(2^n)$

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



Approach 2

- Method: Dynamic Programming O(n)

```
vector<int> v(n+1);
v[0] = 0;
v[1] = 1;
for (int i = 2;i <= n;i++) {
  v[i] = v[i-1] + v[i-2];
}
```

	0	1	2	3	4	5	6	7	8	9
v	0	1	1	2	3	5	8	13	21	34

Approach 3

$$\begin{pmatrix} F_2 \\ F_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

$$\begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \begin{pmatrix} F_1 \\ F_0 \end{pmatrix}$$

```
vector<vector<int>> matrix_expo(const vector<vector<int>>& A, int exp) {
  if (exp == 1) return A;

  vector<vector<int>> half = matrix_expo(A, exp / 2);
  if (exp % 2 == 0) {
    return multiply(half,half);
  } else {
    return multiply(multiply(half, half), A);
  }
}


int fibonacci(int n) {
  if (n == 0) return 0;
  if (n == 1) return 1;

  vector<vector<int>> base = {{1, 1}, {1, 0}};
  vector<vector<int>> result = matrix_expo(base, n - 1);

  return result[0][0];
}
```

- Method: Divide and Conquer O(lg n)

Approach 4

Golden Ratio

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{-2}{1+\sqrt{5}}\right)^n}{\sqrt{5}}$$

- Method: Closed form solution

Conclusion

- Different Design → Difference Performance
- This class emphasizes on designing efficient algorithm

Algorithm?

- Named after a Persian mathematician
  Muhammad ibn Musa al-Khwarizmi
- Wrote books on linear equation
- Introduce number 0