2020 ICPC 6주차
:다이나믹 프로그래밍 초급편

2016024911 이상윤

0. 다이나믹 프로그래밍은?

주어진 문제를

여러개의 부분 문제 들로 나누어 푼 다음,

그결과들로 주어진 문제를 푸는것!

0. 다이나믹 프로그래밍은? : 피보나치 수열

$$F(n) := \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F(n-1) + F(n-2) & \text{if } n > 1. \end{cases}$$

F(0)=0, F(1)=1, F(2)=F(1)+F(0),F(3)=F(2)+F(1) …

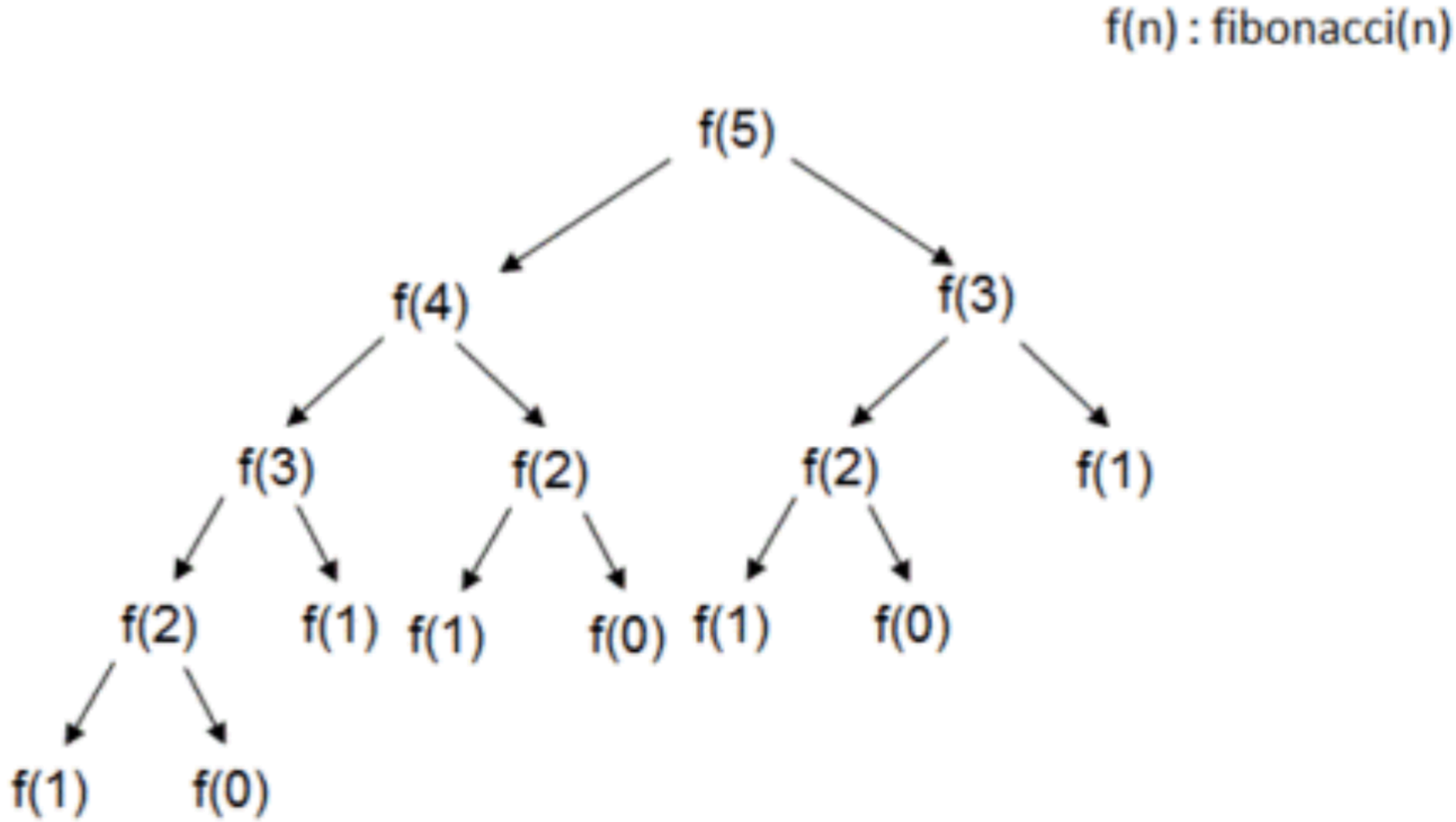
0. 다이나믹 프로그래밍은? : 피보나치 수열

```cpp
#include <cstdio>
using namespace std;

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

int main(){
    int N;
    scanf("%d", &N);
    printf("%d\n", fibonacci(N));
}
```


우리가 파이썬으로든 C로든 한번은 짜본 그런 형태…

0. 다이나믹 프로그래밍은? : 피보나치 수열

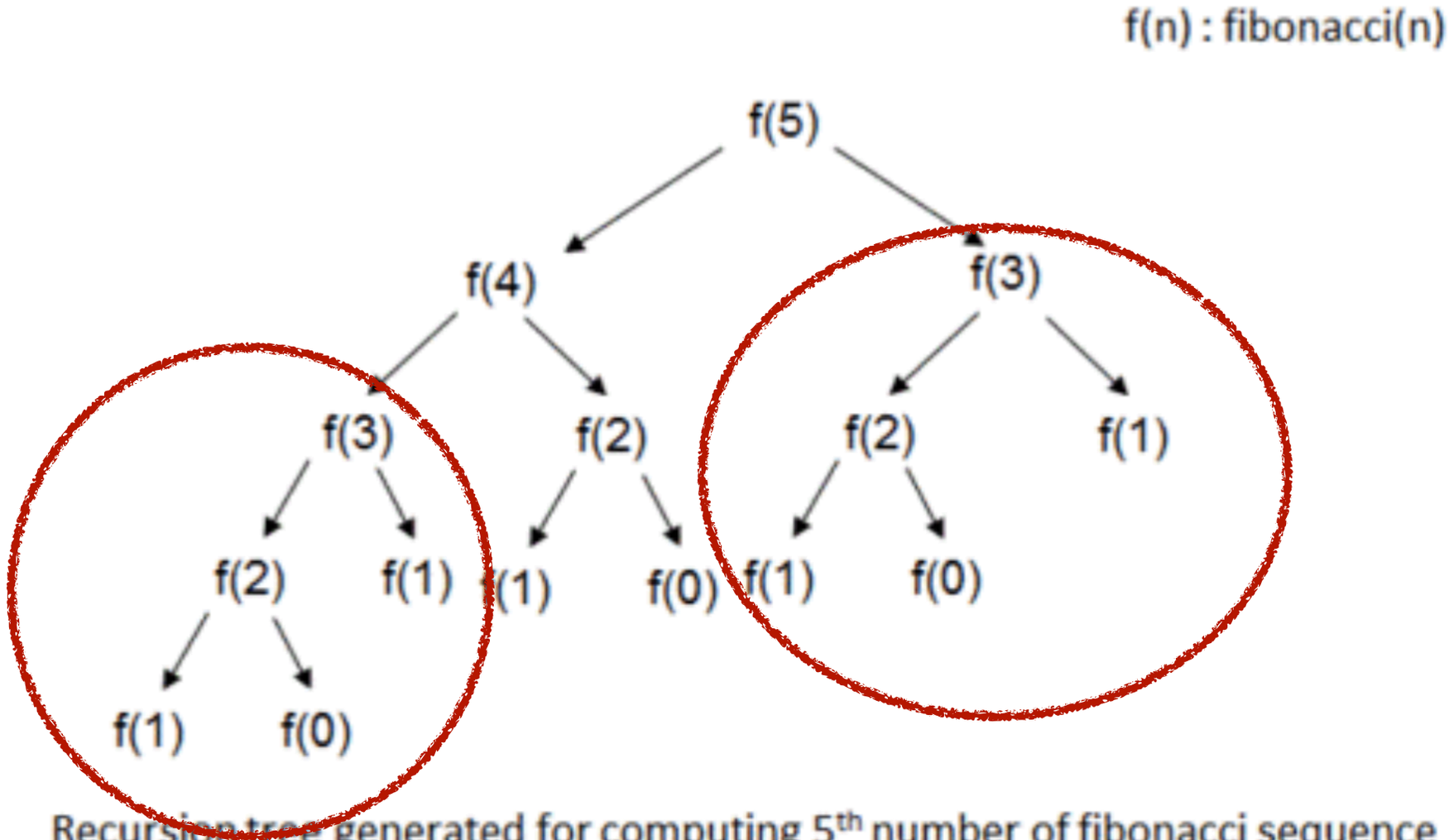
하지만!!!



Recursion tree generated for computing 5th number of fibonacci sequence

0. 다이나믹 프로그래밍은? : 피보나치 수열

반복적인 계산들… -> 시간복잡도 증가!



Recursion tree generated for computing 5th number of fibonacci sequence

0. 다이나믹 프로그래밍은? : 피보나치 수열

```cpp
#include <cstdio>
#include <vector>
using namespace std;

vector<int> dp;

int fibonacci(int n){
    if(n == 0) return 0;
    if(n == 1) return 1;
    // 이미 값을 계산한 적이 있다면 그 값을 바로 리턴
    if(dp[n] != -1) return dp[n];
    // 아니라면 계산해서 dp 리스트에 넣어 보존
    dp[n] = fibonacci(n-2) + fibonacci(n-1);
    return dp[n];
}

int main(){
    int N;
    scanf("%d", &N);
    dp.resize(N+1, -1); // 초기값 -1은 fibonacci 결과로 절대 나올 수 없는 값
    printf("%d\n", fibonacci(N));
}
```


메모이제이션을 활용하자!

0. 다이나믹 프로그래밍은? : 피보나치 수열

<탑다운 > + 메모이제이션

```cpp
#include <cstdio>
#include <vector>
using namespace std;

vector<int> dp;

int fibonacci(int n){
    if(n == 0) return 0;
    if(n == 1) return 1;
    // 이미 값을 계산한 적이 있다면 그 값을 바로 리턴
    if(dp[n] != -1) return dp[n];
    // 아니라면 계산해서 dp 리스트에 넣어 보존
    dp[n] = fibonacci(n-2) + fibonacci(n-1);
    return dp[n];
}

int main(){
    int N;
    scanf("%d", &N);
    dp.resize(N+1, -1); // 초기값 -1은 fibonac
    printf("%d\n", fibonacci(N));
}
```

이해하기 쉬움

O(N)의 시간복잡도

O(N)의 공간복잡도

<바텀업> + 슬라이딩 윈도우

```cpp
#include <cstdio>
#include <vector>
using namespace std;

int main(){
    int N,ans=1;
    scanf("%d",&N);
    int dp=0,ddp=1;
    for (int i=2;i<=N;i++){
        ans=dp+ddp;
        dp=ddp;
        ddp=ans;
    }
    printf("%d\n",ans);
}
```

O(N)의 시간복잡도

O(N)의 메모리가 안필요해짐

함수를 부르지 않아
시간과 메모리 아주조금 이득

예제 생각해보기
:1463 1로만들기

1. 예제 생각해보기 : 1463 1로 만들기

1463 1로만들기

문제

정수 X에 사용할 수 있는 연산은 다음과 같이 세 가지 이다.

1. X가 3으로 나누어 떨어지면, 3으로 나눈다.
2. X가 2로 나누어 떨어지면, 2로 나눈다.
3. 1을 뺀다.

정수 N이 주어졌을 때, 위와 같은 연산 세 개를 적절히 사용해서 1을 만들려고 한다. 연산을 사용하는 횟수의 최솟값을 출력하시오.

2 : 2 > 1 1번

10 : 10 > 9 > 3 > 1: 3번

그럼 다른것들은…?

1. 예제 생각해보기 : 점화식

f(N) = min{

0 (N=1일 경우)

f(N/3)+1, (N이 3의 배수일 경우)

f(N/2)+1, (N이 2의 배수일 경우)

f(N-1)+1

}

1. 예제 생각해보기 : 구현

<탑다운 > + 메모이제이션

```cpp
#include <cstdio>
#include <algorithm>
using namespace std;
const int MAX = 1000001;

int dp[MAX];

int f(int n){
    if(n == 1) return 0; // base case
    if(dp[n] != -1) return dp[n]; // 이미 계산함

    int result = f(n-1) + 1;
    if(n%3 == 0) result = min(result, f(n/3) + 1);
    if(n%2 == 0) result = min(result, f(n/2) + 1);
    dp[n] = result;
    return result;
}

int main(){
    int N;
    scanf("%d", &N);
    fill(dp, dp+MAX, -1);
    printf("%d\n", f(N));
}
```


<바텀업>

```cpp
#include <stdio.h>
#define min(x,y) x>=y?y:x

int result[1000010];

int main() {
    int X;
    scanf("%d", &X);
    result[1] = 0;
    for (int i = 2; i <= X; i++) {
        result[i] = result[i - 1] + 1;
        if (i % 3 == 0) result[i] = min(result[i], result[i / 3] + 1);
        if (i % 2 == 0) result[i] = min(result[i], result[i / 2] + 1);
    }
    printf("%d", result[X]);
    return 0;
}
```

풀어볼문제
:총 10문제 문제 컷 8개!

1904

11726
11727
9465
2193
11051
11057

★

12865
16500
11055