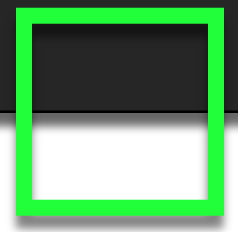
#알고리즘 반
#3주차

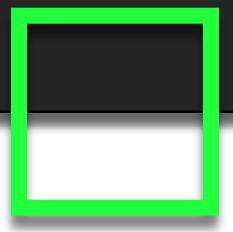
DP 기초

T. 심상용
Asst. 박정호, 최민혁



ALOHA
The algorithm club.

#1 What is DP??



Dynamic Programming 이란?

동적 계획법

- 큰 문제를 작은 문제로 나눠서 푸는 알고리즘
 -> 문제를 여러 개의 하위 문제(subproblem)로 나누어 푼 다음, 그것을 결합하여 최종적인 목적에 도달하는 것
 -> 각 하위 문제의 해결을 계산한 뒤, 그 해결책을 저장하여 후에 같은 하위 문제가 나왔을 경우 그것을 간단하게 해결

- 동적계획법은 계산 횟수를 줄일 수 있다. 특히 하위 문제의 수가 기하급수적으로 증가할 때 유용

Dynamic Programming 이란?

동적 계획법

예를 들어!!

Dynamic Programming 이란?

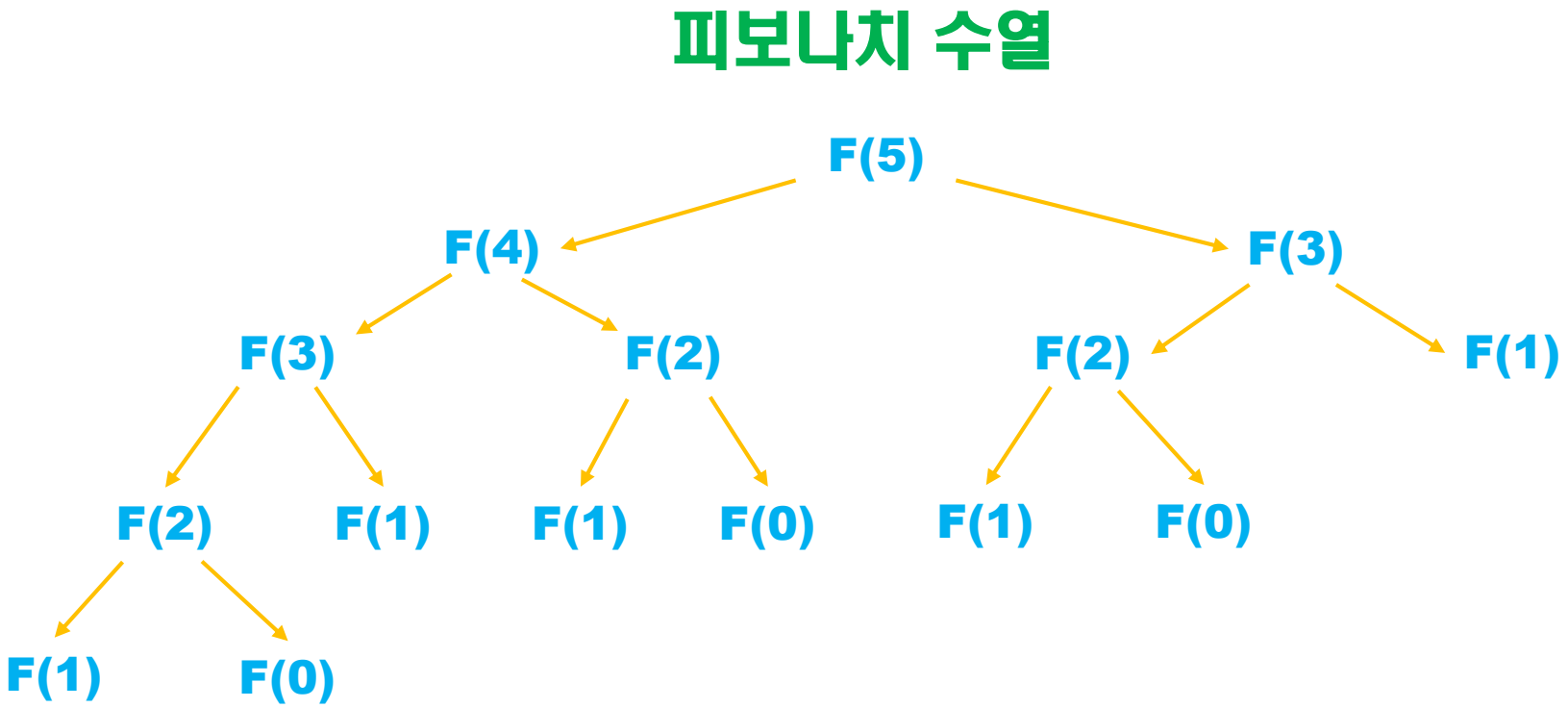
동적 계획법

피보나치 수열

$$F(x) = \begin{cases} 0 \ , x = 0 \\ 1 \ , x = 1 \\ F(x-1) + F(x-2) \ , x > 1 \end{cases}$$

Dynamic Programming 이란?

동적 계획법



Dynamic Programming 이란?

동적 계획법

이를 사용하기 위해서??

Dynamic Programming 이란?

동적 계획법

이를 사용하기 위해서??

2가지 조건!!!

Dynamic Programming 이란?

동적 계획법

이를 사용하기 위해서??

2가지 조건!!!

Overlapping Subproblem

Optimal Structure

Dynamic Programming 이란?

동적 계획법

이를 사용하기 위해서??

2가지 조건!!!

Overlapping Subproblem

- 중복되는 부분 문제
- 다른 subproblem을 구할 때 같은 문제가 겹치는 경우가 발생??

Dynamic Programming 이란?

동적 계획법

이를 사용하기 위해서??

2가지 조건!!!

- 문제의 정답을
  작은 문제의 정답들로

- 이러한 작은 문제들의
  정답들을 기억하기!!
  -> Memoization

Optimal Structure

Dynamic Programming 이란?

동적 계획법

이를 사용하기 위해서??

2가지 조건!!!

Overlapping Subproblem

+

Optimal Structure

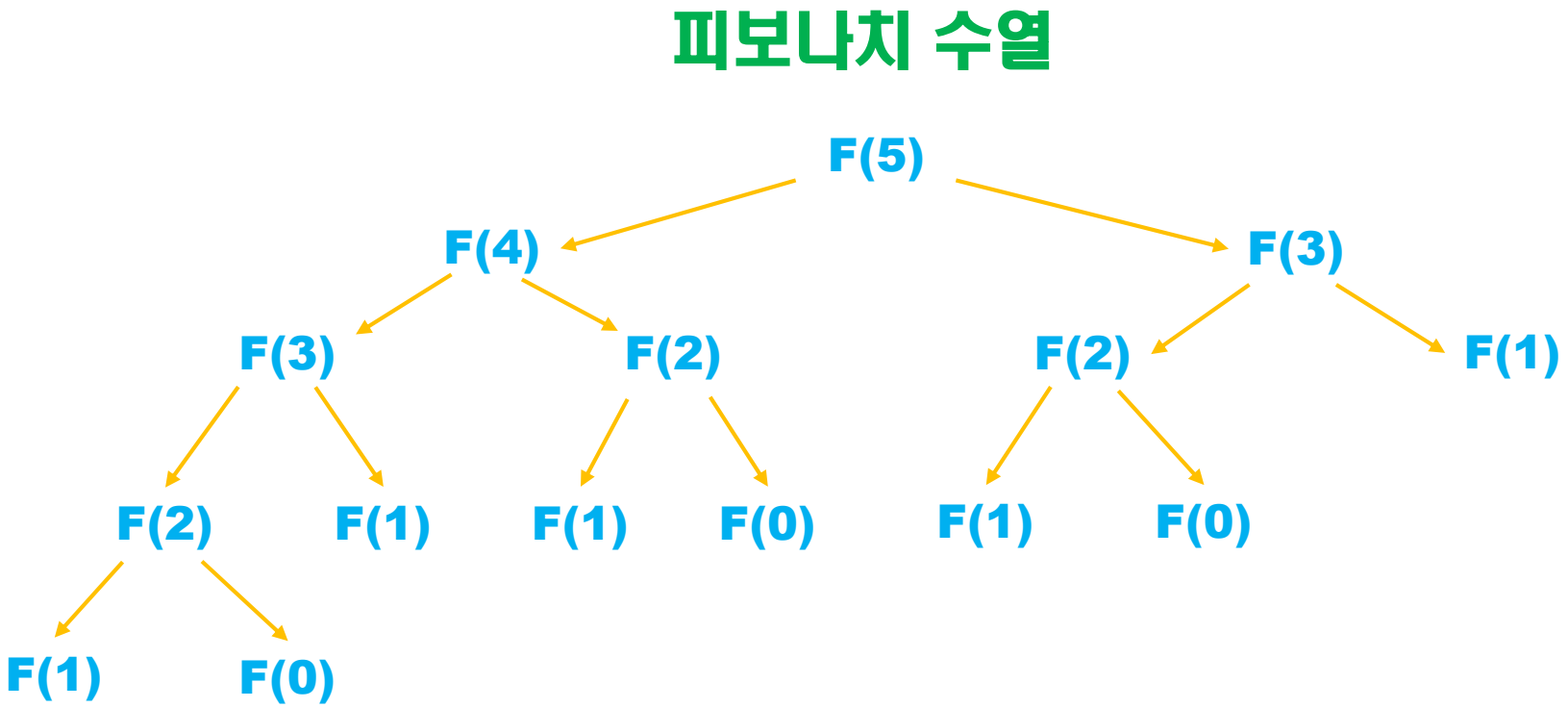
↓

EX: 피보나치 수열

ALOHA
The algorithm club.

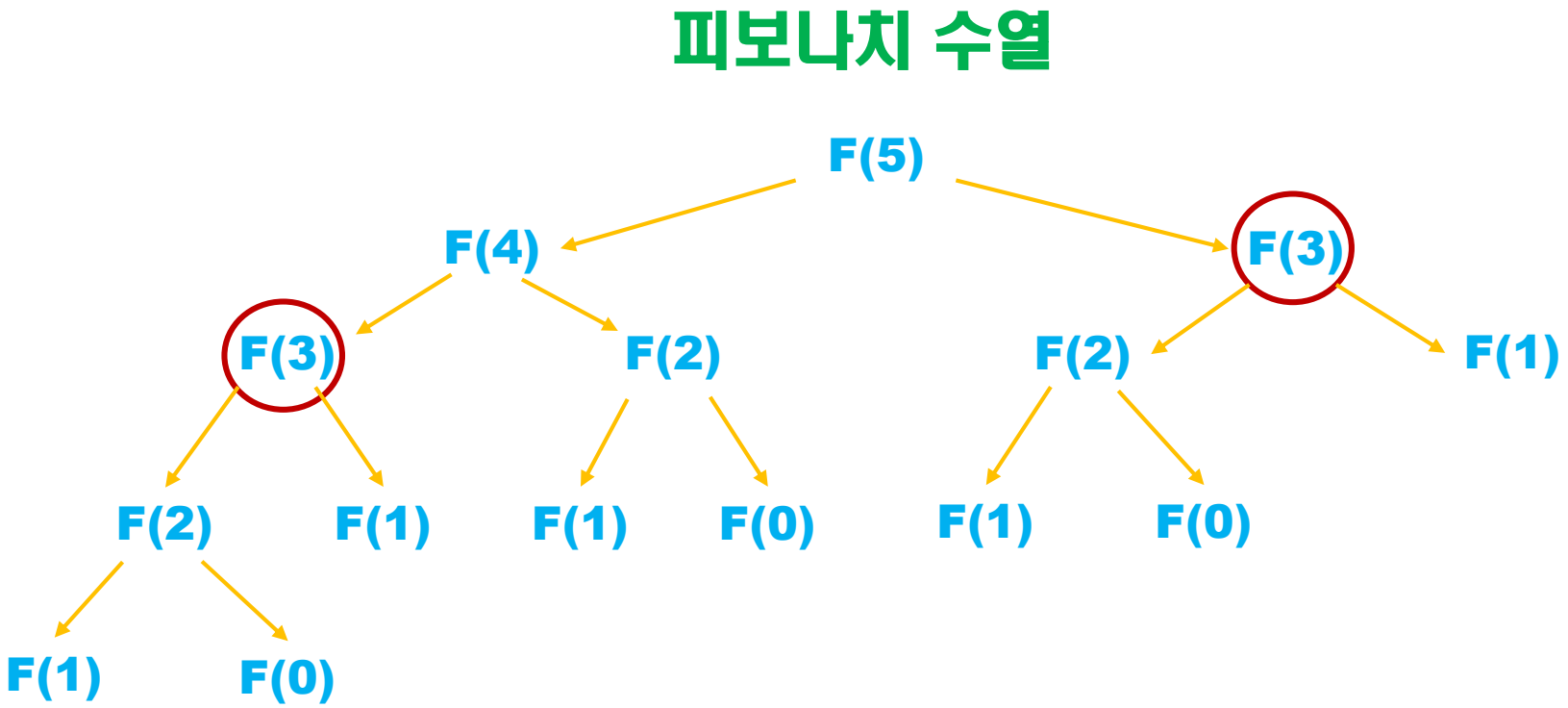
Dynamic Programming 이란?

동적 계획법



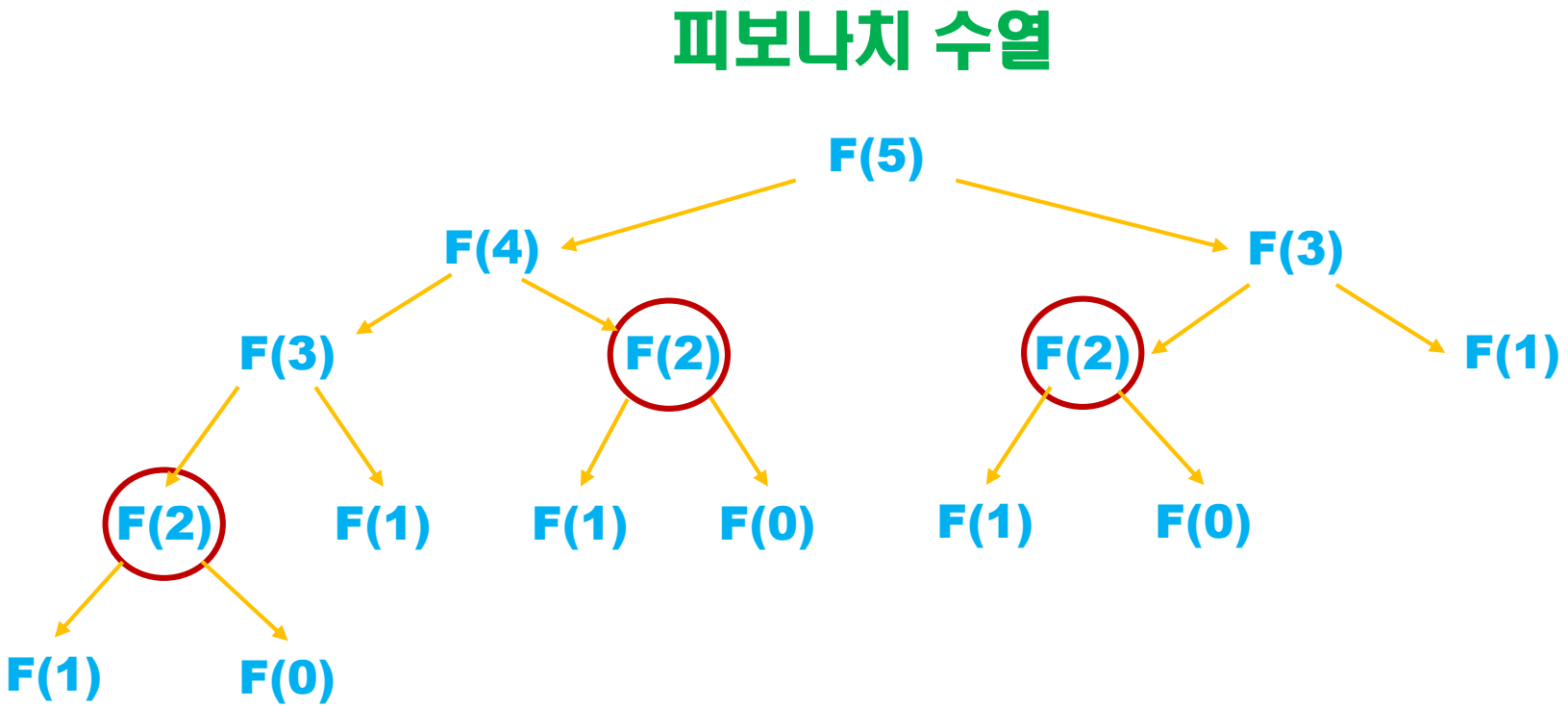
Dynamic Programming 이란?

동적 계획법



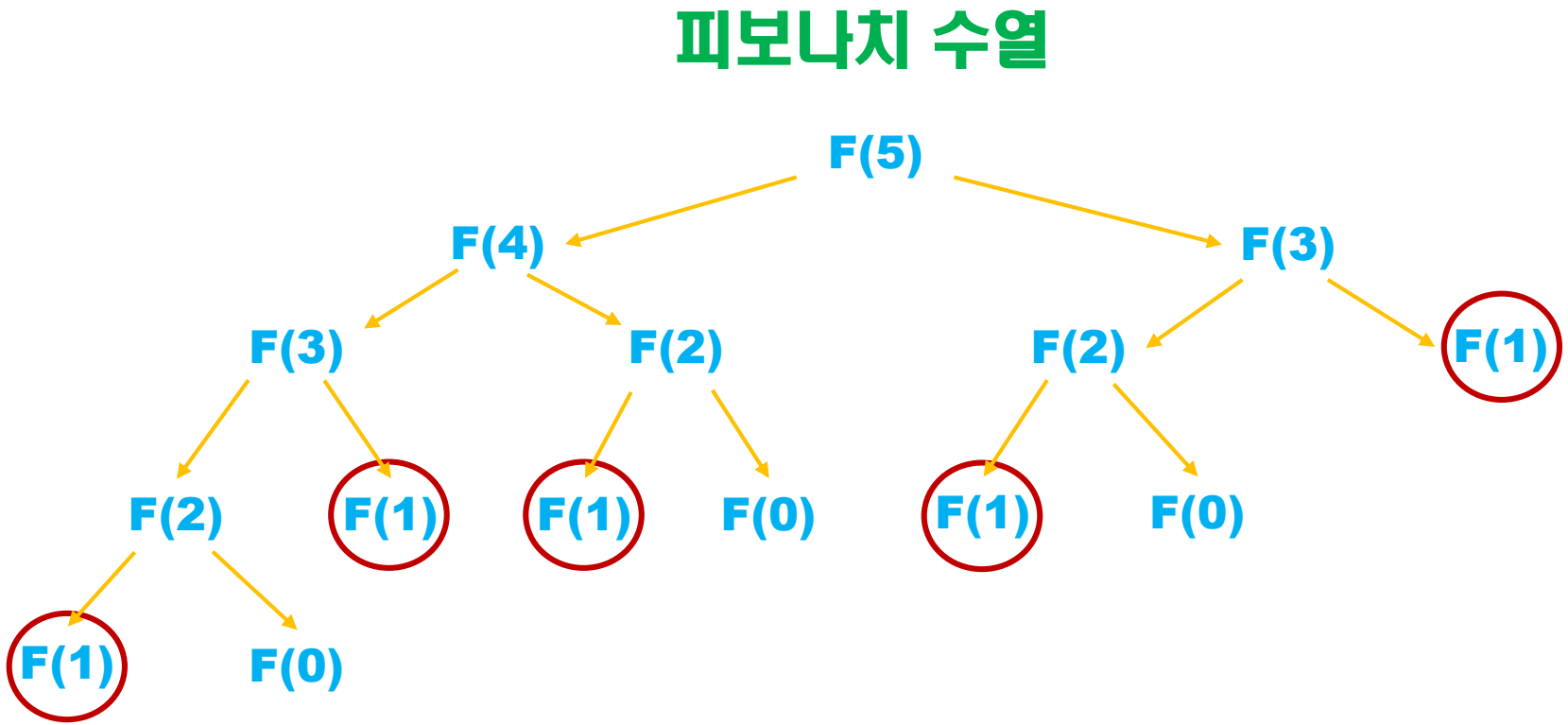
Dynamic Programming 이란?

동적 계획법



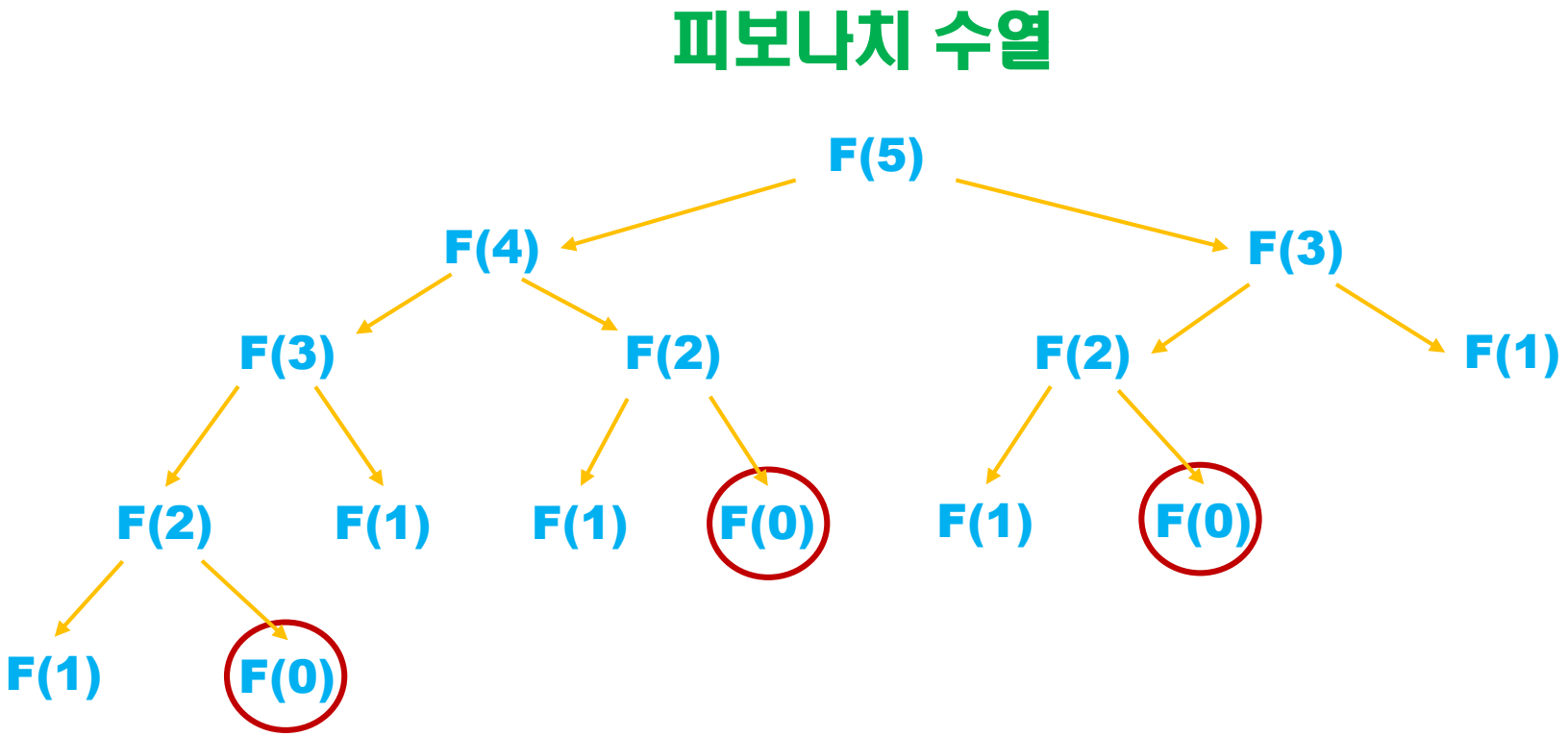
Dynamic Programming 이란?

동적 계획법



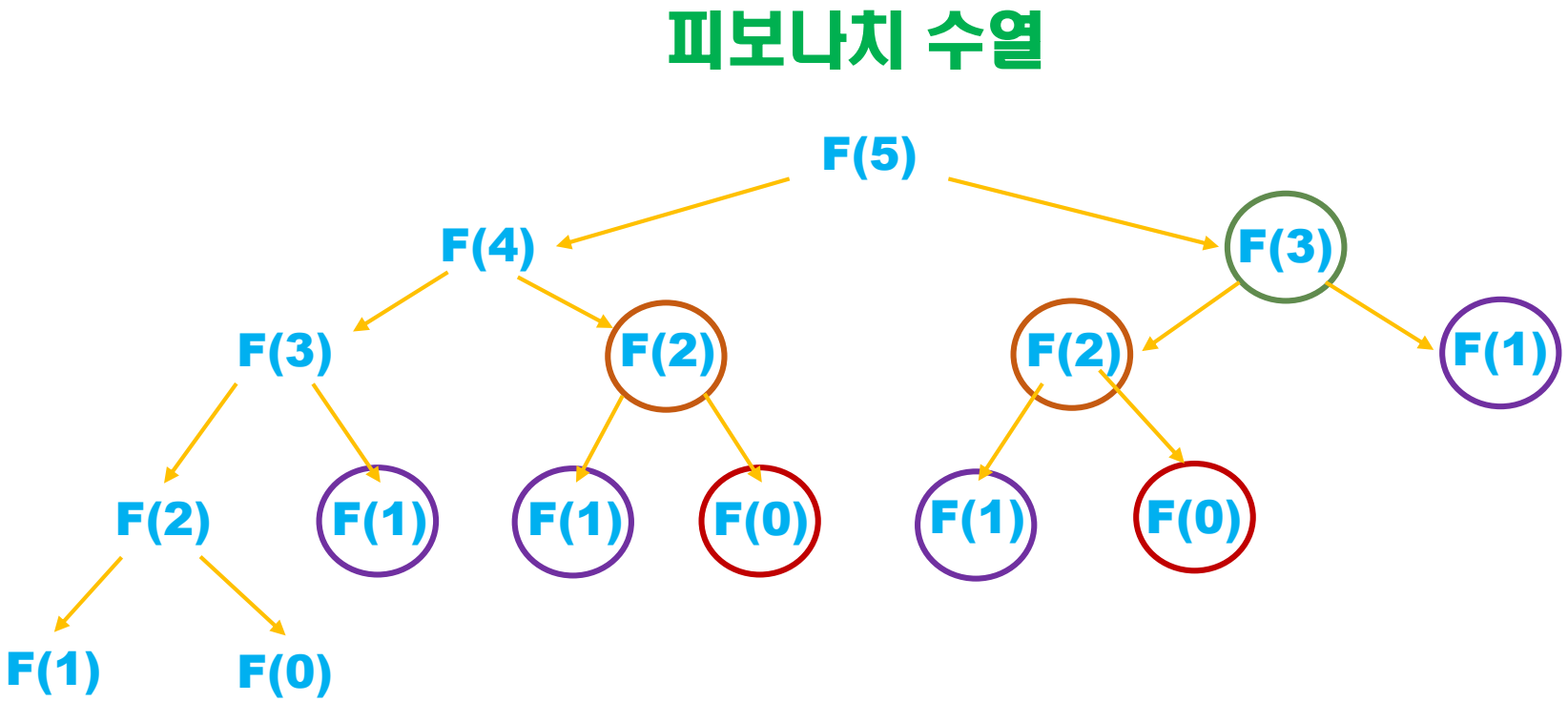
Dynamic Programming 이란?

동적 계획법



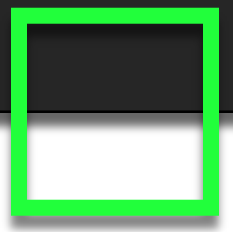
Dynamic Programming 이란?

동적 계획법



#2

Top-Down



Top – Down 해법?

큰 문제에서 작은 문제로

1. 문제를 작은 문제들로 나눈다.

Top – Down 해법?

큰 문제에서 작은 문제로

1. 문제를 작은 문제들로 나눈다.

2. 작은 문제들을 푼다

Top – Down 해법?

큰 문제에서 작은 문제로

1. 문제를 작은 문제들로 나눈다.

2. 작은 문제들을 푼다

3. 작은 문제들의 답으로 전체 문제를 푼다

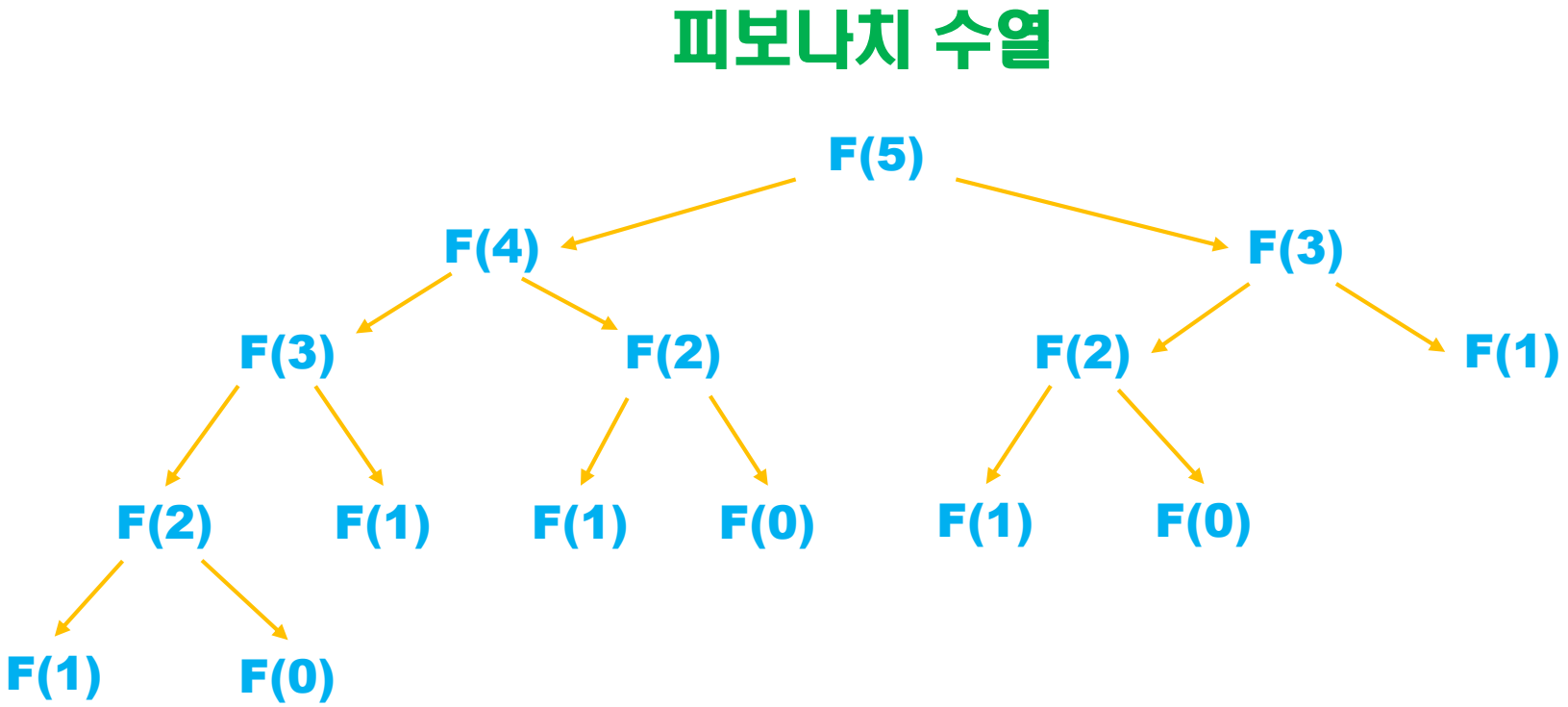
Top – Down 해법?

큰 문제에서 작은 문제로

예를 들어!!

Top – Down 해법?

큰 문제에서 작은 문제로



Top – Down 해법?

큰 문제에서 작은 문제로

2가지 조건!!!

Top – Down 해법?

큰 문제에서 작은 문제로

2가지 조건!!!

Overlapping Subproblem

+

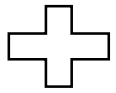
Optimal Structure

Top – Down 해법?

큰 문제에서 작은 문제로

2가지 조건!!!

Overlapping Subproblem



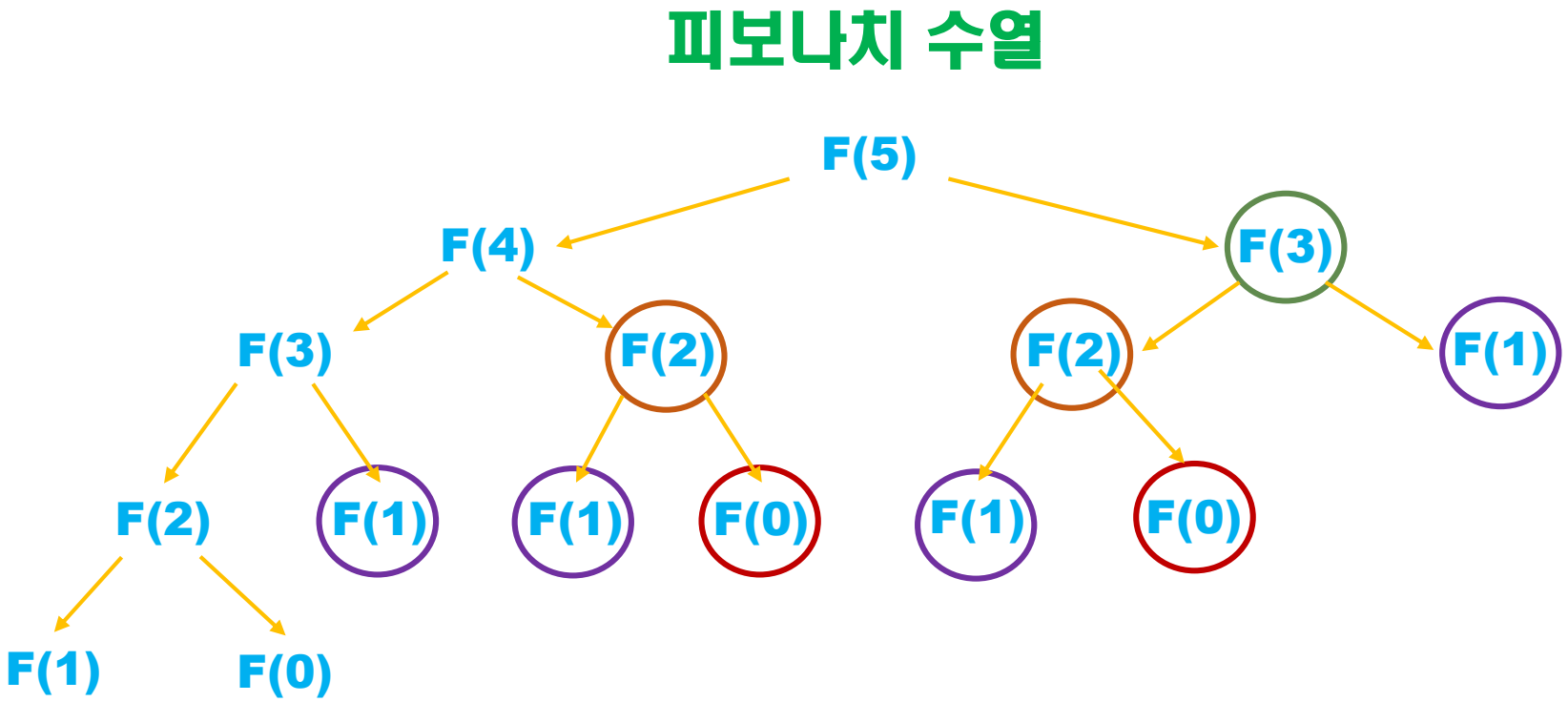
Optimal Structure

만족하는가?

ALOHA
The algorithm club.

Top – Down 해법?

큰 문제에서 작은 문제로



Top – Down 해법?

큰 문제에서 작은 문제로

같은 문제를 여러 번!!

ALOHA
The algorithm club.

Top – Down 해법?

큰 문제에서 작은 문제로

그럼 어떻게 해...??

Top – Down 해법?

큰 문제에서 작은 문제로

이미 해결한 값을 기억하자!!

Top – Down 해법?

큰 문제에서 작은 문제로

Memoization

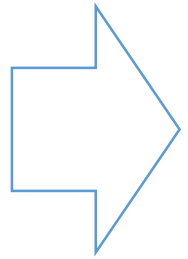
계산된 값을 버리지 말고 기억하자!!
값을 배열에 저장하여 중복으로 쓰이는 경우
다시 계산할 필요없이
가져와서 쓰자!!

Top – Down 해법?

큰 문제에서 작은 문제로

No Memoization

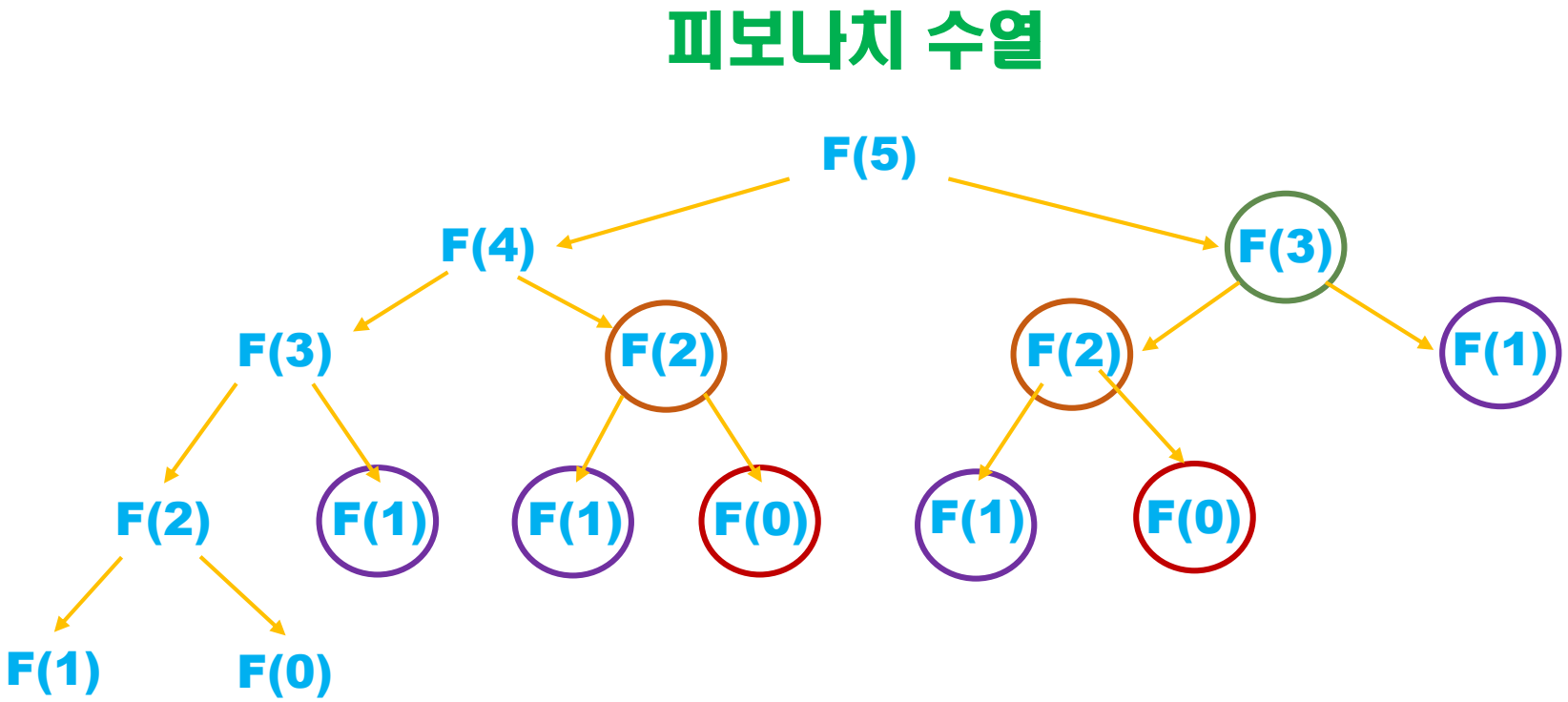
```cpp
#include <iostream>
#include <cstdio>
using namespace std;
long long find_fibo(int num) {
    if (num <= 0)
        return 0;
    else if (num == 1)
        return 1;
    else
        return find_fibo(num - 1) + find_fibo(num - 2);
}
int main(void) {
    int N;
    scanf("%d", &N);
    long long out;
    out = find_fibo(N);
    printf("%d₩n", out);
    return 0;
}
```



이렇게
작성한다면??

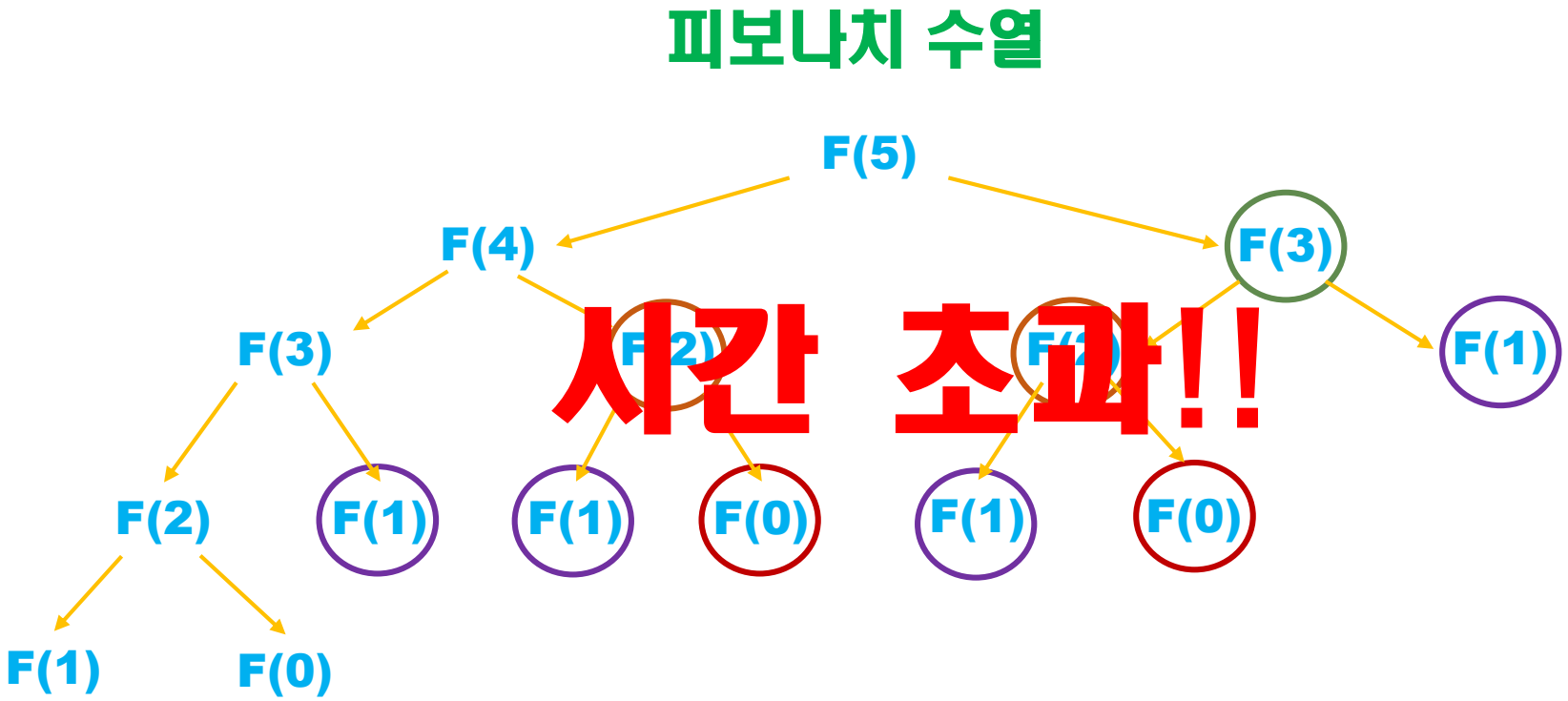
Top – Down 해법?

큰 문제에서 작은 문제로



Top – Down 해법?

큰 문제에서 작은 문제로



Top – Down 해법?

큰 문제에서 작은 문제로

Use Memoization

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
long long memo[50];
long long find_fibo(int num) {
    if (memo[num] > 0)
        return memo[num];
    if (num <= 0)
        return 0;
    else if (num == 1)
        return 1;
    else
        return memo[num] = find_fibo(num - 1) + find_fibo(num - 2);
}
int main(void) {
    int N;
    memo[0] = 0;memo[1] = 1;
    scanf("%d", &N);
    long long out;
    out = find_fibo(N);
    printf("%d₩n", out);
    return 0;
}
```

ALOHA
The algorithm club.

Top – Down 해법?

큰 문제에서 작은 문제로

Use Memoization

```
#include <iostream>
#include <cstdio>
using namespace std;
long long memo[50];
long long find_fibo(int num) {
    if (memo[num] > 0)
        return memo[num];
    if (num <= 0)
        return 0;
    else if (num == 1)
        return 1;
    else
        return memo[num] = find_fibo(num - 1) + find_fibo(num - 2);
}
int main(void) {
    int N;
    memo[0] = 0;memo[1] = 1;
    scanf("%d", &N);
    long long out;
    out = find_fibo(N);
    printf("%d₩n", out);
    return 0;
}
```

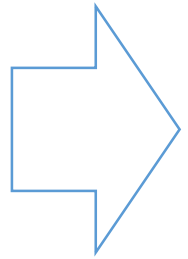
ALOHA
The algorithm club.

Top – Down 해법?

큰 문제에서 작은 문제로

Use Memoization

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
long long memo[50];
long long find_fibo(int num) {
    if (memo[num] > 0)
        return memo[num];
    if (num <= 0)
        return 0;
    else if (num == 1)
        return 1;
    else
        return memo[num] = find_fibo(num - 1) + find_fibo(num - 2);
}
int main(void) {
    int N;
    memo[0] = 0;memo[1] = 1;
    scanf("%d", &N);
    long long out;
    out = find_fibo(N);
    printf("%d₩n", out);
    return 0;
}
```

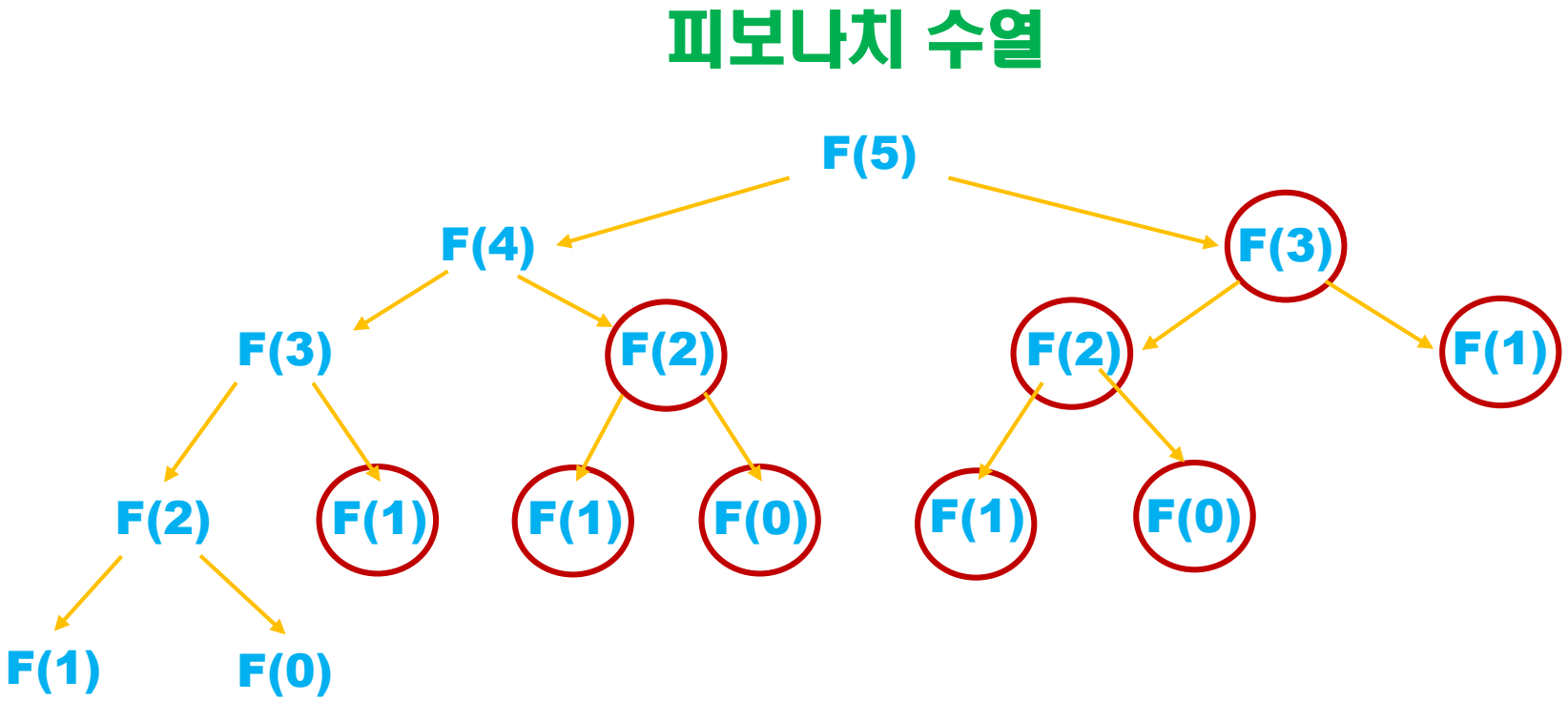


이렇게
작성한다면??

ALOHA
The algorithm club.

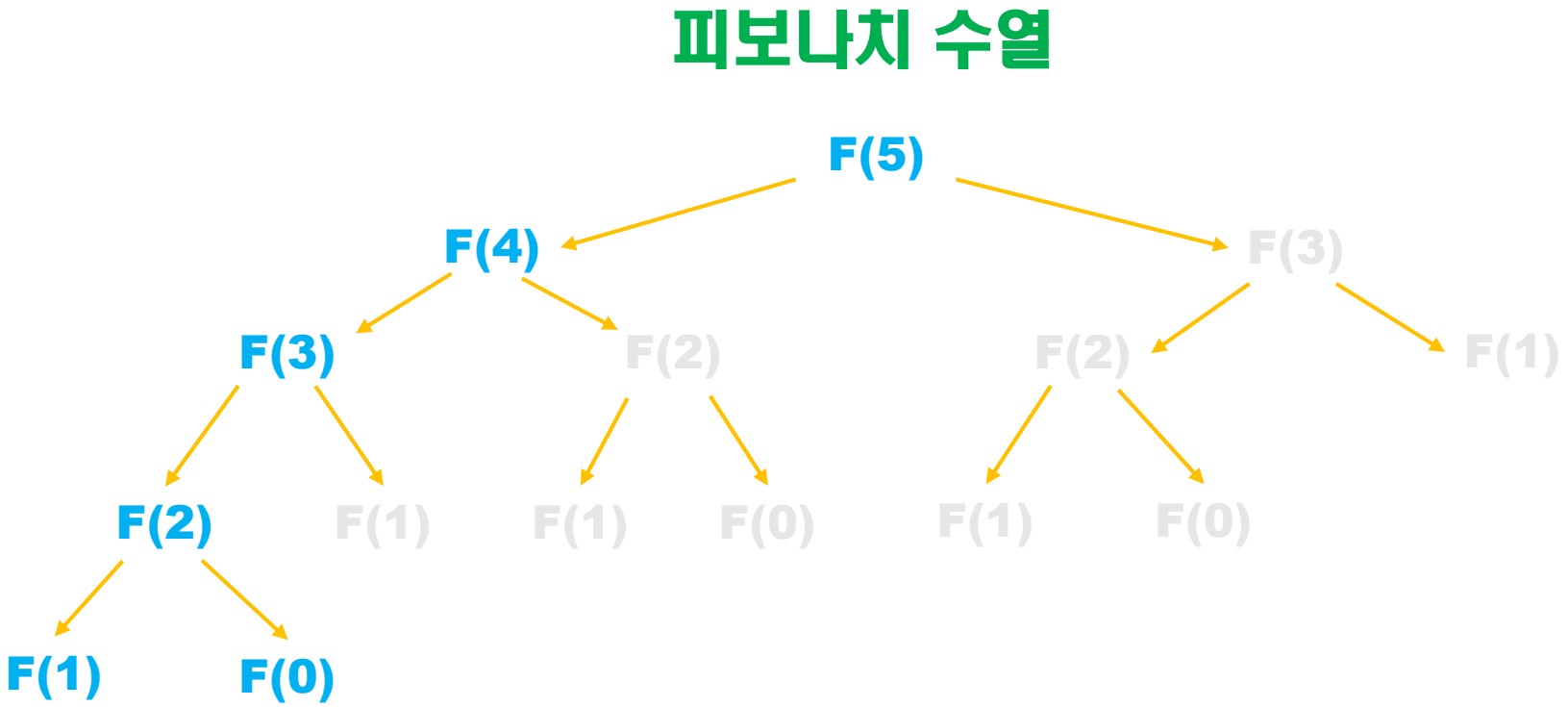
Top – Down 해법?

큰 문제에서 작은 문제로



Top – Down 해법?

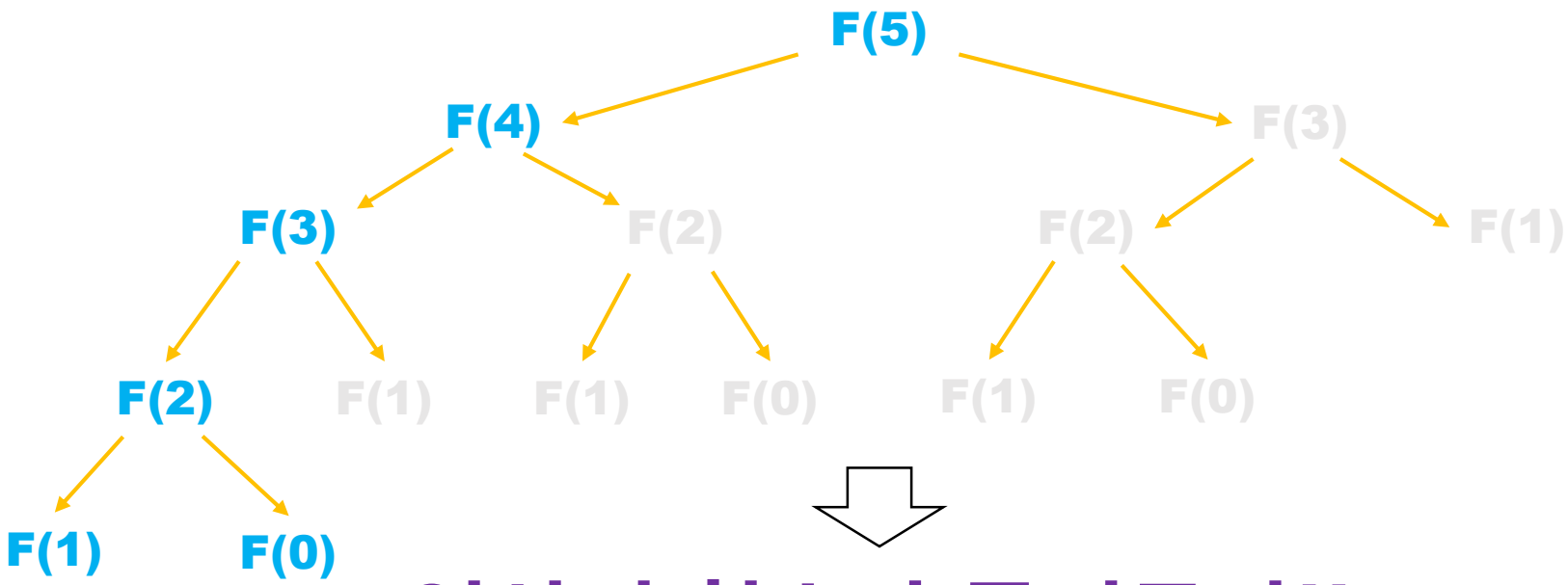
큰 문제에서 작은 문제로



Top – Down 해법?

큰 문제에서 작은 문제로

피보나치 수열



연산의 횟수가 줄어든다!!

연습 문제

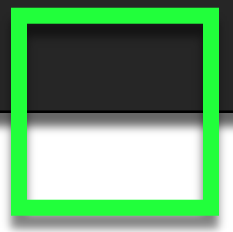
BOJ 1003
피보나치 함수

연습 문제

BOJ 1149
RGB 거리

#3

Bottom-Up



Bottom - Up 해법?

큰 문제에서 작은 문제로

1. 가장 작은 문제부터 푼다.

Bottom - Up 해법?

큰 문제에서 작은 문제로

1. 가장 작은 문제부터 푼다.

2. 문제의 크기를 점점 크게 만들어서 전체 문제를 푼다.

Bottom - Up 해법?

Sliding Window

피보나치 수열

$F(0) = 0;$
$F(1) = 1;$

Bottom - Up 해법?

Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;

Bottom - Up 해법?

Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;

Bottom - Up 해법?

Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;

Bottom - Up 해법?

Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;

Bottom - Up 해법?

Sliding Window

피보나치 수열

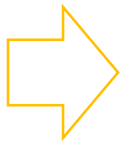
F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;
.....
.....
.....
F(N) = F(N-1) + F(N-2)

Bottom - Up 해법?

Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;
.....
.....
.....
F(N) = F(N-1) + F(N-2)



1. F(0), F(1) 부터 시작
   -> 작은 문제부터 시작
2. 다음 단계로 넘어가면서 N까지 구하기
   -> 크기를 키워서
   -> 전체 문제에 도달

Bottom - Up 해법?

큰 문제에서 작은 문제로

피보나치 수열

```
#include <iostream>
#include <cstdio>
using namespace std;
long long fib[50];
void find_fibo(int num) {
    int i;
    fib[0] = 0;fib[1] = 1;
    for (i = 2;i <= num;i++) {
        fib[i] = fib[i - 1] + fib[i - 2];
    }
}
int main(void) {
    int N;
    scanf("%d", &N);
    find_fibo(N);
    printf("%d₩n", fib[N]);
    return 0;
}
```

연습 문제

BOJ 1003
피보나치 함수

연습 문제

BOJ 11726
2xN 타일링

연습 문제

BOJ 2193
이친수

Bottom - Up 해법?

Sliding Window

Sliding Window

Bottom - Up 해법?

Sliding Window

- Window: 내가 지금 보고 있는 범위 DP 배열로 저장될 영역

- 실질적으로 연산에 필요한 데이터만 저장하고 있으면, 메모리를 절약할 수 있다.

- 메모리 제한이 작거나, 배열로 선언하면 메모리 초과가 날 것 같을 때
  -> 사용해보자!!

Bottom - Up 해법?

Sliding Window

피보나치 수열

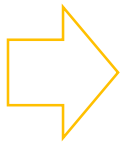
F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;
.....
.....
.....
F(N) = F(N-1) + F(N-2)

Bottom - Up 해법?

Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;
.....
.....
.....
F(N) = F(N-1) + F(N-2)



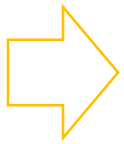
이렇게 0~N까지 배열을
만들어 저장할 필요가??

Bottom - Up 해법?

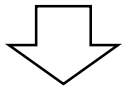
Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;
.....
.....
.....
F(N) = F(N-1) + F(N-2)



이렇게 0~N까지 배열을
만들어 저장할 필요가??



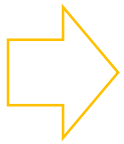
없다!!

Bottom - Up 해법?

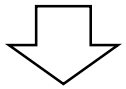
Sliding Window

피보나치 수열

F(0) = 0;
F(1) = 1;
F(2) = F(1) + F(0) = 1 + 0 = 1;
F(3) = F(2) + F(1) = 1 + 1 = 2;
F(4) = F(3) + F(2) = 2 + 1 = 3;
F(5) = F(4) + F(3) = 3 + 2 = 5;
.....
.....
.....
F(N) = F(N-1) + F(N-2)



이렇게 0~N까지 배열을
만들어 저장할 필요가??



없다!!

N번째 수를 구하기 위해서는 N-1, N-2번째의 수가 필요하므로,
실제로 연산에 필요한 배열의 사이즈는 2

ALOHA
The algorithm club.

Bottom - Up 해법?

Sliding Window

피보나치 수열

```cpp
#include <iostream>
#include <cstdio>
using namespace std;
int main(void) {
    int N, N_1 = 1, N_2 = 0, n, i;
    scanf("%d", &n);
    for (i = 2; i <= n; i++) {
        N = N_1 + N_2;
        N_2 = N_1;
        N_1 = N;
    }
    printf("%d₩n", N);
    return 0;
}
```

ALOHA
The algorithm club.

연습 문제

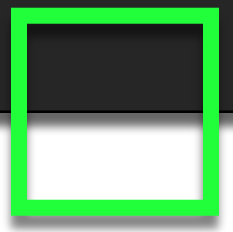
BOJ 1003
피보나치 함수

연습 문제

BOJ 2096
내려가기

#4

추가 문제들



연습 문제

BOJ 11727
2xN 타일링 2

연습 문제

BOJ 11057
오르막 수

연습 문제

BOJ 10844
쉬운 계단 수

연습 문제

BOJ 9461
파도반 수열

수고했어요!