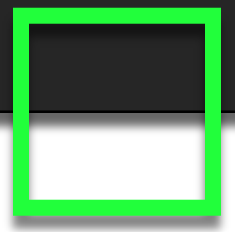
#C언어 반
#5주차

DP기초

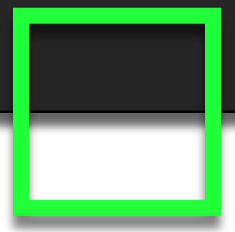
T. 강민구
Asst. 이준규 김재형



ALOHA
The algorithm club.

#1

DP가 무엇인가요?



DP가 무엇인가요?

-여러 단계에 걸친 복잡한 문제의 해결을 위한 동적 계획법이에요!

-작은(하위) 문제를 해결하여 큰(상위) 문제를 해결해 나가는 방식이에요!
(여기서 Memoization 기법을 이용하기때문에 같은 계산을 반복할 필요가 없어요^^)

-알고 나면 점화식과 비슷하다는 느낌을 받을거에요!

-이 기법을 사용하면 Time Limit, Memory Limit 등의 문제를 해결할 수 있어요!

문제 해결방식

-복잡한 문제를 해결하는 방법에는 Top-Down방식과 Bottom-Up방식 이렇게 두 가지가 있어요!

-Top-Down 방식은 위에서 내려오는 것, 즉, 큰 문제부터 시작해서 계속 작은 문제로 분할해 가면서 푸는 것이에요. 재귀함수가 주된 방법이 되는 방식이에요.

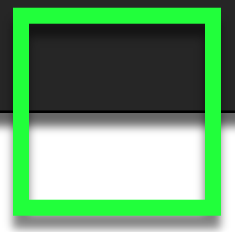
-Bottom-Up 방식은 바닥에서 올라오는 것, 즉, 작은 문제부터 시작해서 작은 문제를 점점 쌓아 큰 문제를 푸는 것이에요. 주로 배열을 사용하여 문제를 풀게돼요.

Memoization?

-어떤 단계에서 이미 계산해 나온 값을 배열에 저장해 놓는 기법이에요!
이러한 작업을 해주면, 나중에 그 단계의 값이 필요할 때 다시 계산을 해줄 필요가 없어요!
(배열에서 값을 가져오는 작업보다 계산하는 시간이 더 오래 걸리기 때문에 사용해요!)

#2

Fibonacci Sequence

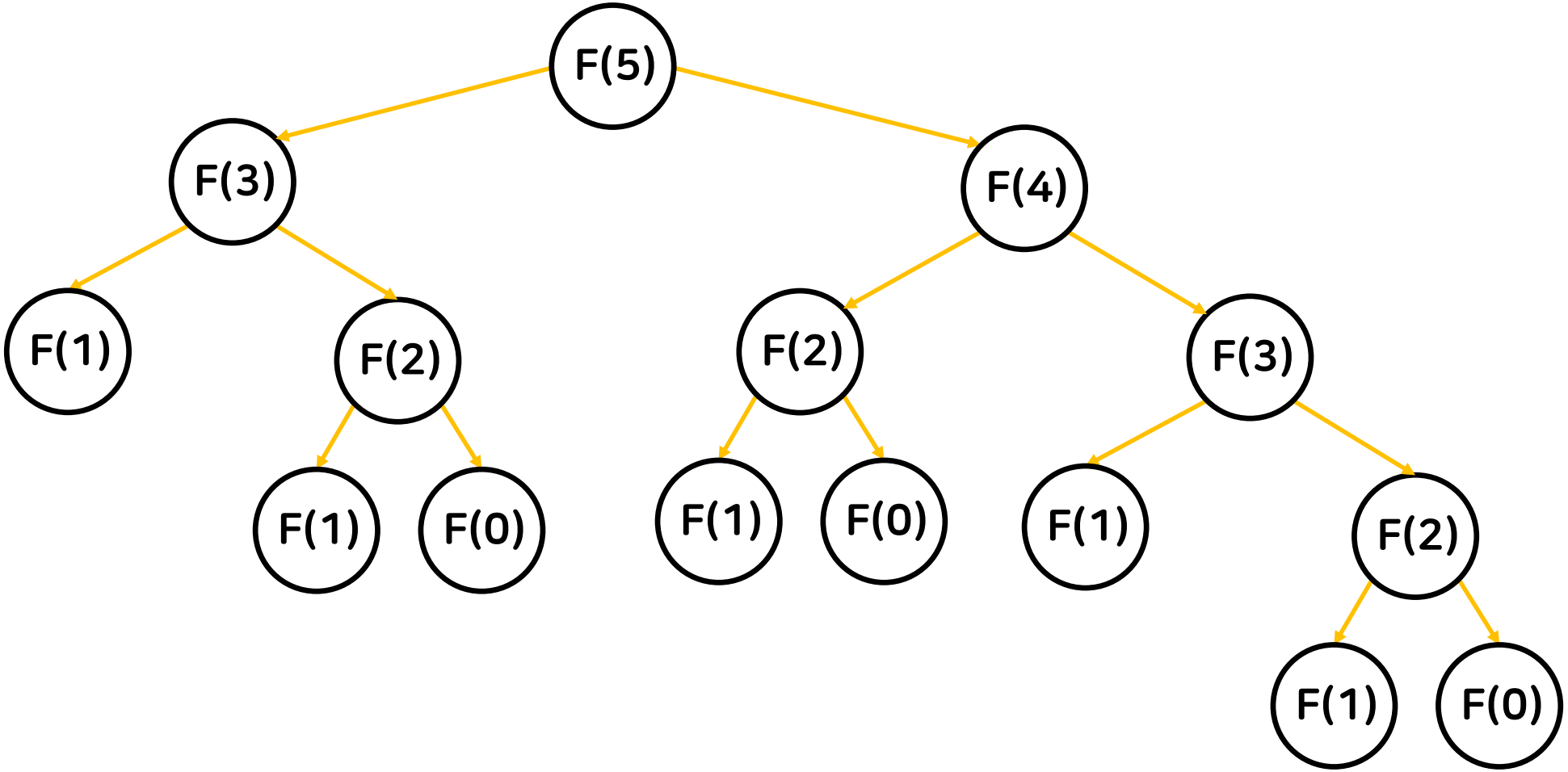


ALOHA
The algorithm club.

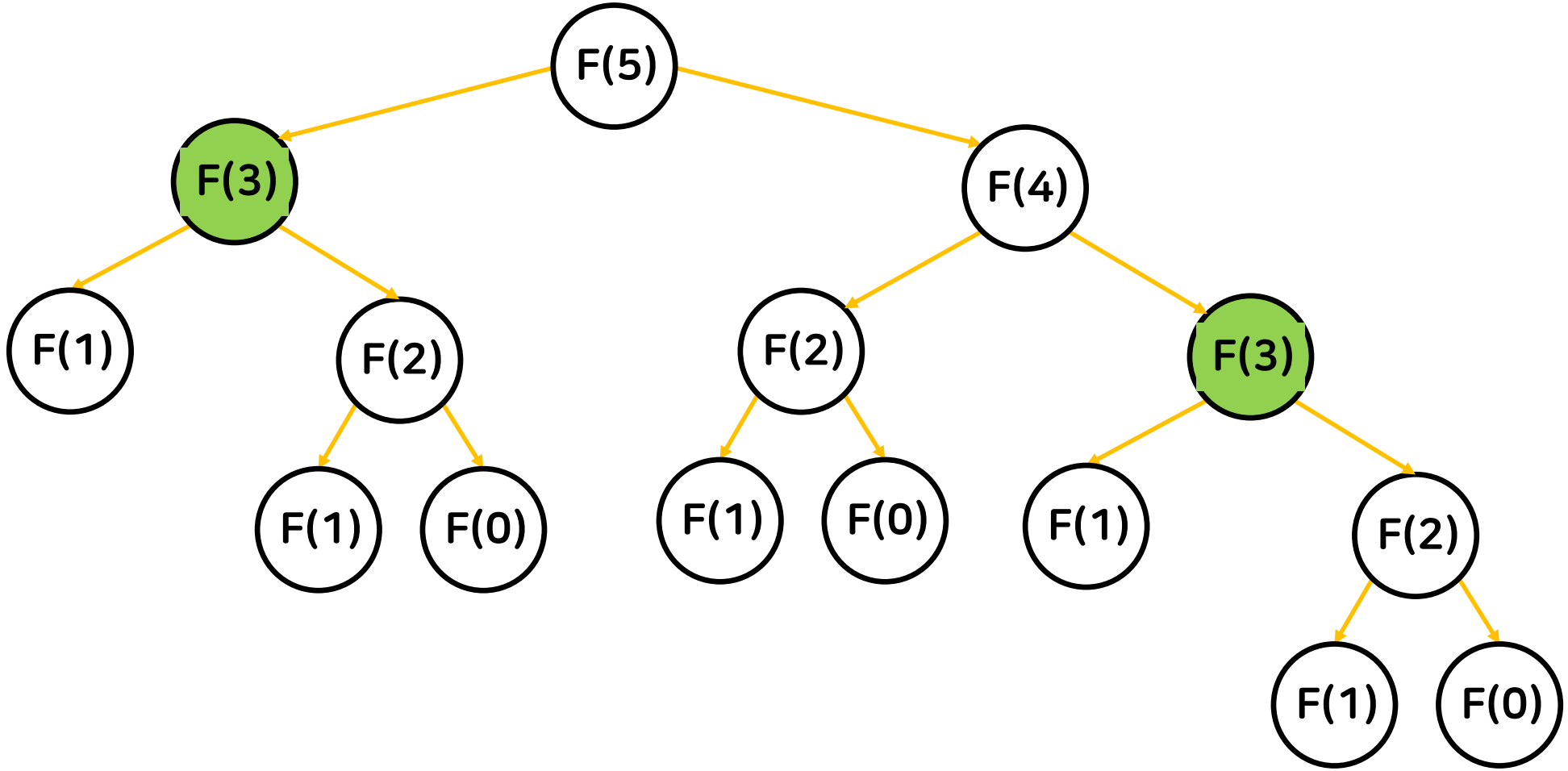
피보나치 수열(재귀함수 활용)

```c
#include <stdio.h>

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

int main()
{
        int n;
        scanf(“%d”, &n);
        printf(“%d\n”, fibonacci(n));
        return 0;
}
```

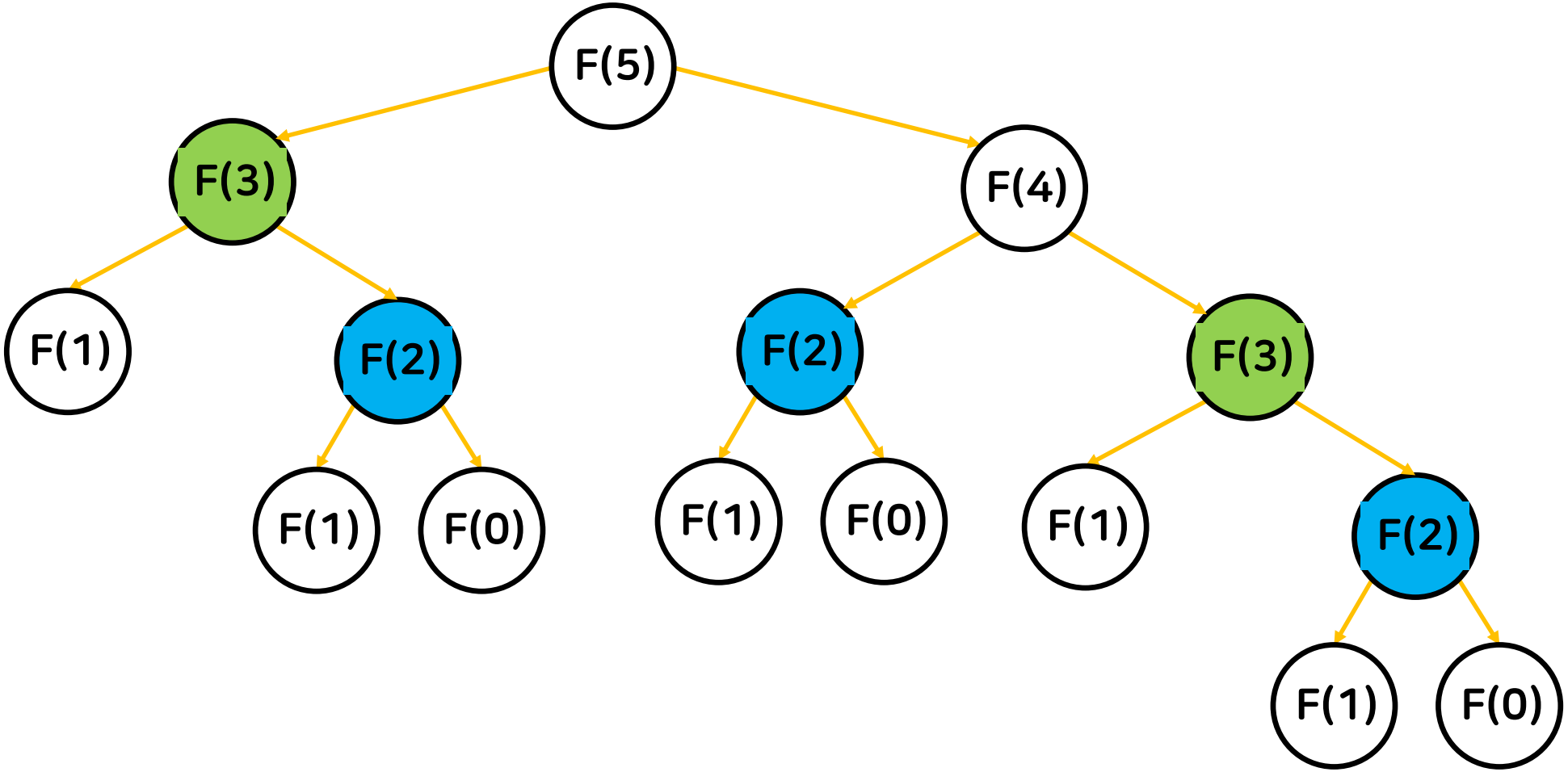
피보나치 수열(재귀함수 활용)



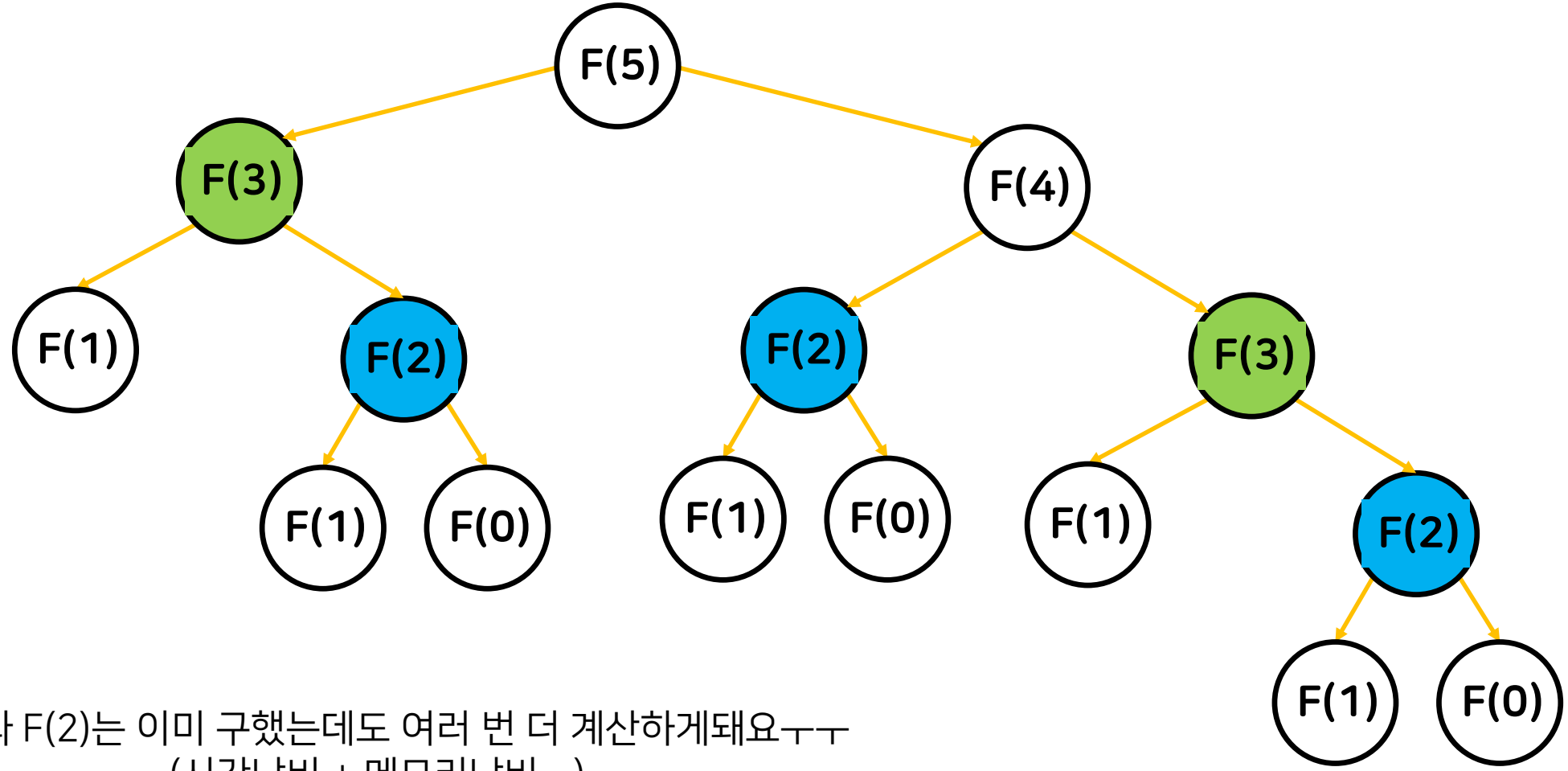
피보나치 수열(재귀함수 활용)



피보나치 수열(재귀함수 활용)



피보나치 수열(재귀함수 활용)



-> F(3)과 F(2)는 이미 구했는데도 여러 번 더 계산하게돼요ㅜㅜ
(시간낭비 + 메모리낭비…)

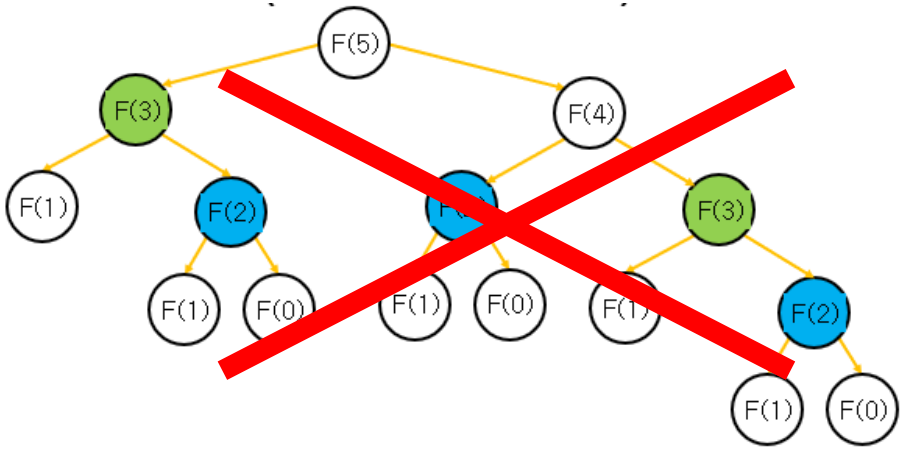
피보나치 수열(Memoization 활용)

-Memoization :어떤 단계에서 이미 계산해 나온 값을 배열에 저장해 놓는 기법이었어요!
이러한 작업을 해주면, 나중에 그 단계의 값이 필요할 때 다시 계산을 해줄 필요가 없어요!
(배열에서 값을 가져오는 작업보다 계산하는 시간이 더 오래 걸리기 때문에 사용해요!)

피보나치 수열(Memoization 활용)

```c
#include <stdio.h>
int cache[10] = {0, };//cache 배열에 저장함에 따라 다시 계산을 하지 않아도 됩니다.
int fibonacci(int n)
{
        if(n == 1 || n == 2) return 1;
        if(n == cache[n]) return cache[n];//저장된 값을 return
        else return cache[n] = fibonacci(n - 1) + fibonacci(n - 2);
        //Top-Down을 이용하여 계산 후 저장
}
int main()
{
        int n;
        scanf("%d", &n);
        printf("%d\n", fibonacci(n));
        return 0;
}
```

피보나치 수열(Bottom-Up 활용)



Index	0	1	2	3	4	5
value	0	1	1	2	3	5

또 다른 방법으로 Bottom-Up 방법으로 해결할 수 있어요.

값을 저장할 배열의 이름을 fibo라고 가정해봐요!

F(3), F(2)의 값이 각각 fibo[3], fibo[2]에 이미 저장되어 있었다면?

F(4) = fibo[4] = fibo[3] + fibo[2]
->저장된 F(3)값과 F(2)값을 이용하여 훨씬 빠르게 F(4)를 구할 수 있어요!
->값을 계산해줬으니 fibo[4]에 F(4)값을 저장해줘요!

F(5) = fibo[5] = fibo[4] + fibo[3]
->F(4)값을 이전에 fibo[4]에 저장해줬으니 훨씬 빠르게 F(5)를 구할 수 있어요!
->이번에도 fibo[5]에 F(5)값을 저장을 해줘요!

피보나치 수열(Bottom-Up 활용)

```
#include <stdio.h>

int fibonacci[100] = {0, 1};

int main()
{
        int n;
        scanf("%d", &n);

        for ( int i = 2; i <= n; i++ )
                fibonacci[ i ] = fibonacci[ i - 1 ]+fibonacci[ i - 2 ];

        printf("%d\n", fibonacci[n]);
        return 0;
}
```

피보나치 수열(Bottom-Up 활용)

Top-Down보다 훨씬 빨라요!

연습 문제

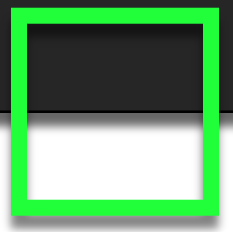
BOJ 2193
이친수

연습 문제

BOJ 11726
2xn 타일링

#3

Combination



조합(재귀함수 활용)

고등학교 확통시간에 배웠던… 조합 기억나시죠??
구현하는 방법은 다양하겠지만

일단!

C(n, r) = n! / (r! * (n – r)!)

이식이 제일 먼저 떠오르실거에요!

팩토리얼은…재귀함수로 구현을 할 수 있어요.

.

.

.

아까 재귀함수는 시간 오래걸린다는거 배웠죠?
심지어 팩토리얼은 계산과정에서
Overflow가 발생할 수 있어요! ㅎㄷㄷ
그럼 저 식은 효율적이진 않네요ㅜㅜ

조합(???)

.
.
.
우리가 지금 배우는게 DP…
아까 언젠가 누군가 갑자기 DP가 점화식과 관련이 있다고 했는데…
지금은 조합을 구현을 할 건데…
.
.
.

조합(???)

아!!! 고등학교 수업시간에 배웠던 점화식!!!

C(n, r) = C(n-1, r) + C(n-1, r-1)!!

맞아요 바로 그거에요. 고등학교를 열심히 다녔나보네요.^^
뭐라구요? 저거 파스칼삼각형으로 할 수 있는거 아니냐구요?

…당신은 천재에요!

이제 조합을 DP로 구현할 수 있게 되었어요!
파스칼 삼각형을 구현하면 조합을 구현한거나 마찬가지겠네요!

파스칼삼각형(Bottom-Up 활용)

```
#include <stdio.h>

int pascal[100][100];

int main()
{
        int a, b;
        scanf("%d%d", &a, &b);
        pascal[1][0] = 1;
        pascal[1][1] = 1;
        for ( int i = 2; i <= a; i++ )
                for ( int j = 0; j <= i; j++ )
                        pascal[i][j] = pascal[i – 1][j – 1] + pascal[i - 1][j];
        printf("%d\n", pascal[a][b]);
        return 0;
}
```

조합(Bottom-Up 활용)

```
#include <stdio.h>

int c[100][100];

int main()
{
        int n, r;
        scanf("%d%d", &n, &r);
        c[1][0] = 1;
        c[1][1] = 1;
        for ( int i = 2; i <= n; i++ )
                for ( int j = 0; j <= i; j++ )
                        c[i][j] = c[i – 1][j – 1] + c[i - 1][j];
        printf("%d\n", c[n][r]);
        return 0;
}
```

조합(Bottom-Up 활용)

재귀보다 훨씬 빨라요!

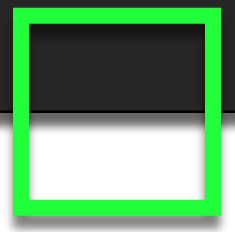
연습 문제

BOJ 11050
이항계수1

연습 문제

BOJ 11051
이항계수2

Q & A



연습 문제

BOJ 11727
2xn 타일링 2

연습 문제

BOJ 1904
01타일

수고했어요!