6주차 : DP기초

강사 : 심기호

챕터 0: DP

Dynamic Programming에 대해

DP란?

- Dynamic Programming의 줄임말!(동적 계획법)
- 복잡한 문제를 간단한 여러 개의 문제로 나누어 푸는 방법입니다!
- 하위 문제로 나누어 푼 다음 그 것을 종합하여 큰 문제를 해결한다!

~~(되게 추상적인 말들이죠? 뒤에 자세하게 설명이 나올거에요~)!~~

챕터 1: DP의 방법

피보나치 함수를 통해 이해해보자!

DP의 방법

- 큰 문제를 작은 문제로 분할해서 푼다고 하였는데!

- 그 분할하는 방법이 두 종류가 있습니다!

  - Bottom-Up(작은 문제를 구해서 쌓아 올리는..)
    - 순차적일 때 유리
    - 상대적으로 빠름
    - 주로 표의적으로 구현됨

  - Top-Down(재귀함수를 통해 큰 문제에서 작은 문제로 들어가는..)
    - 비순차적일 때 유리
    - 상대적으로 느림
    - 주로 *Memoization*으로 구현됨

Fibonacci Sequence

- 피보나치 수열을 통해 DP의 두가지 방식을 이해해 볼 건데요!
그 전에 PPT 구조에 대해 알고 갑시다!

- 1. 피보나치 값을 동적계획법이 아닌 방식으로 구하기

- 2. 피보나치 값을 Top-Down 방식으로 구하기

- 3. 피보나치 값을 Bottom-Up 방식으로 구하기

Fibonacci Sequence-**Without DP**

- 피보나치 함수는
현재 항을 구하려면 **전 항**과 **그 전 항**을 구해서 더해야 한다…
이 말을 말 그대로 코드로 구현한다면..?(점화식 느낌)

Fibonacci Sequence-**Without DP**

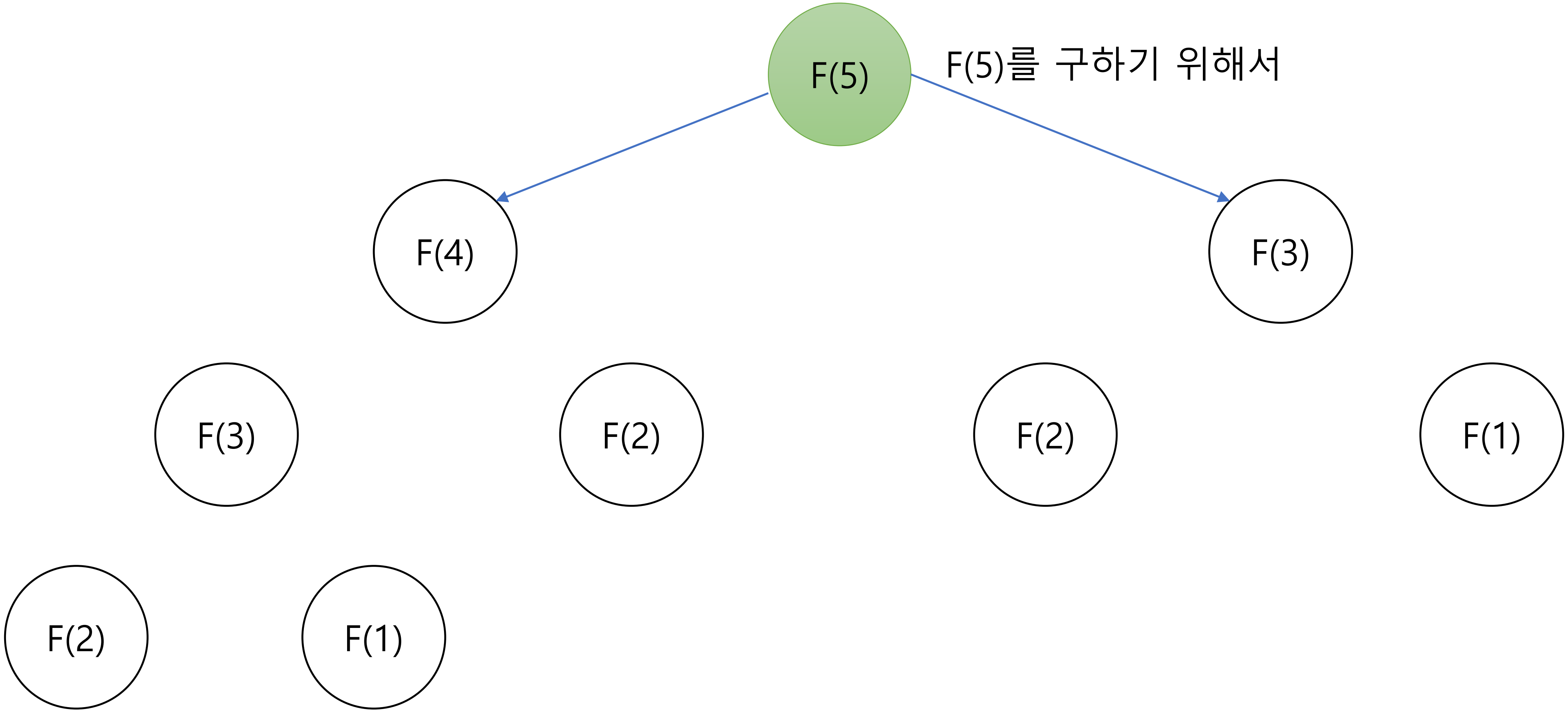
```
int Fibo(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;

    return Fibo(n - 1) + Fibo(n - 2);
}
```

전 항과 **그 전 항**을 더한다!

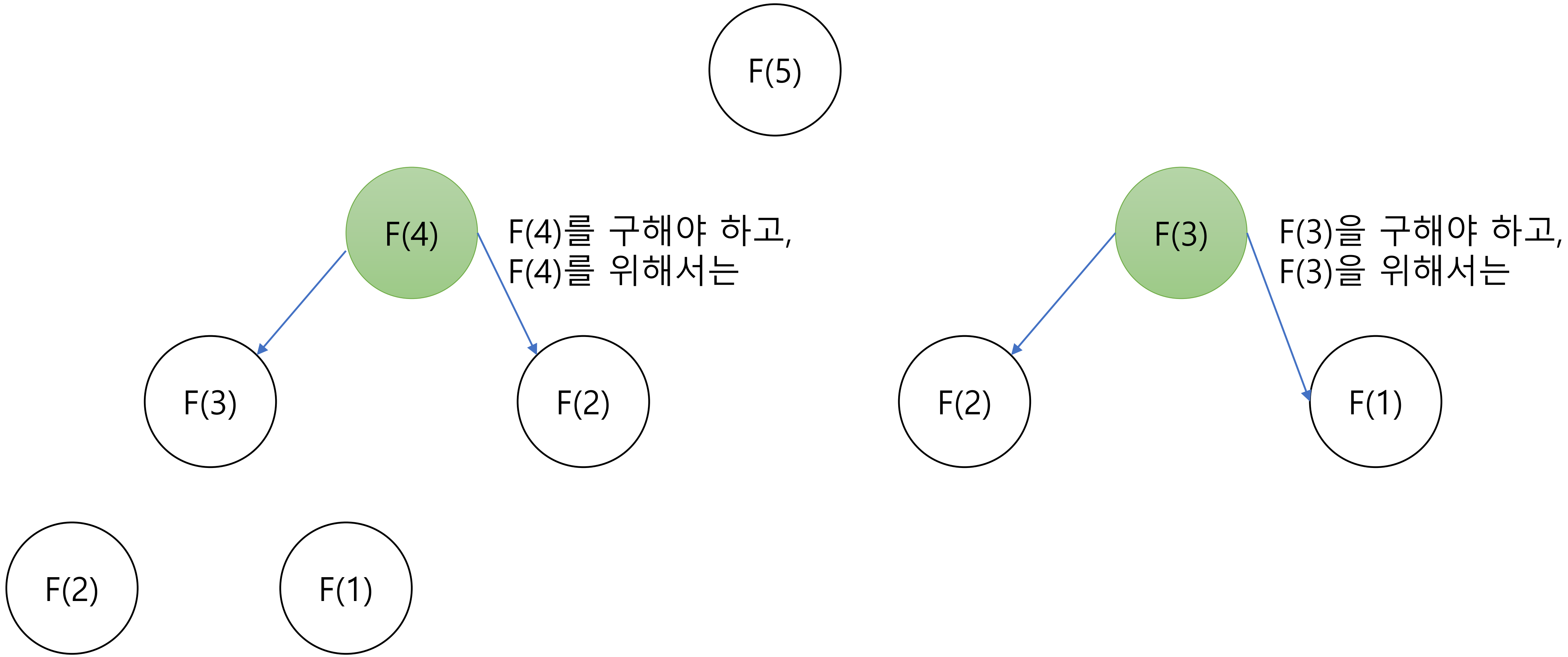
그런데 이렇게 코드를 짜면...

Fibonacci Sequence-**Without DP**



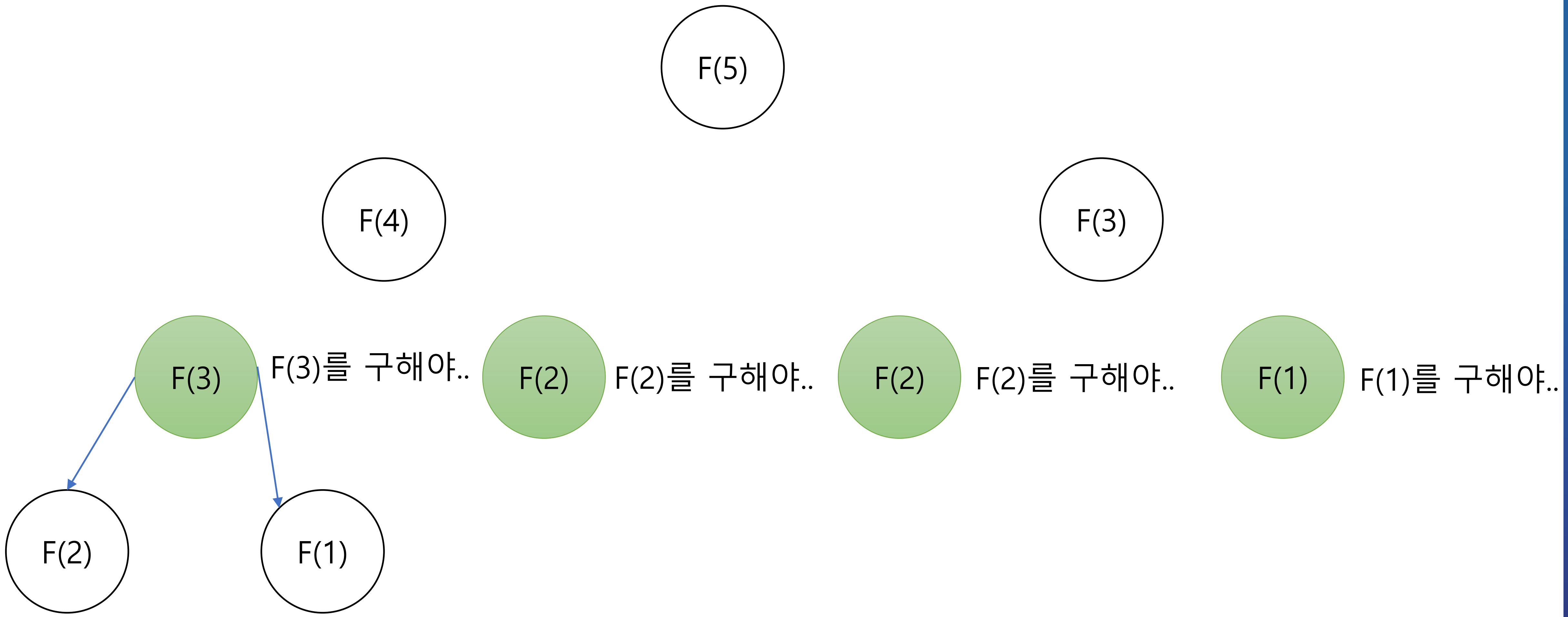

```
return Fibo(n - 1) + Fibo(n - 2);//현재 항을 구하려면 전 항과 그 전 항을 더해야 한다!
```

Fibonacci Sequence-**Without DP**



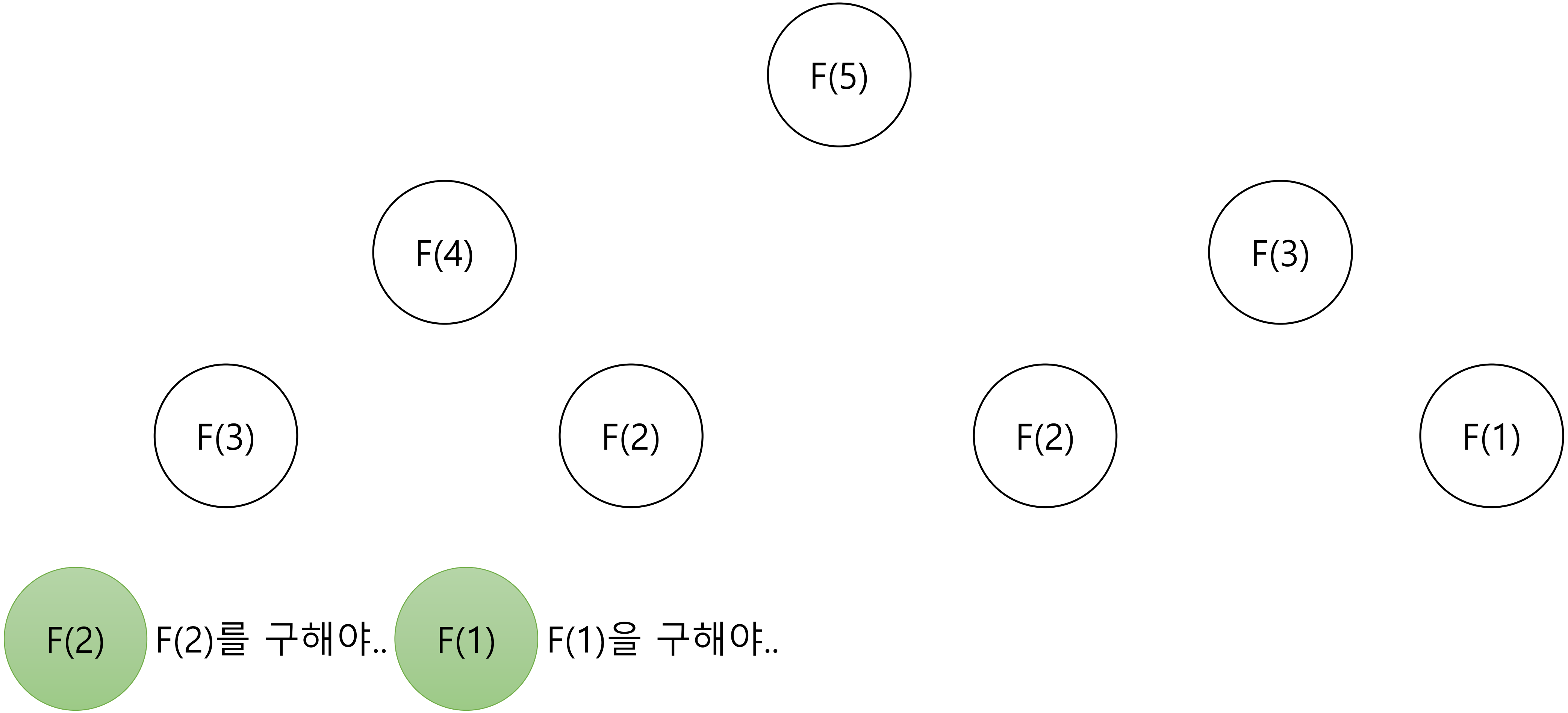

```
return Fibo(n - 1) + Fibo(n - 2);//현재 항을 구하려면 전 항과 그 전 항을 더해야 한다!
```

Fibonacci Sequence-**Without DP**



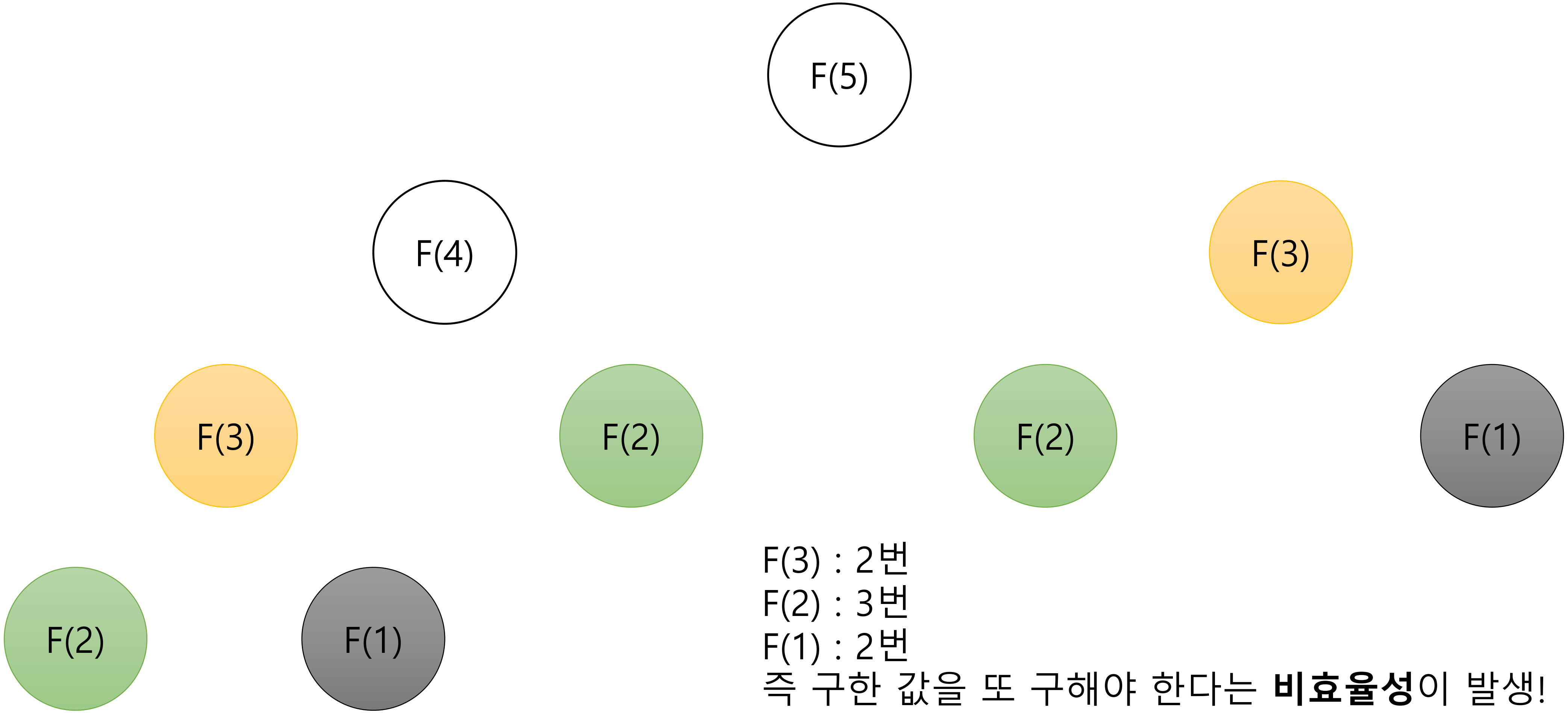

```
return Fibo(n - 1) + Fibo(n - 2);//현재 항을 구하려면 전 항과 그 전 항을 더해야 한다!
```

Fibonacci Sequence-**Without DP**




```
return Fibo(n - 1) + Fibo(n - 2);//현재 항을 구하려면 전 항과 그 전 항을 더해야 한다!
```

Fibonacci Sequence-**Without DP**



F(3) : 2번
F(2) : 3번
F(1) : 2번
즉 구한 값을 또 구해야 한다는 **비효율성**이 발생!

```
return Fibo(n - 1) + Fibo(n - 2);//현재 항을 구하려면 전 항과 그 전 항을 더해야 한다!
```

Fibonacci Sequence-**Top Down**

- Top down 방식을 이해하기 전에…
- Memoization이란??(Memorization X)
  - 어떤 단계에서 이미 계산해 나온 값을 배열에 저장해 놓는 기법입니다!
  - 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!)

Memoization을 이용해서 Top down 방식으로 피보나치수열을 구현한다면…?

Fibonacci Sequence-**Top Down**

```
#include <stdio.h>

int arr[100]; 값 저장해줄 배열
int Fibo(int n)
{




    arr[n] = Fibo(n - 1) + Fibo(n - 2); 점화식&메모이제이션
    return arr[n];
}
```

Fibonacci Sequence-**Top Down**

```c
#include <stdio.h>

int arr[100];
int Fibo(int n)
{


    if (arr[n] > 0) return arr[n];

    arr[n] = Fibo(n - 1) + Fibo(n - 2);
    return arr[n];
}
```

계산해놨다면? 바로 계산 값 리턴!

Fibonacci Sequence-**Top Down**

```
int arr[100];
int Fibo(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;

    if (arr[n] > 0) return arr[n];

    arr[n] = Fibo(n - 1) + Fibo(n - 2);
    return arr[n];
}
```

기저조건!(문제에 나오는 초기값)

Fibonacci Sequence-**Top Down**

DP없이

```
int Fibo(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;

    return Fibo(n - 1) + Fibo(n - 2);
}
```

DP
(Top Down)

```
int arr[100]; 값 저장해줄 배열
int Fibo(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;      기저조건

    if (arr[n] > 0) return arr[n]; 계산해놨다면? 바로 계산 값 리턴!

    arr[n] = Fibo(n - 1) + Fibo(n - 2);  점화식 & 메모이제이션
    return arr[n];
}
```

Fibonacci Sequence-**Bottom Up**

```c
#include <stdio.h>

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

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

    for (int i = 2; i <= n; i++) {
        fibo[i] = fibo[i - 1] + fibo[i - 2];//배열에 담긴 값을 더해주자!
    }
    printf("%d", fibo[n]);
}
```

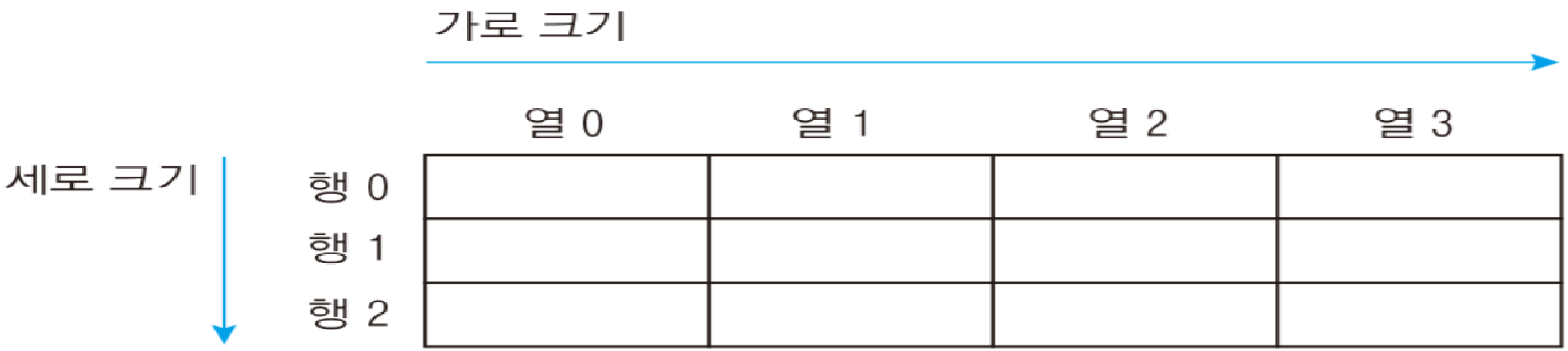
Bottom Up 방식은 4주차 때 배웠으므로 설명은 생략,,,

챕터 2 : Combination

조합을 구해보자!

2차원 배열

- 조합을 배우기 앞서서...2차원 배열을 배우고 들어갈게요!
- 2차원 배열이란?
  - 배열 안에 배열이 들어간 형태
  - 2차원 좌표평면이라고 편하게 생각하면 됩니다!
  - 좌표평면과 다른점은! (0,0) 즉 [0][0]이 좌측상단이라는 점!



2차원 배열

행 X

열 Y

	열 0	열 1	열 2	열 3
행 0	[0][0]	[0][1]	[0][2]	[0][3]
행 1	[1][0]	[1][1]	[1][2]	[1][3]
행 2	[2][0]	[2][1]	[2][2]	[2][3]

이런 식으로 이차원 배열을 생각해주시면 될 것 같아요!
이차원 배열을 오늘 몇 번 사용하게 될 텐데.. 익숙지 않다면?
멘토를 괴롭혀주세요

조합 구하기!

- 조합은 $C(n, r) = n!/(r! * (n - r)!)$ 이렇게 구할 수 있습니다!
- 팩토리얼은 재귀함수를 통해 구하지만,,,n이 매우 크다면 Stack-overflow가 날 것 같은데,,,?

- 다른 형태로 $C(n, r)$을 구하자!

조합 구하기

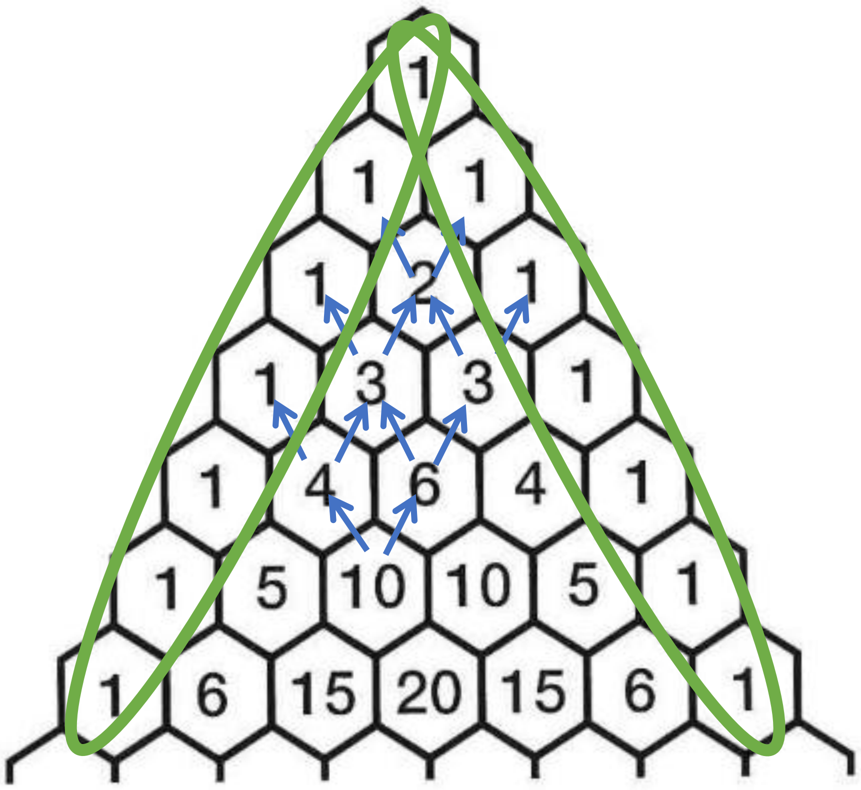
- $C(n,r) = n!/(r! * (n-r)!)$도 있지만
- $C(n,r) = C(n-1,r) + C(n-1,r-1)$ 이 식도 있다!
- 재귀함수를 통해서 쉽게 구할 수 있는 느낌이 드는데..?

- *힌트* C(n,r) 조합을 구하는 함수라 두고, 함수 안에서 메모이제이션과 기저조건을 잘 만들어준다면,,,?

조합 구하기

- Boj 11050 이항 계수 1

조합 구하기



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

$$C(5,2) = C(4,2) + C(4,1)$$
$$C(4,2) = C(3,2) + C(3,1)$$
$$C(4,1) = C(3,1) + C(3,0)$$
$$C(3,2) = C(2,2) + C(3,1)$$

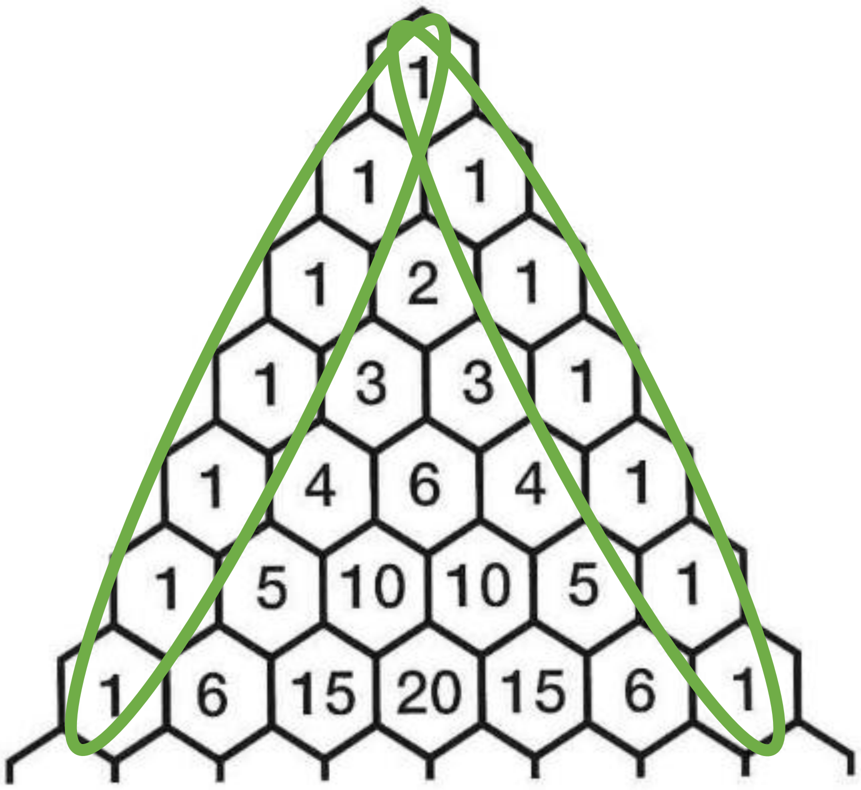
->재귀함수 느낌!

```
int N, K, ncr[12][12];

int Combi(int n, int k)
{
    if (ncr[n][k] > 0) return ncr[n][k];
    return ncr[n][k] = Combi(n - 1, k) + Combi(n - 1, k - 1);
}
```

메모이제이션!

조합 구하기



main문에서 파스칼의 삼각형의
양 변을 1로 채워줍시다!
(배열 이름 ncr[][] 라고 했어요!)

```
int main()
{
    scanf("%d%d", &N, &K);

    for (int i = 0; i <= N; i++) {
        for (int j = 0; j <= i; j++) {
            if (i == j || j == 0) ncr[i][j] = 1;
        }
    }

    printf("%d", Combi(N,K));
}
```

연습 문제

- Boj 2193 이친수

Boj 2193 이친수

- 문제 요약
  - 0과 1로 이루어진 수를 이친수라 부름
  - 0으로 시작하지 않는다!
  - 1이 연속으로 나타나지 않는다!(11을 부분 문자열로 갖지 않는다)
- 예시
  - 1자리 이친수 : 1
  - 2자리 이친수 : 10
  - 3자리 이친수 : 101, 100
  - 4자리 이친수 : 1000, 1001, 1010
  - 5자리 이친수 : 10000, 10001, 10010, 10101, 10100
- N자리 이친수의 개수는 …? 규칙을 찾아봅시다!

Boj 2193 이친수

- 먼저 앞자리가 무조건 10으로 시작하는 걸 발견!(n>=2)
- 예시
  - 1자리 이친수 : 1
  - 2자리 이친수 : 10
  - 3자리 이친수 : 101, 100
  - 4자리 이친수 : 1000, 1001, 1010
  - 5자리 이친수 : 10000, 10001, 10010, 10101, 10100
- 4자리 이친수 중에서
  - 1000은 3자리 이친수 100 + 0
  - 1001은 3자리 이친수 100 + 1
  - 1010은 3자리 이친수 101 + 0

Boj 2193 이친수

- 예시
  - 1자리 이친수 : 1
  - 2자리 이친수 : 10
  - 3자리 이친수 : 101, 100
  - 4자리 이친수 : 1000, 1001, 1010
  - 5자리 이친수 : 10000, 10001, 10010, 10101, 10100
- 5자리 이친수 중에서
  - 10000은 4자리 이친수 1000 + 0
  - 10001은 4자리 이친수 1000 + 1
  - 10010은 4자리 이친수 1001 + 0
  - 10101은 4자리 이친수 1010 + 1
  - 10100은 4자리 이친수 1010 + 0

Boj 2193 이친수

- N자리 이친수 중 0으로 끝나는 것 :
  - n-1자리 이친수 중에서 0으로 끝나는 것
  - +
  - n-1자리 이친수 중 1로 끝나는 것
- N자리 이친수 중 1로 끝나는 것 :
  - N-1자리 이친수 중 0으로 끝나는 것 Only!

- 점화식 발견!
  - Pinary_num[n][0]=pinary_num[n-1][0]+pinary_Num[n-1][1]
  - Pinary_num[n][1]=pinary_num[n-1][0]

Boj 2193 이친수

- 하지만...int형 배열로 만들게 되면 틀렸습니다가 나옵니다!
그 이유는

Pinary_num[n][0]을 F(n)이라 하고, pinary_num[n][1]을 G(n)이라 하면

F(n)=F(n-1)+G(n-1) ,G(n)=F(n-1)입니다.

우리가 구해야 하는 답은 F(n)+G(n)이고

F(n)+G(n)=F(n-1)+G(n-1)+F(n-1)

=F(n-1)+G(n-1)+F(n-2)+G(n-2)입니다.

F(n)+G(n)을 H(n)이라 한다면

H(n)=H(n-1)+H(n-2)꼴 이므로... 어디선가 많이 본..?

결론 : 피보나치 수열 형태입니다! 그런데 피보나치 수열은 n이 46일 때 28억이 되어서 int형 범위를 초과하게 되고 이 문제는 n이 90까지 이기 때문에 long long형 배열을 써야 정답이 됩니다!

Boj 2193 이친수

```c
#include <stdio.h>

int n;
long long pinary_num[100][2];
int main()
{
    scanf("%d", &n);

    pinary_num[1][0] = 0;
    pinary_num[1][1] = 1;
    pinary_num[2][0] = 1;
    pinary_num[2][1] = 0;

    for (int i = 3; i <= n; i++) {
        pinary_num[i][0] = pinary_num[i - 1][0] + pinary_num[i - 1][1];
        pinary_num[i][1] = pinary_num[i - 1][0];
    }
    printf("%lld", pinary_num[n][0] + pinary_num[n][1]);
}
```

Int형 배열 쓰면 틀려요!

연습 문제

- Boj 11051 이항 계수 2
- Boj 9095 1,2,3 더하기
- Boj 11726 2xn 타일링
- Boj 1003 피보나치 함수
- Boj 9461 파도반 수열
- Boj 1149 RGB거리(도전!)

수고하셨습니다~