

# Dynamic programming

(동적 프로그래밍)

-오유진-

<https://youtu.be/VBuO2J3USXw>

# Contents

Dynamic Programming 정의

Dynamic Programming vs Divide and Conquer

피보나치 수열 - Divide and Conquer

피보나치 수열 - Dynamic Programming

타일링 - Dynamic Programming



# Dynamic Programming(DP)

- 큰 문제를 작은 문제들로 쪼개어 푸는 방식.(주로 Bottom-up 방식)
- 하나의 문제는 단 한번만 푼다.

작은 문제들이 중복될 때 한번 계산한 결과는 저장해두고 다시 사용한다. (즉, 메모이제이션 기법을 이용한다.)

- 메모이제이션

- 동일한 계산을 반복해야 할 경우 한 번 계산한 결과를 메모리에 저장해 두었다가 꺼내 씬으로써 중복 계산을 방지할 수 있게 하는 기법이다.

**!다이나믹 프로그래밍의 핵심 기법!**

- 조건

- 작은문제가 반복해서 발생하는 경우
- 같은 문제는 구할 때마다 값이 동일

- 규칙을 찾아 점화식을 세워야 한다. (Ex.  $DP[i] = DP[i-1] + DP[i-2]$ )

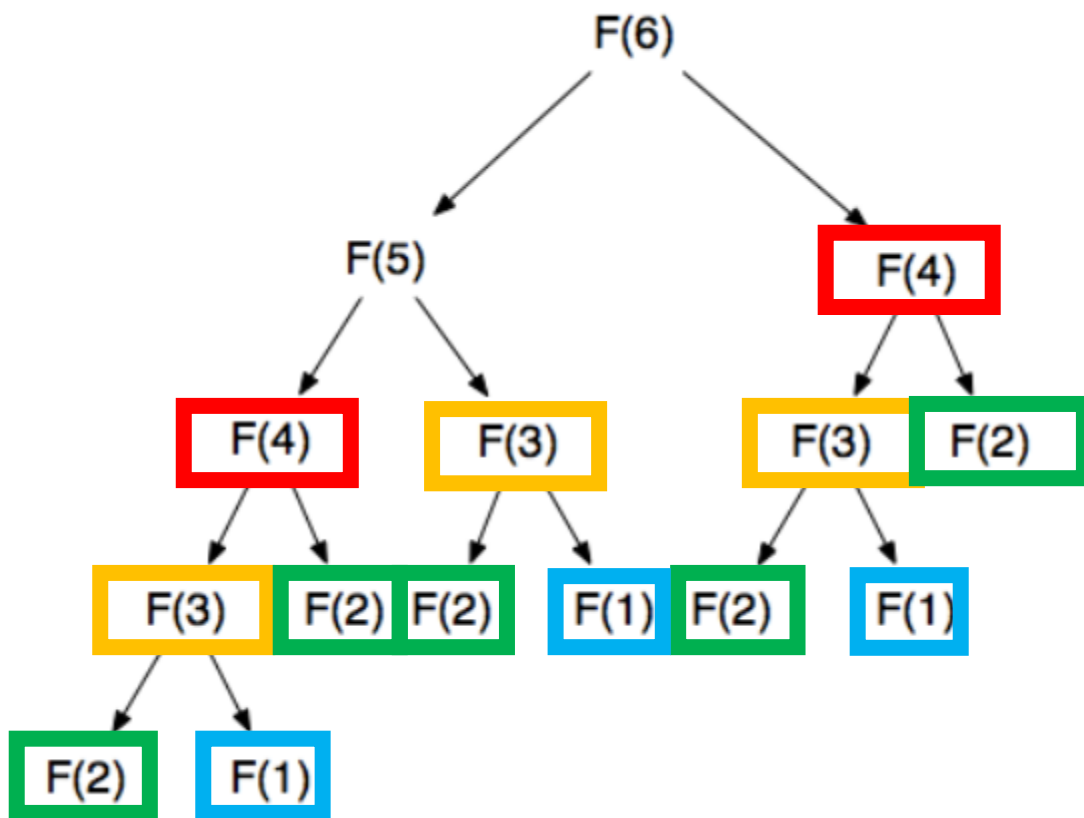
# Dynamic Programming vs Divide and Conquer

Dynamic Programming 다이나믹 프로그래밍	Divide and Conquer 분할정복
주로 Bottom-up	Top-down
Memoization 기법 사용	부분 문제 중복 x, memoization 필요 x
Ex) fibonacci	Ex) quick sort, merge sort

## Memoization?

한번 계산한 결과는 저장  
해두고 다시 사용한다.(결  
과 재활용)

# 피보나치 수열 - Divide and Conquer (메모이제이션 x)



```
#include <stdio.h>
int fibo(int x){
    if(x==1) return 1;
    if(x==2) return 1;
    return fibo(x-1) + fibo(x-2);
}
```

```
int main(void){
    printf("%d",fibo(6));
}
```

```
int main(void){
    printf("%d",fibo(50));
}
```

# 피보나치 수열(DP로 해결-Top down)

## Top-down 방식(재귀)

1. 큰 문제를 먼저 방문 후 작은 문제로 나눈다.(호출한다.)
2. 작은 문제를 푼다.
3. 작은 문제를 풀었으니, 이제 큰 문제를 푼다.

Ex)

fibonacci(n)을 풀어야한다.

fibonacci(n-1)과 fibonacci(n-2)로 문제를 나눈다.

.

.

fibonacci(n-1)과 fibonacci(n-2)의 값을 더해 fibonacci(n)을 구한다.

```
#include <stdio.h>
```

```
int d[100];
```

```
int fibo(int x){
```

```
    if(x==1) return 1;
```

```
    if(x==2) return 1;
```

```
    if(d[x]!=0) return d[x];
```

```
    return d[x] = fibo(x-1) + fibo(x-2);
```

```
}
```

```
int main(void){
```

```
    printf("%d", fibo(6));
```

```
}
```

# 피보나치 수열(DP로 해결 – Bottom-up)

## Bottom-up 방식(반복문)

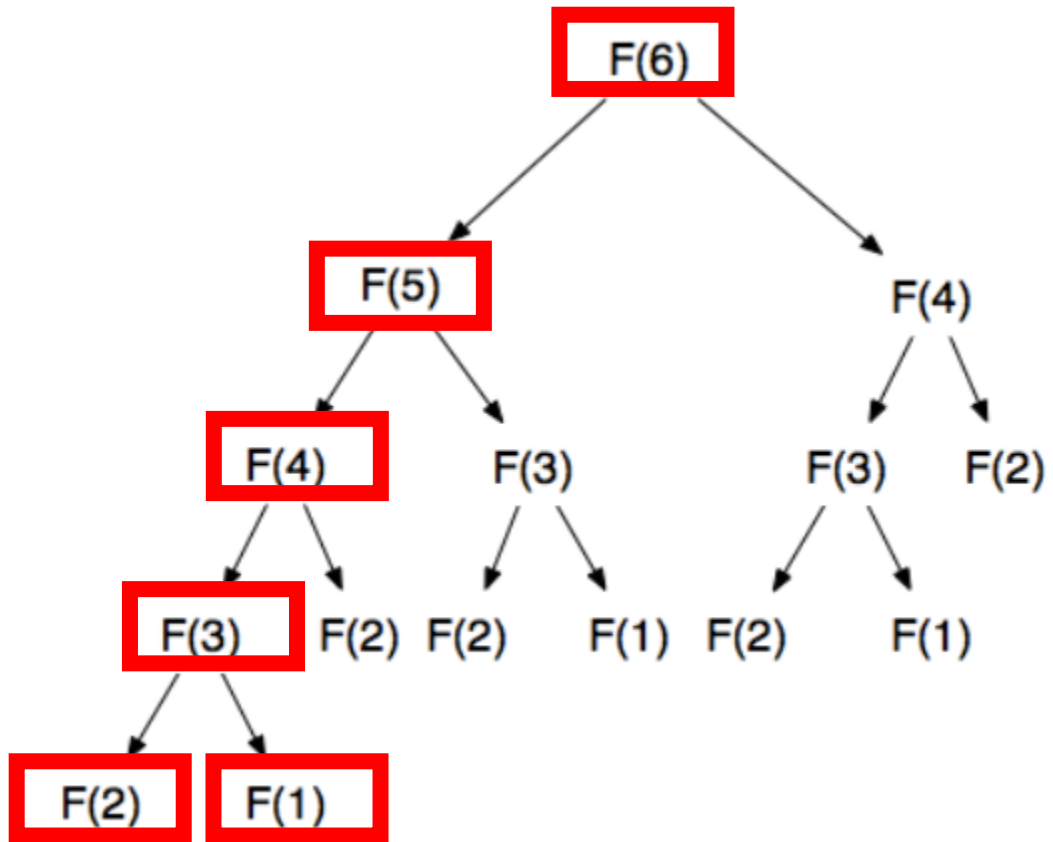
1. 문제를 크기가 작은 문제부터 차례대로 푼다.
2. 문제의 크기를 조금씩 크게 만들면서 문제를 푼다.

반복하다 보면 가장 큰 문제를 풀 수 있다.

```
#include <stdio.h>
int d[100];
int fibo(int x){
    d[0]=0;
    d[1]=1;
    for(int i=2;i<=x;i++){
        d[i]=d[i-1]+d[i-2];
    }
    return d[x];
}

int main(void){
    printf("%d",fibo(6));
}
```

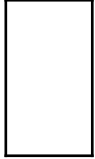
# 피보나치 수열(DP)



- 시간복잡도  $O(N)$
- 모든 노드들 방문 x, 한번씩만 방문.



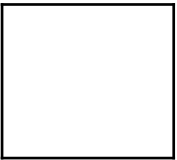
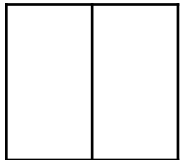

# 타일링

• 2x1 


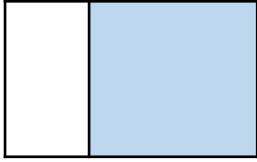
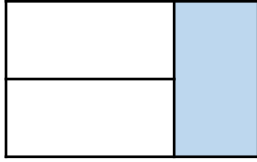
$dp[1]=1$

※ 점화식



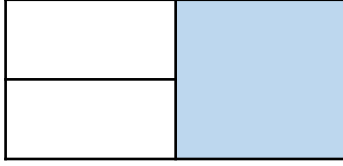
$dp[n] = dp[n-1] + dp[n-2]$

• 2x2  =  + 

$dp[2]=2$

• 2x3  =  + 

$dp[3] = dp[2] + dp[1]$

• 2x4  =  + 

$dp[4] = dp[3] + dp[2]$

# 다이나믹 프로그래밍 정리

- 중복되는 문제들이 발생할 때 유용.
- 메모이제이션 활용.
- Ex) 피보나치수열, 타일링 문제 해결

Q & A

