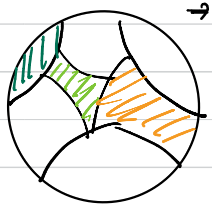
동적 계획법 (Dynamic Programming)

$Sum(n) = 1 + 2 + 3 + \cdots n$
$= \underbrace{(1 + 2 + 3 + \cdots n{-}1)}_{Sum(n-1)} + n$

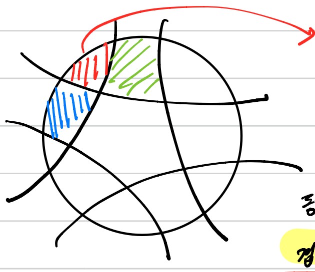


→ 문제를 나누어 풀면서 부분문제의 값을 모아 전체 해를 구하는 것. (재귀적으로)

$Sum(n) = Sum(n-1) + n \quad (sum(1) = 1)$ ⇐ 재귀호출 (Divide & conquer)

$Sum[n] = S[n-1] + n \; , \; S[1] = 1$ ⇐ 값을 리스트에 저장 재사용 (Dynamic Programming)

$S[n-1]$: 값을 미리저장 → 재 사용



중복 (겹치는 부분)

Divide & conquer는 두부 쪼개 듯이 딱딱 잘라지지만 동적 계획은 작은 부분이 **서로 겹칠 수 있다.**

⇓

테이블에 저장 (1차원 리스트, 2차원 리스트)

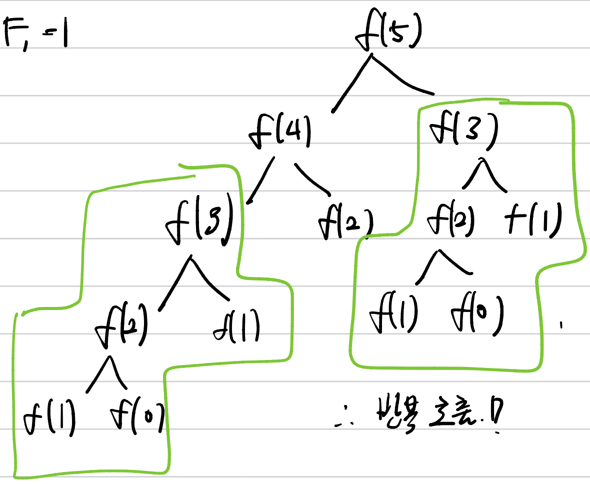
⇒ 중복을 피할 수 있다.

동적 계획법 예시) Fibonacci

피보나치 수열: $F_n = F_{n-1} + F_{n-2} \; , \; F_0 = 1 \quad F_1 = 1$

f(n) : Divide and Conquer

```
if n==0 or n==1:
    return n
return f(n-1) + f(n-2)
```



∴ 반복 호출.!

f(n) : Dynamic Programming

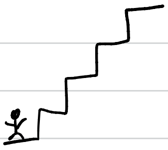
$F = [0, 1]$ # $F[0] = F_0 = 0 , \; F[1] = 1$

O(n)
```
for i in range[2, n+1]:
    F[i] = F[i-1] + F[i-2]    O(1)  # 상수 연산
return F
```

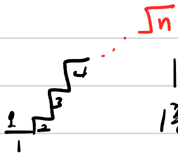
$F_0, \cdots F_n$ 모두계산 ⇒ 리스트에 값들을 미리 저장 해 놓아야 함.

예제) 계단 오르기

- 1칸씩
- 2칸씩



1층 → n층 : 경우의 수



1층 → 2층 : 1개

1층 → 3층 : 2개

※ n층 까지 올라 갔다는 것

n층 까지 오는 경우의 수

= n-1 층까지 오는 경우의 수 + n-2 층까지 오는 경우의 수

A[n] = 1 → n층까지 오르는 경우의 수

A[1] = 1, A[2] = 1, A[3] = 2 (바닥 조건)

A[n] = (1 → n-1)가 이르는 경우의 수 + (1 → n-2) 경우의 수

(A[n-1]) (A[n-2])

A[n] = A[n-1] + A[n-2] (점화식)

= 피보나치 수열

```
A = [None, 1, 1]      # A[1], A[2]
for i in range(3, n+1):
    A[i] = A[i-1] + A[i-2]
```

→ 이전 문제들을 계산하기 위해서

* 작은 문제의 계산이 선행되어야 한다.