

SCV Junior

DYNAMIC PROGRAMMING

Contents



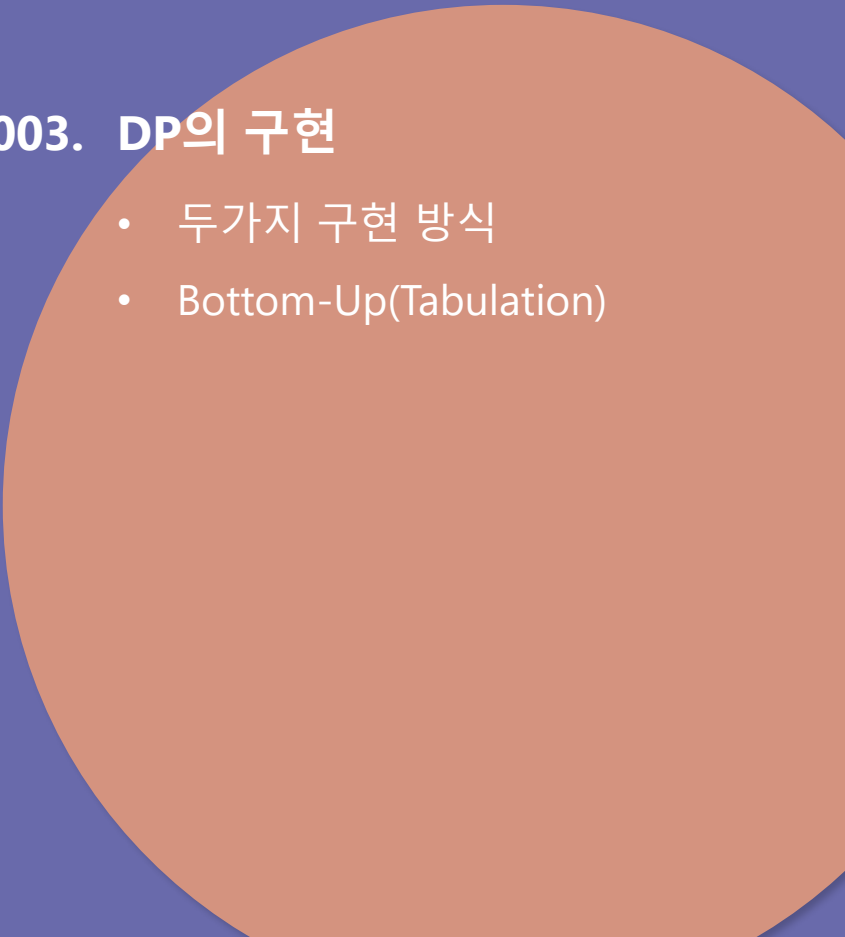
001. 기본 개념

- 탐욕법이 실패하는 경우
- 최적해 구하기

002. DP와 재귀

- 큰 문제와 작은 문제
- 피보나치 수열의 재귀적 구현

003. DP의 구현

- 두가지 구현 방식
 - Bottom-Up(Tabulation)
- 

001. 기본개념

| 탐욕법이 실패하는 경우

동전 교환 문제 최적해를 보장하는 경우

1원, 5원, 10원, 50원짜리 동전이 있을 때

328원을 만들 수 있는 최소 동전 개수는?

쓸 수 있는 동전 개수는 제한이 없다

328을 50으로 나누기: **6** ... 28

28을 10으로 나누기: **2** ... 8

8을 5로 나누기: **1** ... 3

3을 1로 나누기: **3**

..... **$6+2+1+3 = 12$**

001. 기본개념

| 탐욕법이 실패하는 경우

동전 교환 문제
최적해를 보장하지 않는 경우

1원, 7원, 12원, 50원짜리 동전이 있을 때

328원을 만들 수 있는 최소 동전 개수는?

쓸 수 있는 동전 개수는 제한이 없다

328을 50으로 나누기: **6** ... 28

28을 12으로 나누기: **2** ... 4

4을 7로 나누기: **0** ... 4

4을 1로 나누기: **4**

..... **$6+2+0+4 = 12$**

001. 기본개념

| 탐욕법이 실패하는 경우

동전 교환 문제
최적해를 보장하지 않는 경우

1원, 7원, 12원, 50원짜리 동전이 있을 때

328원을 만들 수 있는 최소 동전 개수는?

쓸 수 있는 동전 개수는 제한이 없다

328을 50으로 나누기: 6 ... 28

28을 12으로 나누기: 2 ... 4

4을 7로 나누기: 0 ... 4

4을 1로 나누기: 4

~~..... 6 + 2 + 0 + 4 = 12~~

50원짜리 6개(300)
7원짜리 4개(28)

... 10개!

001. 기본개념

| 최적해 구하기

탐욕법으로
최적해를 구할 수 없는 경우엔...

무차별 알고리즘

모든 경우를 시도해보는 알고리즘
최적해를 구할 수 있음
입력의 크기가 커지면 매우 느려짐

Dynamic Programming

무차별 알고리즘보다 효율적
최적해를 구할 수 있음

002. DP와 재귀

I 큰 문제와 작은 문제

DP문제를 풀기 위해서는
문제의 재귀적인 구조를 찾아야 한다

문제의 재귀적인 구조란

큰 문제의 답을 작은 문제의 답을 가지고 구하는 것을 의미한다

작은 문제를 원래 주어진 문제와 크기만 다를 뿐, 동일한 문제이다

DP는 작은 문제의 **답을 표에 저장**해 두었다가 이를 활용하는 알고리즘

002. DP와 재귀

| 피보나치 수열의 재귀적 구현

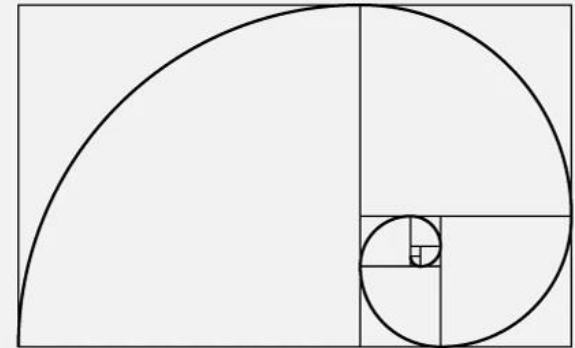
피보나치 수열

앞선 두 수의 합이 자신의 값이 되는 수열

$$a_n = a_{n-2} + a_{n-1}$$

$$\text{즉, } F(n) = F(n-2) + F(n-1) \quad * (n \geq 2)$$

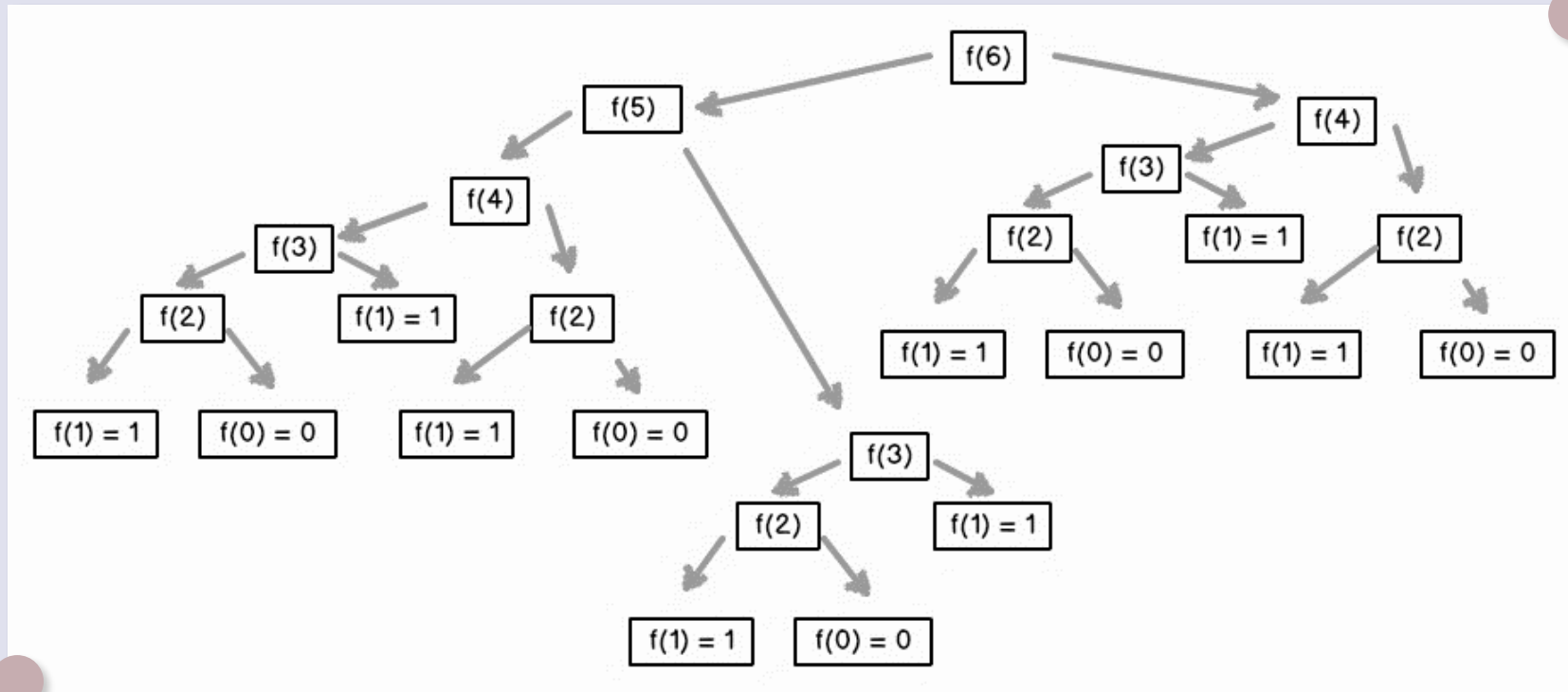
자신보다 작은 문제의 답을 이용하여 자신의 답을 구할 수 있음



0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

002. DP와 재귀

| 피보나치 수열의 재귀적 구현



재귀적으로 구현할 시 엄청난 중복 호출이 발생

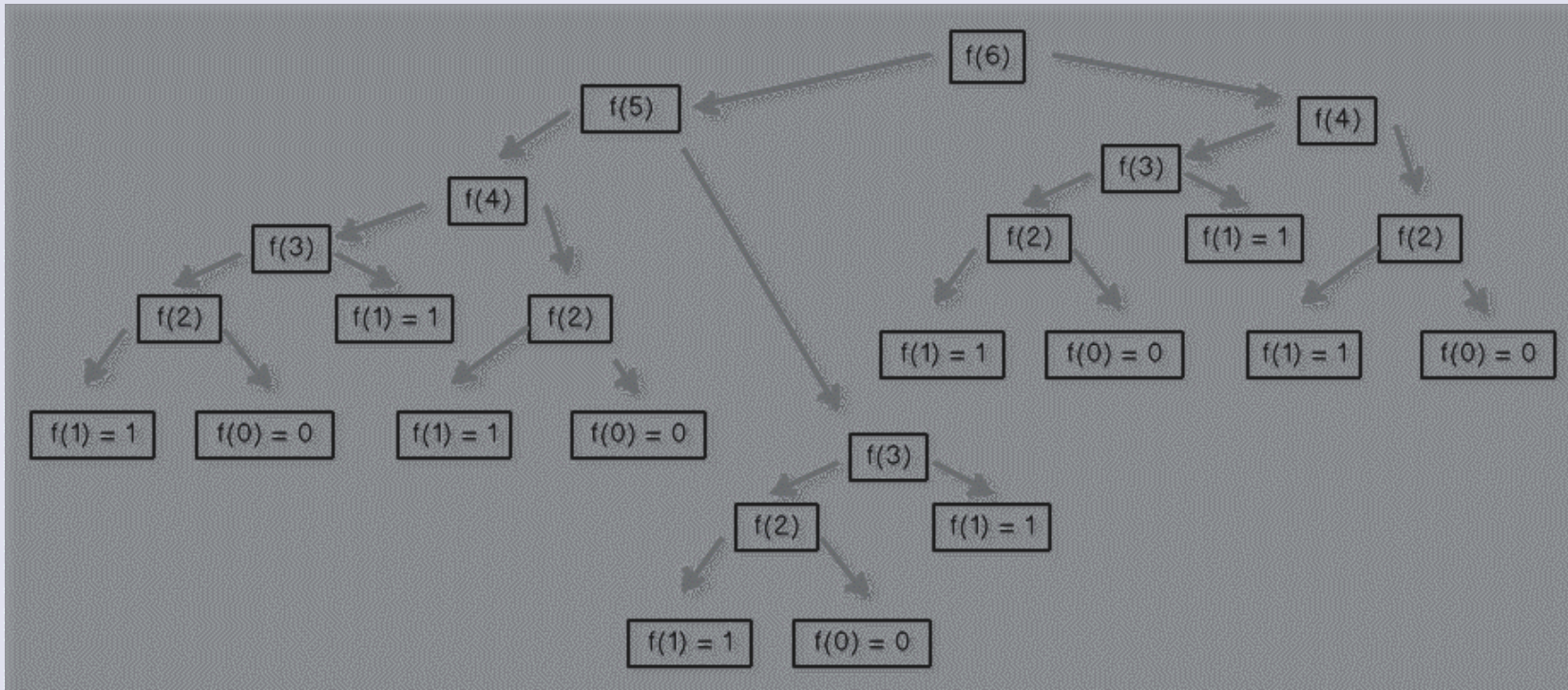
002. DP와 재귀

| 피보나치 수열의 재귀적 구현

이미 구한 값을 어딘가(표)에 저장해두고,
필요할 때 사용한다면 어떨까?

	0	1	2	3	4	5	6
arr	0	1					

arr[k]: 피보나치 k번째 수



002. DP와 재귀

| 피보나치 수열의 재귀적 구현

DP 알고리즘을 사용할 때는
베이스 값을 초기화 하여 무한 호출과 인덱싱 에러를 피한다

```
1 int fibonacci(int n)
2 {
3     if (n == 0) return 0;
4     if (n == 1) return 1;
5
6     if (dp[n] != -1) return dp[n];
7
8     dp[n] = fibonacci(n - 1) + fibonacci(n - 2);
9     return dp[n];
10 }
```

Colored by Color Scriptor CS

재귀함수의 베이스 조건.
베이스 조건이 없다면
재귀함수는 무한히 자신을 호출한다

003. DP의 구현

| 두가지 구현 방식

구현방식	Top-Down(Memoization)	Bottom-Up(Tabulation)
테이블 접근 순서	N부터 베이스조건까지	베이스조건부터 N까지
구현방법	재귀	반복문
설명	가장 큰 문제를 방문 후, 작은 문제를 호출하여 답을 찾는 방식	가장 작은 문제들부터 답을 구해가며 전체 문제의 답을 찾는 방식

003. DP의 구현

| Bottom-Up(tabulation)

```
1 int fibonacci(int n)
2 {
3     dp[0] = 0, dp[1] = 1;
4     for (int i = 2; i <= n; i++)
5         dp[i] = dp[i - 1] + dp[i - 2];
6 }
```

Colored by Color Scripter

→ 베이스 값 초기화.

0과 1을 채웠으므로,
반복문은 2부터 시작한다

00#. 문제 풀기

같이 풀기

BOJ 10826 피보나치 수 4

같이 풀기

BOJ 2579 계단 오르기

숙제

LeetCode 1025 Division Game

숙제

BOJ 2294 동전 2

+ 그래프 탐색 문제 2개
다익스트라 알고리즘 조사해오기