04. Dynamic Programming

Div. 3 알고리즘 스터디 / 임지환

Dynamic Programming?

= 동적 계획법

Dynamic???

Programming???

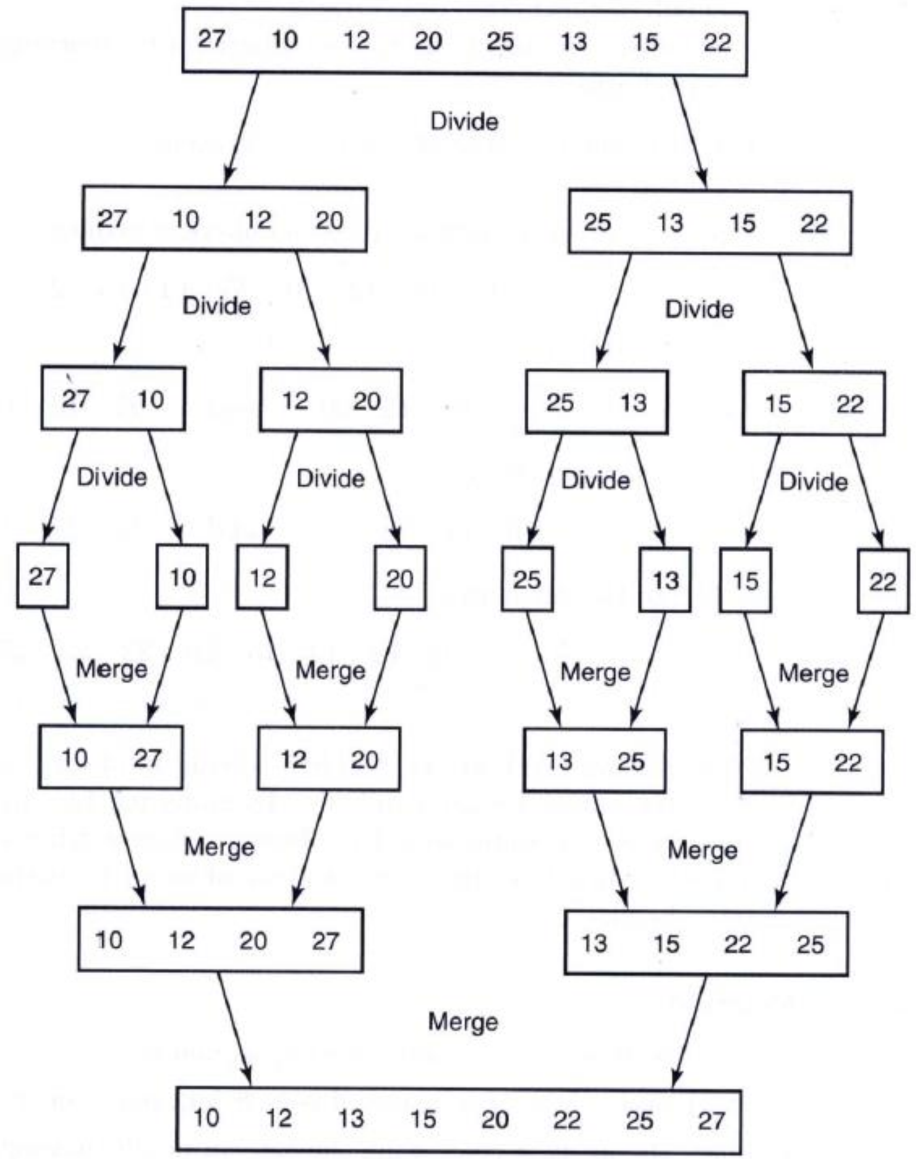
동적 계획법

- **최적화 문제(Optimization Problem)**를 연구하는 이론에서 유래
  - Computational Problem, Decision Problem, Optimization Problem….
- 주어진 문제를 더 작은 문제들로 나눈 뒤 이들로부터 원래 문제에 대한 해를 구하는 방법

Example : Merge Sort

합병 정렬을 하기 위해

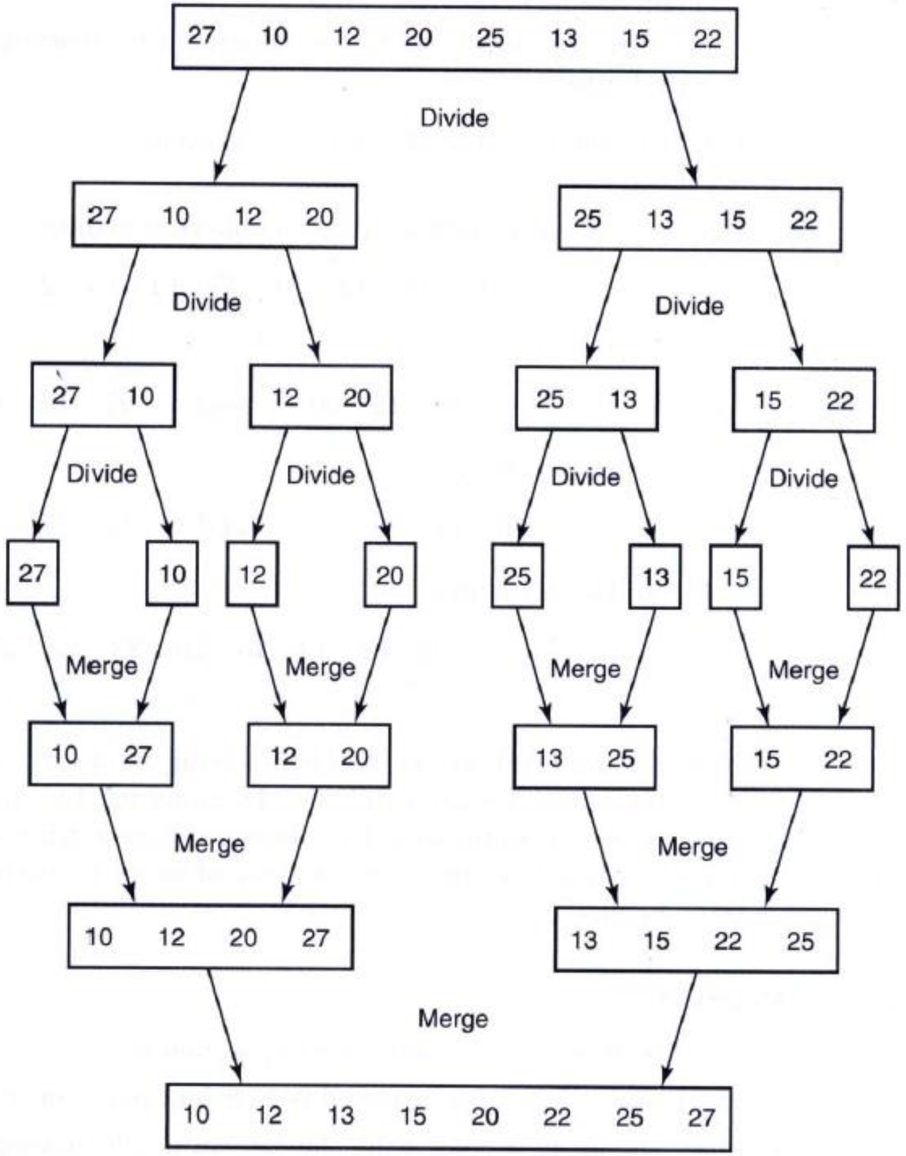
1) 작은 문제들로 나누었고
2) 이 결과들을 통해 전체 해를 구했다.



Example : Merge Sort

합병 정렬을 하기 위해

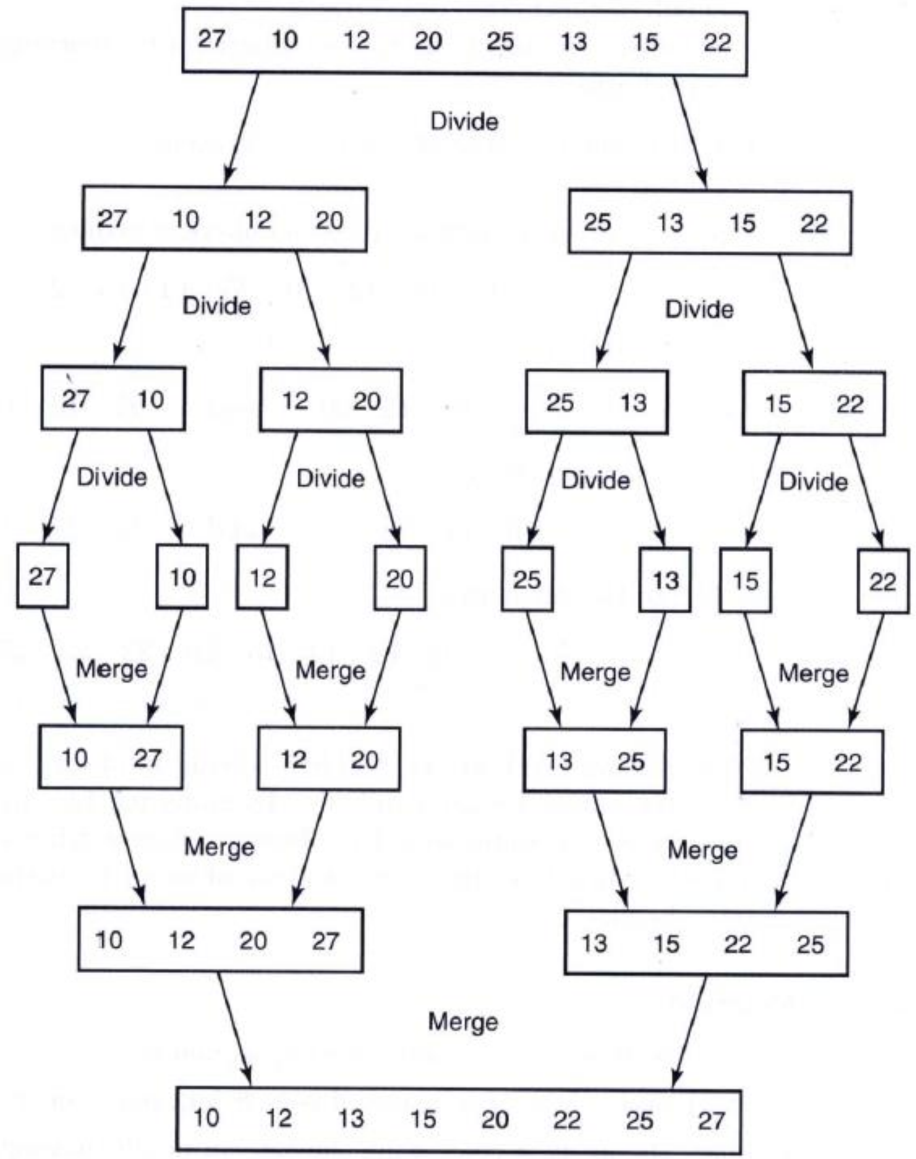
1) 작은 문제들로 나누었고
2) 이 결과들을 통해 전제 해를 구했다.
3) 이것도 동적 계획법인가?



Example : Merge Sort

합병 정렬을 하기 위해

1) 작은 문제들로 나누었고
2) 이 결과들을 통해 전제 해를 구했다.
3) 이것도 ~~[illegible]~~가?



Example 2 : Fibonacci Number

$$F_n = F_{n-1} + F_{n-2} \quad (n \geq 2)$$

Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

Example 2 : Fibonacci Number

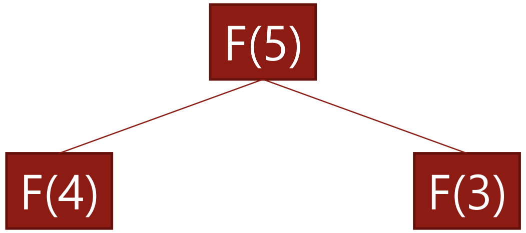
$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

F(5)

```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

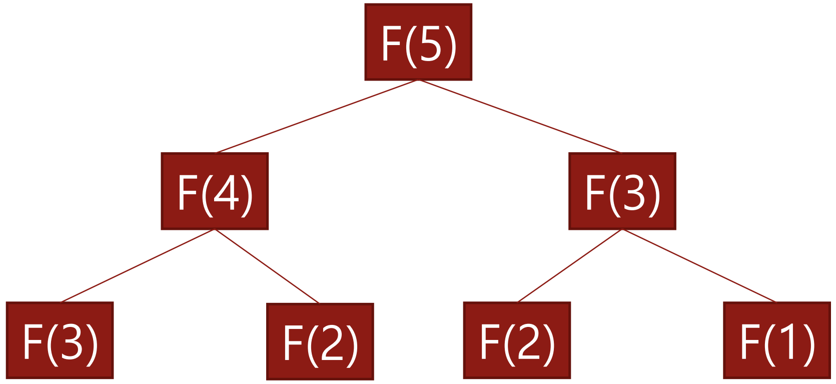



```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

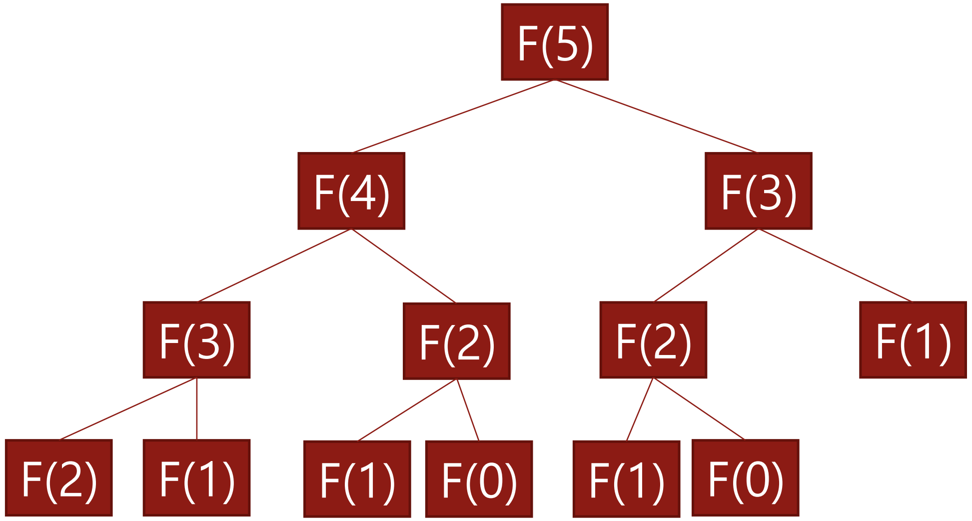
```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```



Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

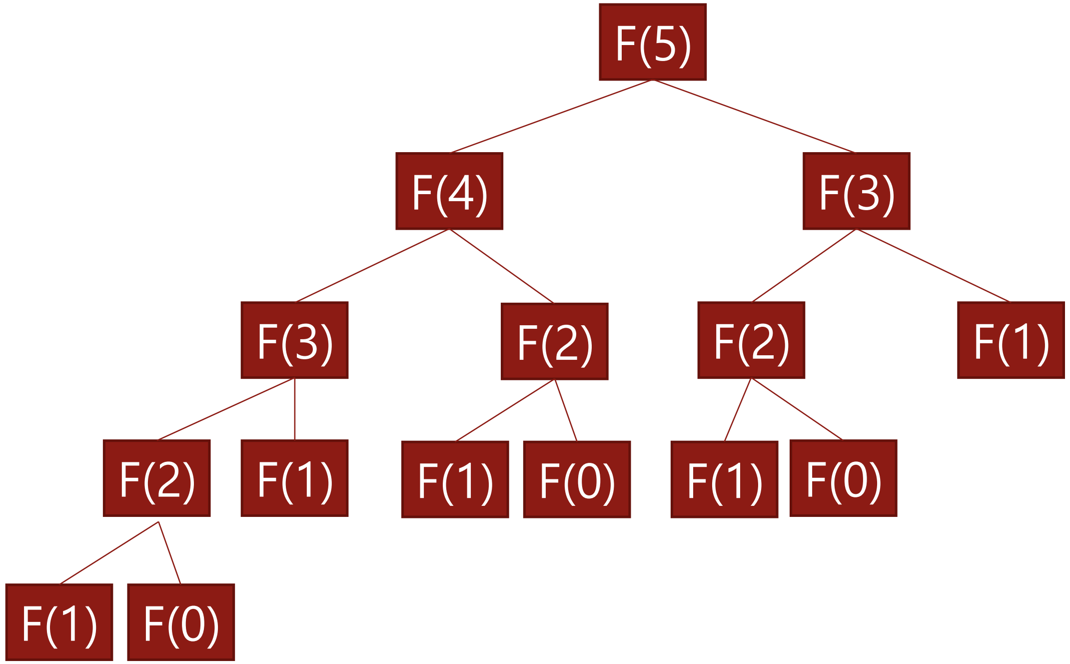
```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```



Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

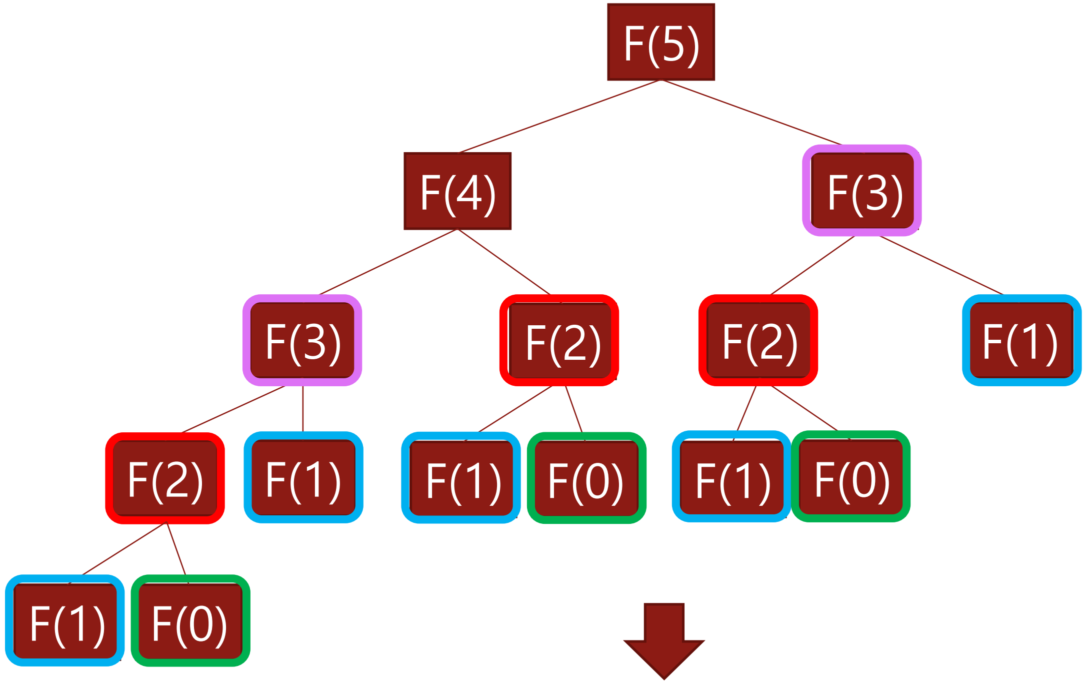
```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```



Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```



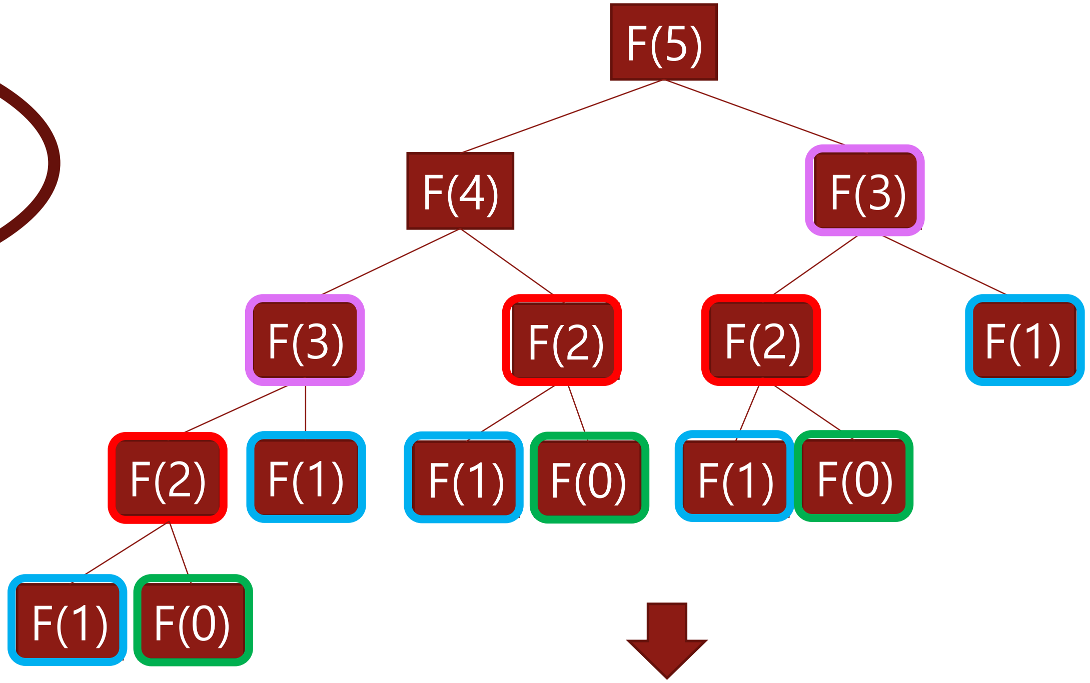
중복되는 부분 문제(Overlapping Subproblem)

Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

점화식(Recurrence)

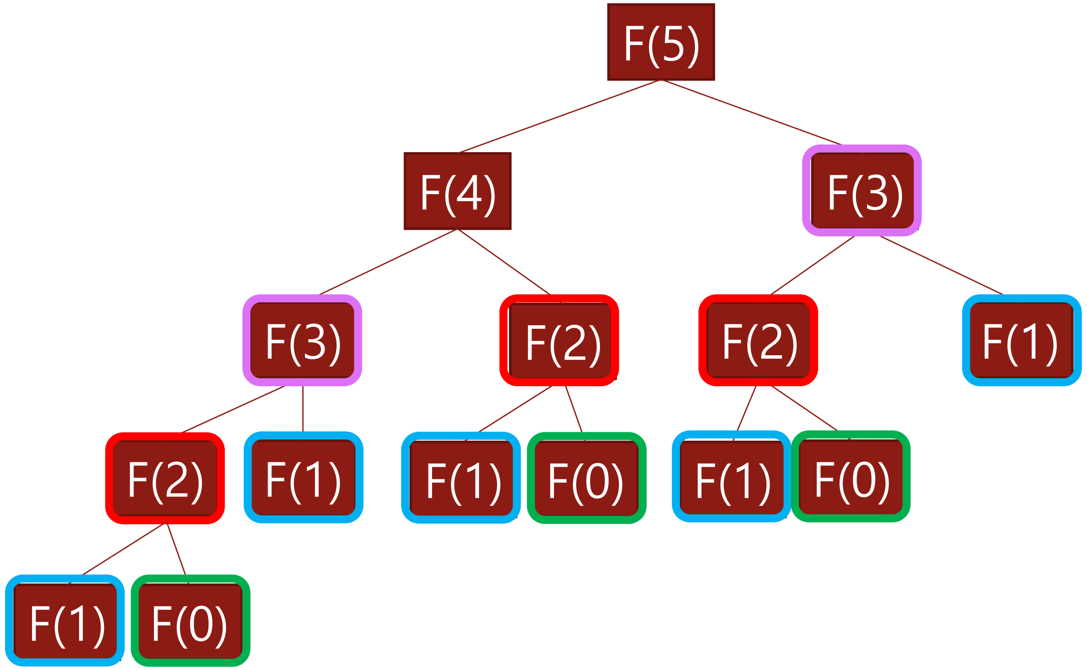


중복되는 부분 문제(Overlapping Subproblem)

Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

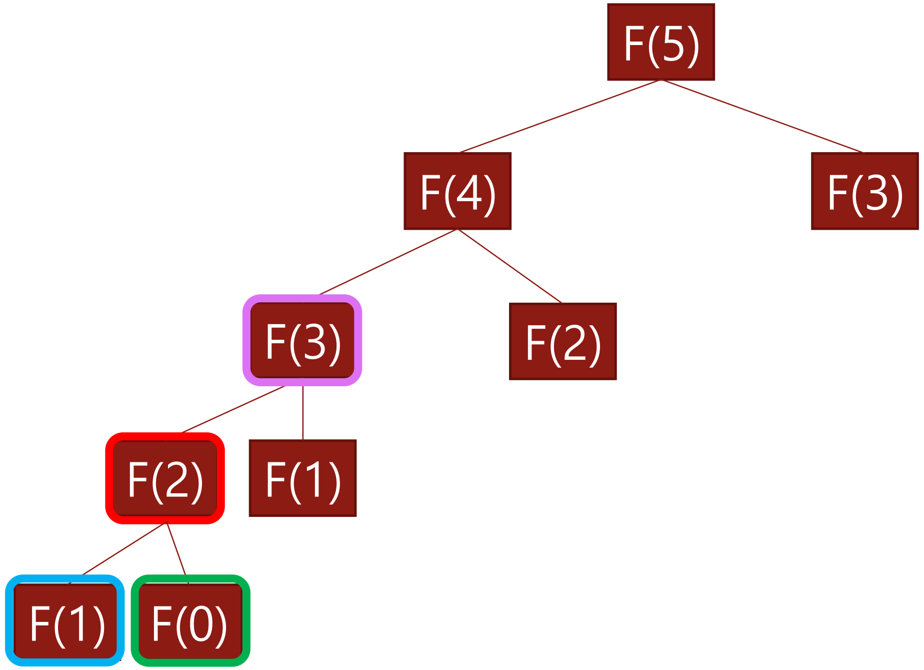
```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```



Example 2 : Fibonacci Number

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad (n \geq 2) \end{cases}$$

```
int fib (int n) {
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```



이미 계산된 부분 문제의 해

```
int fib (int n) {
    if (이미 계산된 값이라면) return 이미 계산된 그 값;
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

이미 계산된 부분 문제의 해

```
int fib (int n) {
    if (이미 계산된 값이라면) return 이미 계산된 그 값;
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

메모이제이션(Memoization)

```
int cache[MAXN];    //나올 수 없는 값으로 초기화
int fib (int n) {
    if (cache[n] != -1) return cache[n];
    if (n <= 1) return n;
    return fib(n-1) + fib(n-2);
}
```

Example 3 : Binomial Coefficient

- 이항계수

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

1 10 45 120 210 252 210 120 45 10 1

Example 3 : Binomial Coefficient

- 이항계수

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28   8   1
  1   9  36  84 126 126  84  36   9   1
1  10  45 120 210 252 210 120  45  10   1
```

Example 3 : Binomial Coefficient

$$
\begin{cases} 1 & (r = 0 \quad or \quad n = r) \\ \binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r} & \end{cases}
$$

Example 3 : Binomial Coefficient

$$\begin{cases} 1 & (r = 0 \ \ or \ \ n = r) \\ \binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r} & \end{cases}$$

```
int bino (int n, int r) {
    if (r == 0 || n == r) return 1;
    return bino(n-1, r-1) + bino(n-1, r);
}
```

Example 3 : Binomial Coefficient

$$\begin{cases} 1 & (r = 0 \quad or \quad n = r) \\ \binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r} & \end{cases}$$

```
int bino (int n, int r) {
    if (r == 0 || n == r) return 1;
    return bino(n-1, r-1) + bino(n-1, r);
}
```

```
int cache[MAXN][MAXR];
int bino (int n, int r) {
    if (r == 0 || n == r) return 1;
    if (cache[n][r] != -1) return cache[n][r];
    return bino(n-1, r-1) + bino(n-1, r);
}
```

Time Complexity?

(존재하는 부분 문제의 수)×(한 부분 문제를 풀 때 필요한 반복문의 수행 횟수)

#1932 정수 삼각형

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	21592	12421	9110	57.989%

문제

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

위 그림은 크기가 5인 정수 삼각형의 한 모습이다.

맨 위층 7부터 시작해서 아래에 있는 수 중 하나를 선택하여 아래층으로 내려올 때, 이제까지 선택된 수의 합이 최대가 되는 경로를 구하는 프로그램을 작성하라. 아래층에 있는 수는 현재 층에서 선택된 수의 대각선 왼쪽 또는 대각선 오른쪽에 있는 것 중에서만 선택할 수 있다.

삼각형의 크기는 1 이상 500 이하이다. 삼각형을 이루고 있는 각 수는 모두 정수이며, 범위는 0 이상 9999 이하이다.

#1932 정수 삼각형

- 갈 수 있는 방향 : 왼쪽아래, 혹은 오른쪽 아래

7

3 8

8 1 0

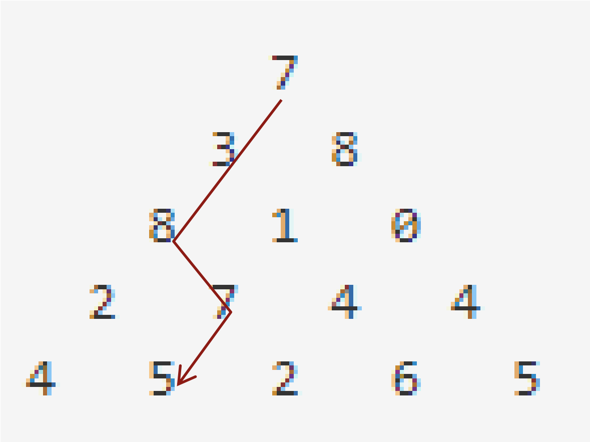
2 7 4 4

4 5 2 6 5

- 이렇게 보면? : 아래, 혹은 오른쪽 아래

#1932 정수 삼각형

1) 시작은 첫행 첫번째 원소에서 시작
2) 각 자리에서 왼쪽 아래, 혹은 오른쪽 아래로 이동가능
3) 모든 경로를 다 탐색한다면? -> 2^{n-1}가지 경우의 수
4) n은 500 이하이니 아마 안될 것이다.



점화식 구성

- $path\,(r,c)$ = (r, c)에서 시작해서 맨 아래줄까지 내려가는 부분 경로의 최대 합이라 가정하면

- $path\,(r,c) \;=\; triangle[r][c] \;+\; \max\begin{cases} path\,(r+1,c) \\ path\,(r+1,c+1) \end{cases}$

```
int n, triangle[500][500];
int cache[500][500];
int path (int r, int c) {
    if (r == n-1) return triangle[r][c];
    if (cache[r][c] != -1) return cache[r][c];
    return cache[r][c] = triangle[r][c] + max(path(r+1, c), path(r+1, c+1));
}
```

점화식 구성2

- $path\,(r,c)$ = (r, c)에 도착하였을 때의 최대 합이라 가정하면

- $path\,(r,c) \;=\; triangle[r][c] \,+\, \max\begin{cases} path\,(r-1,c) \\ path\,(r-1,c-1) \end{cases}$

```
int n, triangle[500][500];
int cache[500][500];
cache[0][0] = triangle[0][0];
for (int i = 1; i < N; i++) {
    for (int j = 0; j <= i; j++) {
        if (j == 0) cache[i][j] = triangle[i][j] + cache[i-1][j];
        else if (j == i)
            cache[i][j] = triangle[i][j] + cache[i-1][j-1];
        else
            cache[i][j] = triangle[i][j] + max(cache[i-1][j-1], cache[i-1][j];
    }
}
```

점화식 구성을 다시 봅시다.

- $path(r, c)$ = (r, c)에서 시작해서 맨 아래줄까지 내려가는 부분 경로의 최대 합이라 가정

점화식 구성을 다시 봅시다.

- $path\,(r,c)$ = (r, c)에서 시작해서 맨 아래줄까지 내려가는 부분 경로의 최대 합이라 가정

➡ (r, c)의 위쪽의 합이 path(r, c)의 값을 구하는데 영향을 주는가?

점화식 구성을 다시 봅시다.

- $path(r, c)$ = (r, c)에서 시작해서 맨 아래줄까지 내려가는 부분 경로의 최대 합이라 가정

➡ (r, c)의 위쪽의 합이 path(r, c)의 값을 구하는데 영향을 주는가?

➡ No. 즉, 남은 부분 문제는 항상 최적으로 풀어도 상관 없다!

점화식 구성을 다시 봅시다.

- $path(r,c)$ = (r, c)에서 시작해서 맨 아래줄까지 내려가는 부분 경로의 최대 합이라 가정

➡ (r, c)의 위쪽의 합이 path(r, c)의 값을 구하는데 영향을 주는가?

➡ No. 즉, 남은 부분 문제는 항상 최적으로 풀어도 상관 없다!

⬇

최적 부분 구조(Optimal substructure)

#11048 이동하기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	11408	6524	4511	58.131%

문제

준규는 N×M 크기의 미로에 갇혀있다. 미로는 1×1크기의 방으로 나누어져 있고, 각 방에는 사탕이 놓여져 있다. 미로의 가장 왼쪽 윗 방은 (1, 1)이고, 가장 오른쪽 아랫 방은 (N, M)이다.

준규는 현재 (1, 1)에 있고, (N, M)으로 이동하려고 한다. 준규가 (r, c)에 있으면, (r+1, c), (r, c+1), (r+1, c+1)로 이동할 수 있고, 각 방을 방문할 때마다 방에 놓여져있는 사탕을 모두 가져갈 수 있다. 또, 미로 밖으로 나갈 수는 없다.

준규가 (N, M)으로 이동할 때, 가져올 수 있는 사탕 개수의 최댓값을 구하시오.

입력

첫째 줄에 미로의 크기 N, M이 주어진다. (1 ≤ N, M ≤ 1,000)

둘째 줄부터 N개 줄에는 총 M개의 숫자가 주어지며, r번째 줄의 c번째 수는 (r, c)에 놓여져 있는 사탕의 개수이다. 사탕의 개수는 0보다 크거나 같고, 100보다 작거나 같다.

#11048 이동하기

- 이동 가능 방향은 오른쪽, 오른쪽 아래, 아래 총 3방향
- (N, M)까지 가는 경로에 있는 수들의 합이 최대

점화식 구성

- $path(r,c)$ = (r, c)에 도착하였을 때의 최대 합이라 가정하면

- $path(r,c) = candy[r][c] + max\begin{cases} path(r-1,c) \\ path(r-1,c-1) \\ path(r,c-1) \end{cases}$

```
int candy[MAX][MAX], cache[MAX][MAX], N, M;
cache[0][0] = candy[0][0];
for (int i = 1; i < N; i++)
    cache[i][0] = cache[i-1][0] + candy[i][0];
for (int j = 1; j < N; j++)
    cache[0][j] = cache[0][j-1] + candy[0][j];
for (int i = 1; i < N; i++)
    for (int j = 1; j < N; j++)
        cache[i][j] = candy[i][j] + max3(cache[i-1][j], cache[i-1][j-1],
                                         cache[i][j-1]);
```

Problem Set

1463 1로 만들기

9095 1,2,3 더하기

2579 계단 오르기

1003 피보나치 함수

1149 RGB거리

1932 정수 삼각형

1912 연속합

10844 쉬운 계단 수

1010 다리 놓기

11048 이동하기

11051 이항 계수2

1520 내리막 길