

# 다이나믹 프로그래밍

최백준 [choi@startlink.io](mailto:choi@startlink.io)

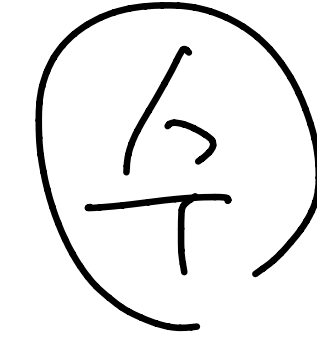
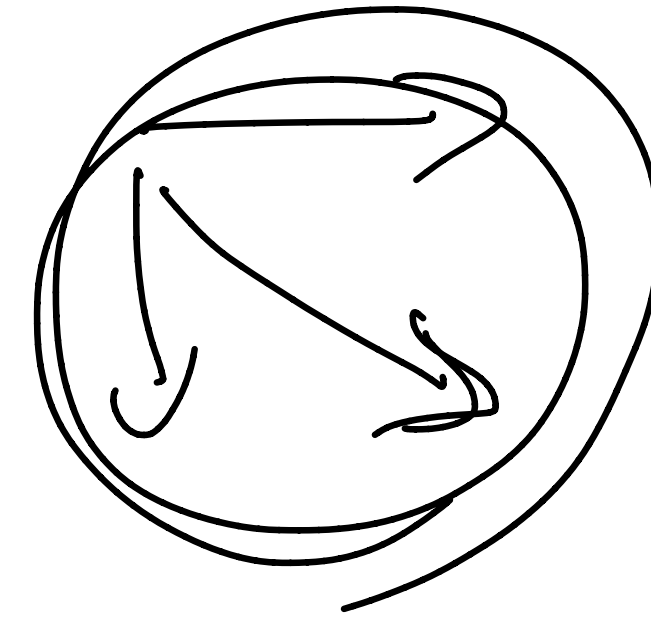
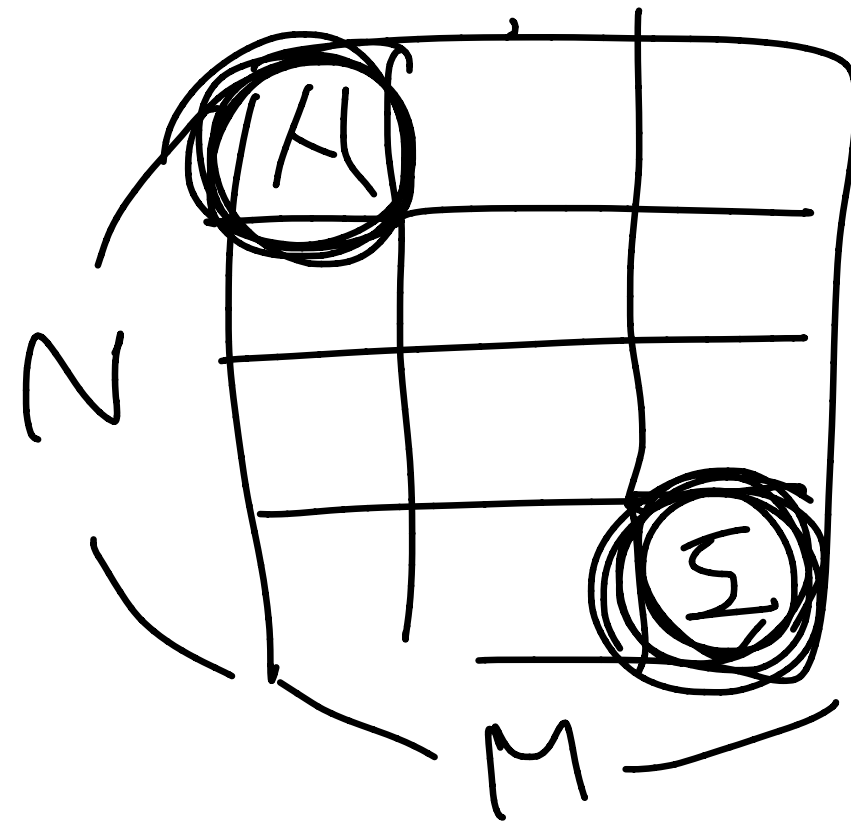
---

# 이동하기

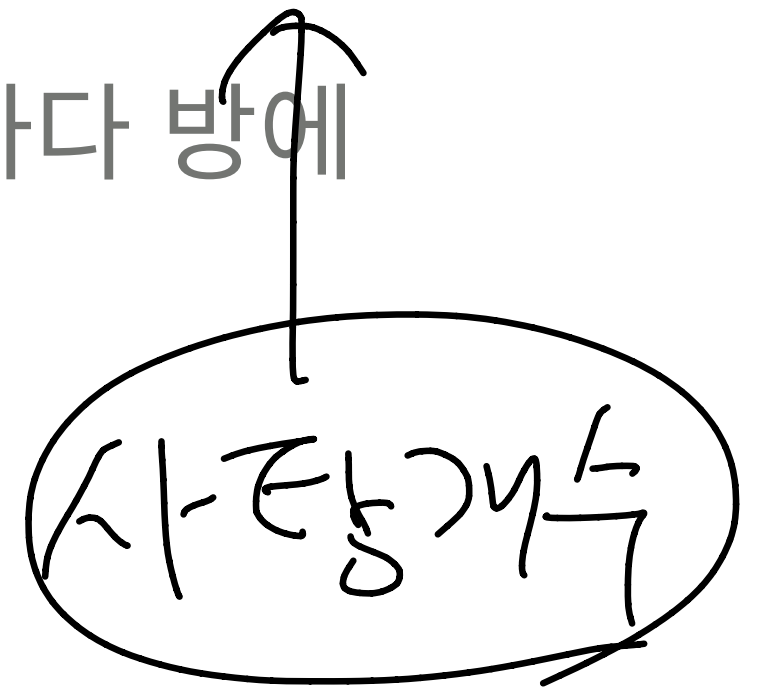
<https://www.acmicpc.net/problem/11048>

2

- 준규는  $N \times M$  크기의 미로에 갇혀있다
- 미로는  $1 \times 1$  크기의 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  $(i, j)$ 에 있으면,  $(i+1, j)$ ,  $(i, j+1)$ ,  $(i+1, j+1)$ 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 최대값



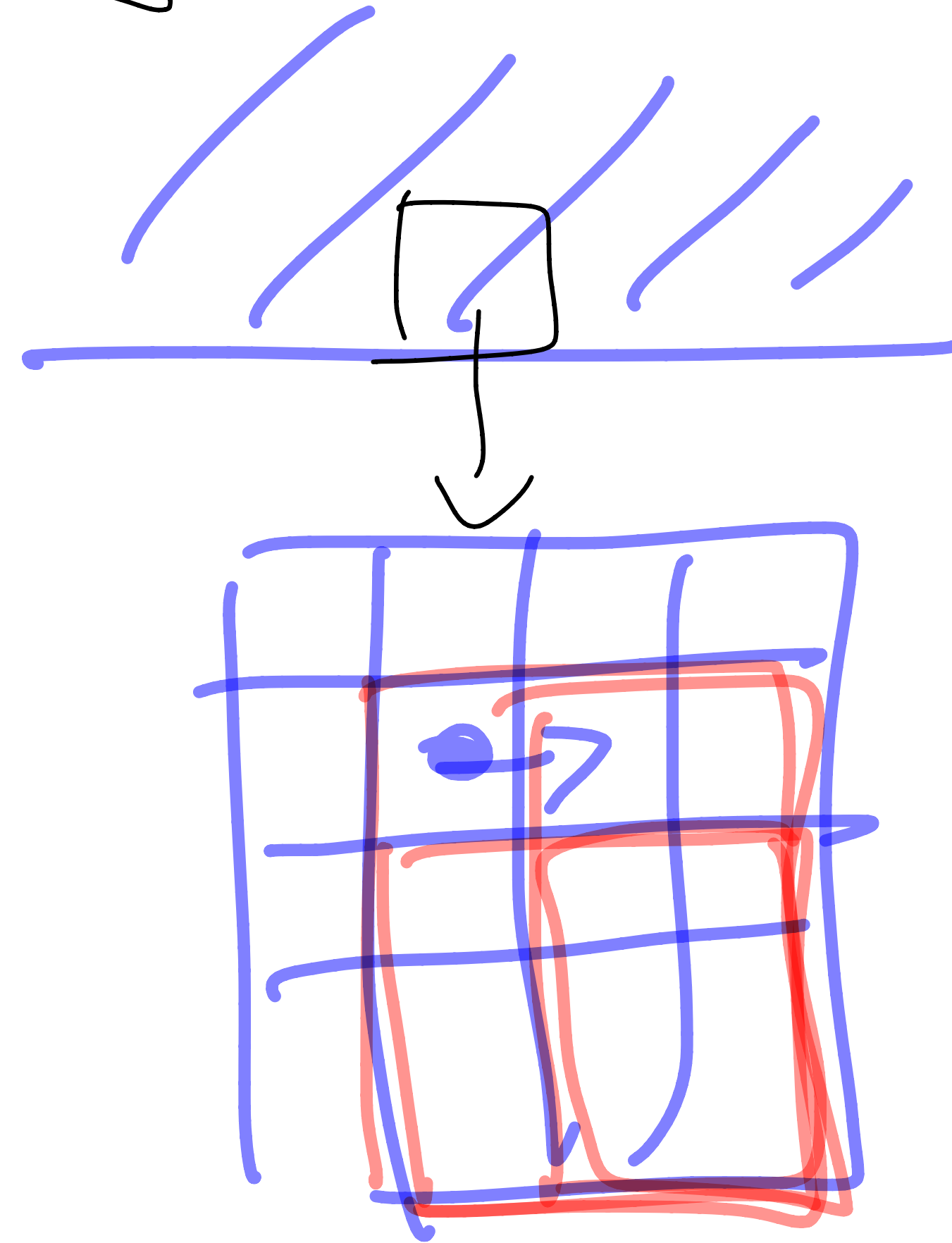
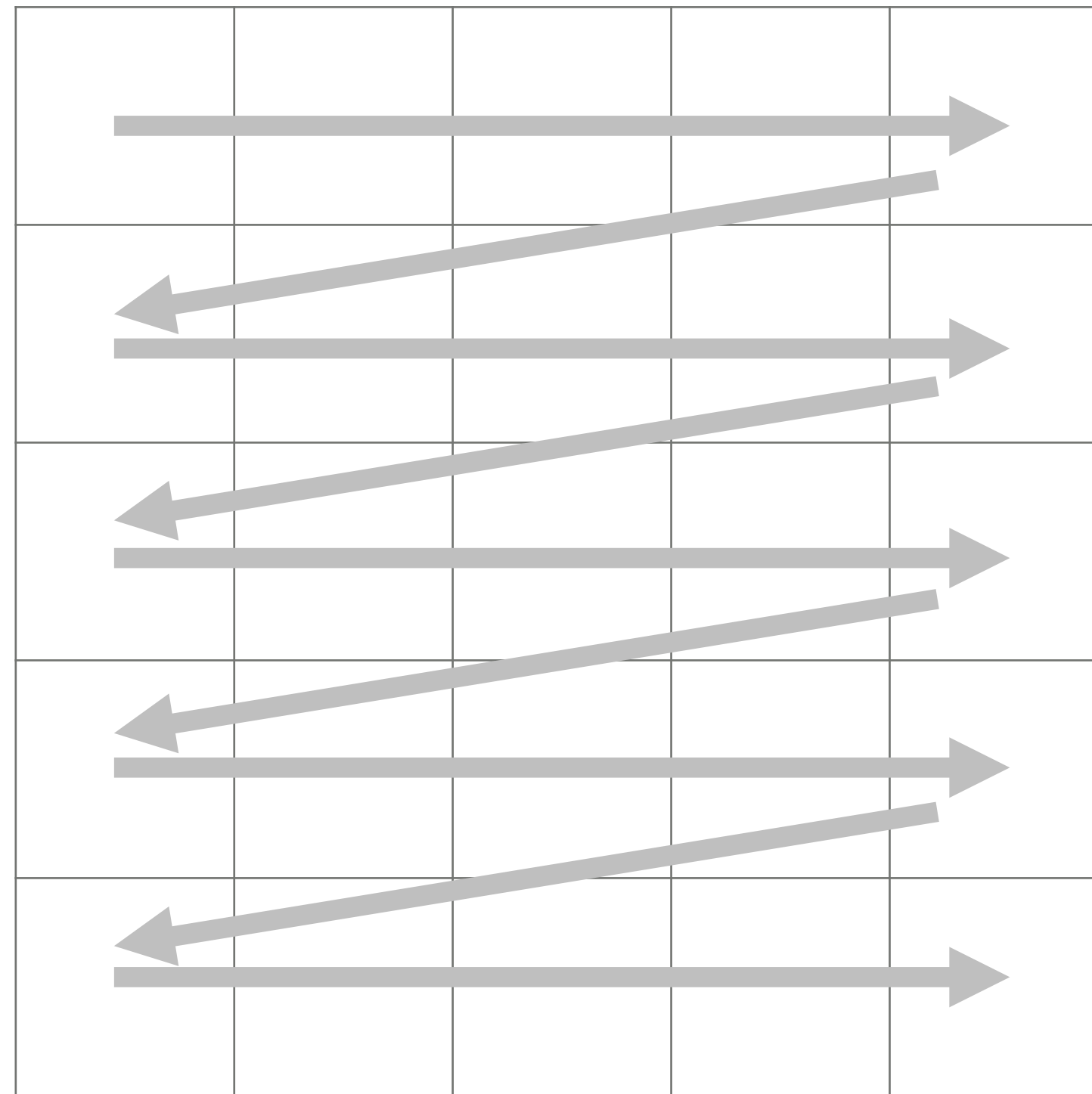
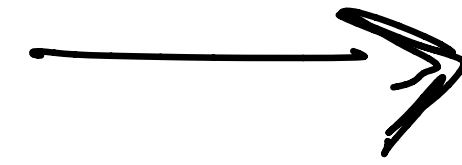
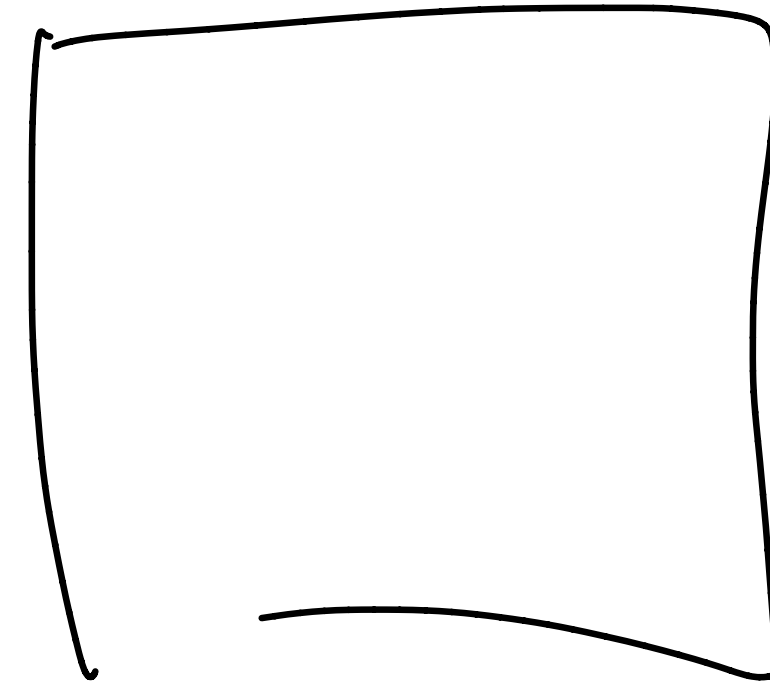
$(T, S)$  ATCCTCCT



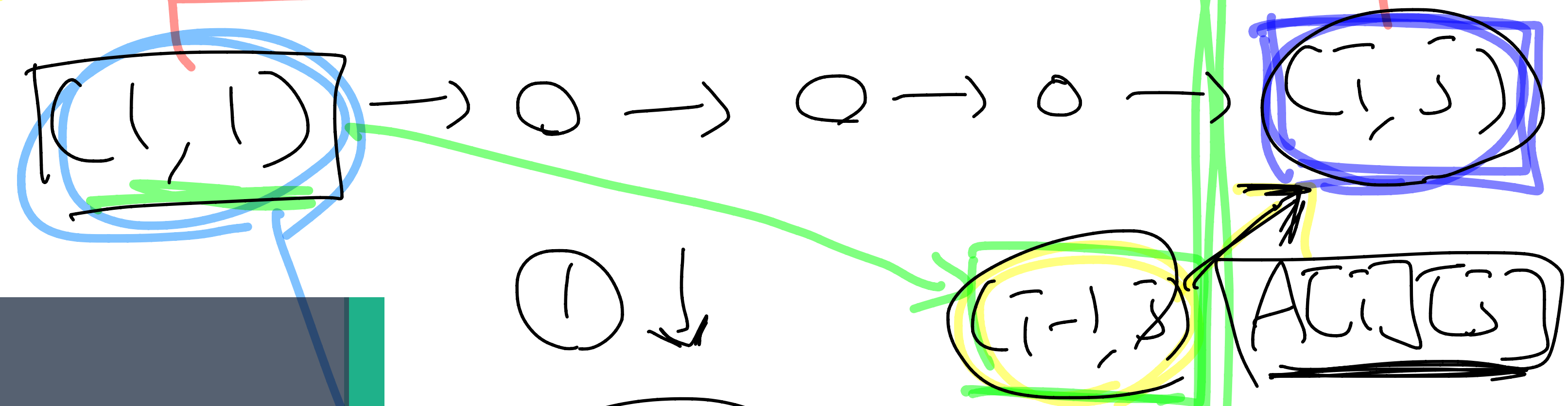
# 이동하기

<https://www.acmicpc.net/problem/11048>

- 항상 아래와 오른쪽으로만 갈 수 있다.
- $(i,j)$ 에서 가능한 이동:  $(i+1, j)$ ,  $(i, j+1)$ ,  $(i+1, j+1)$



$D[i][j] = (1, 1)$  시작,  $(i, j)$  도착 행의 최더값



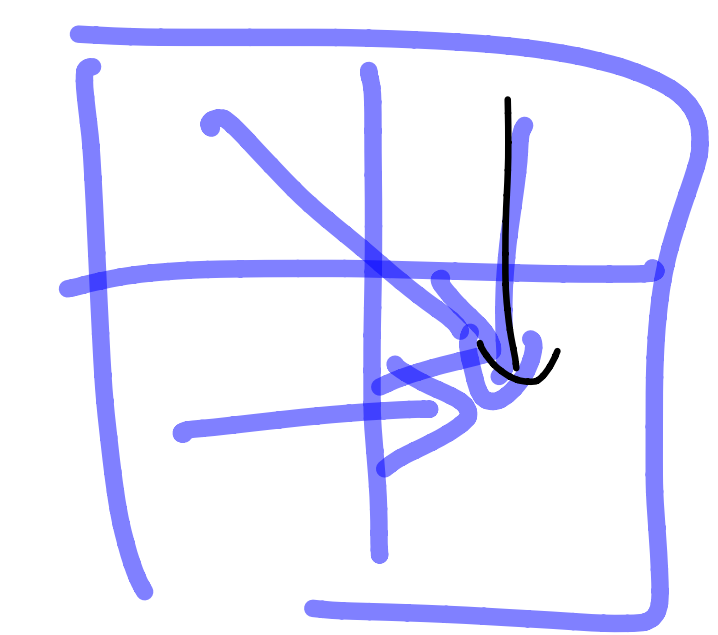
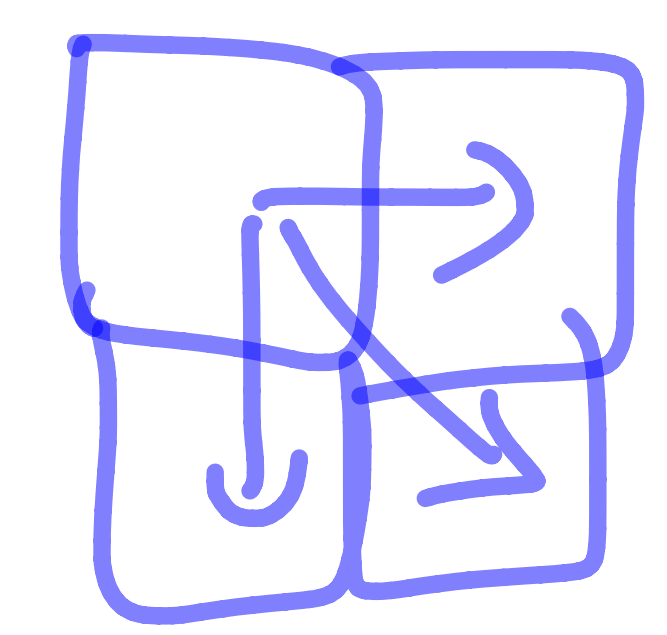
① ↓

최더값

$$D[i-1][j] + A[i][j]$$

② →  $(i, j-1) \rightarrow (i, j)$   
 $D[i][j-1] + A[i][j]$

# 방법 1

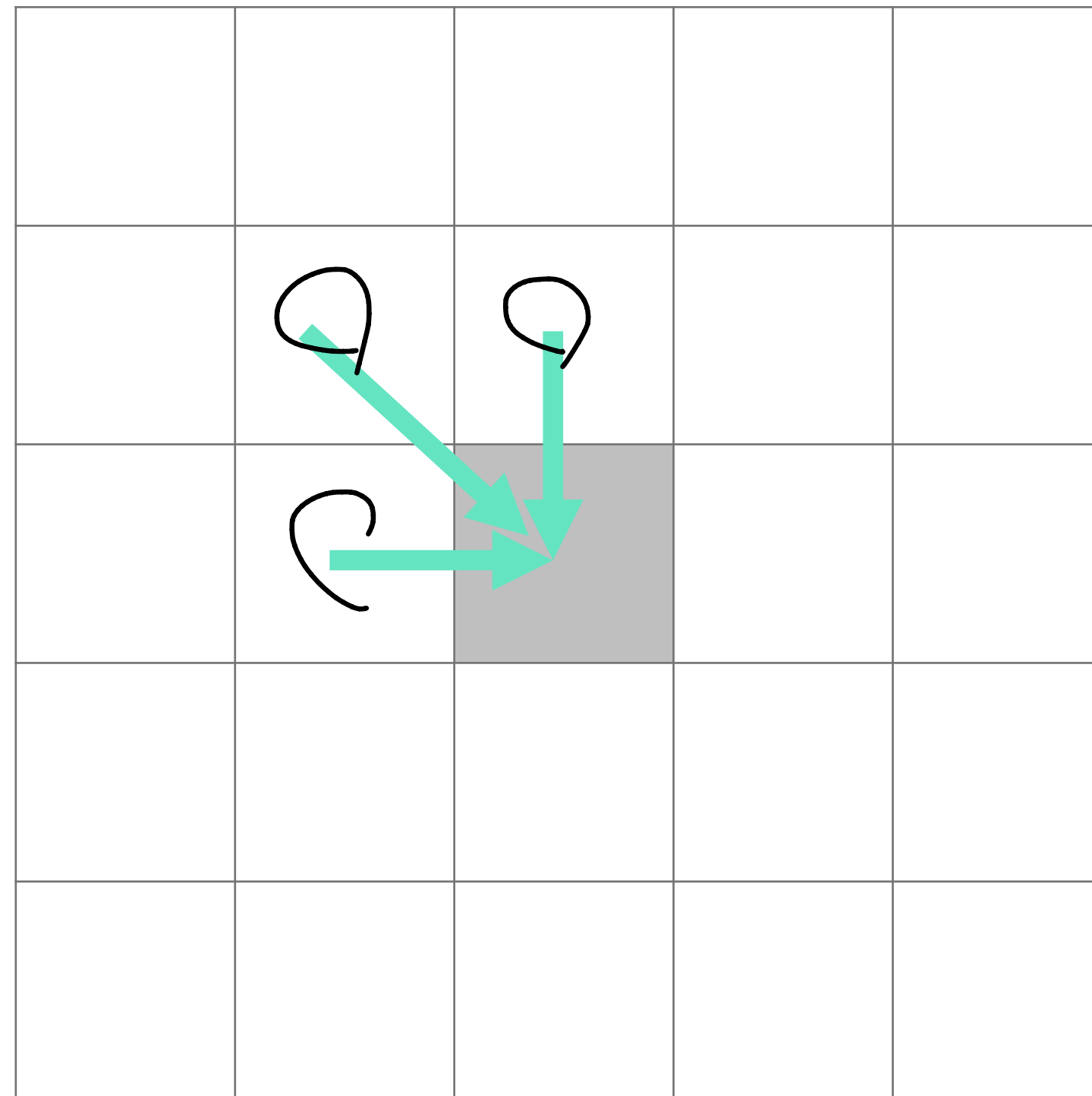


# 이동하기

<https://www.acmicpc.net/problem/11048>

- 항상 아래와 오른쪽으로만 갈 수 있다.
- $(i,j)$ 에서 가능한 이동:  $(i+1, j)$ ,  $(i, j+1)$ ,  $(i+1, j+1)$

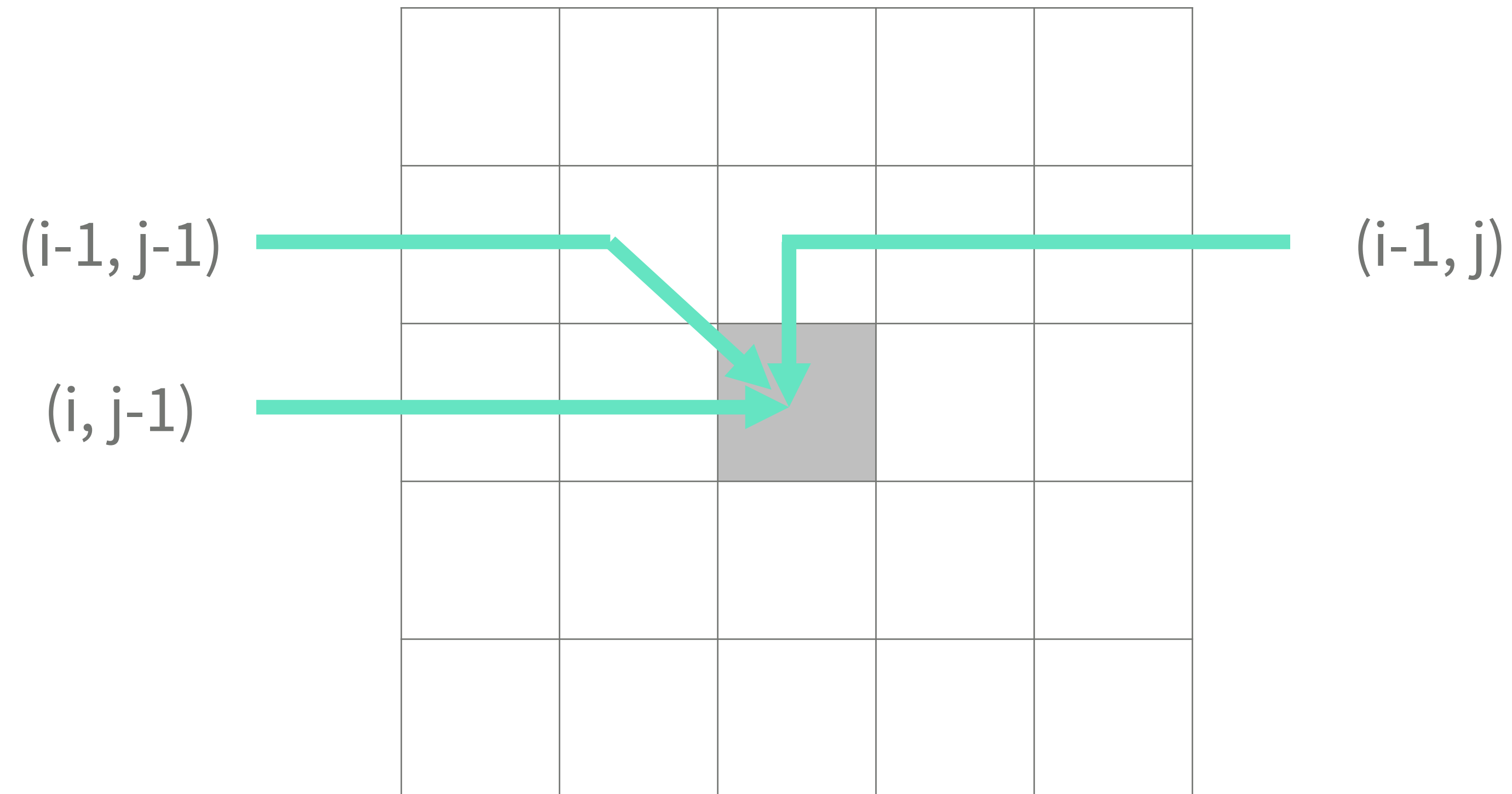
$$D[i][j] = \max \begin{pmatrix} D[i-1][j] \\ D[i][j-1] \\ D[i-1][j-1] \end{pmatrix} + A[i][j]$$



# 이동하기

<https://www.acmicpc.net/problem/11048>

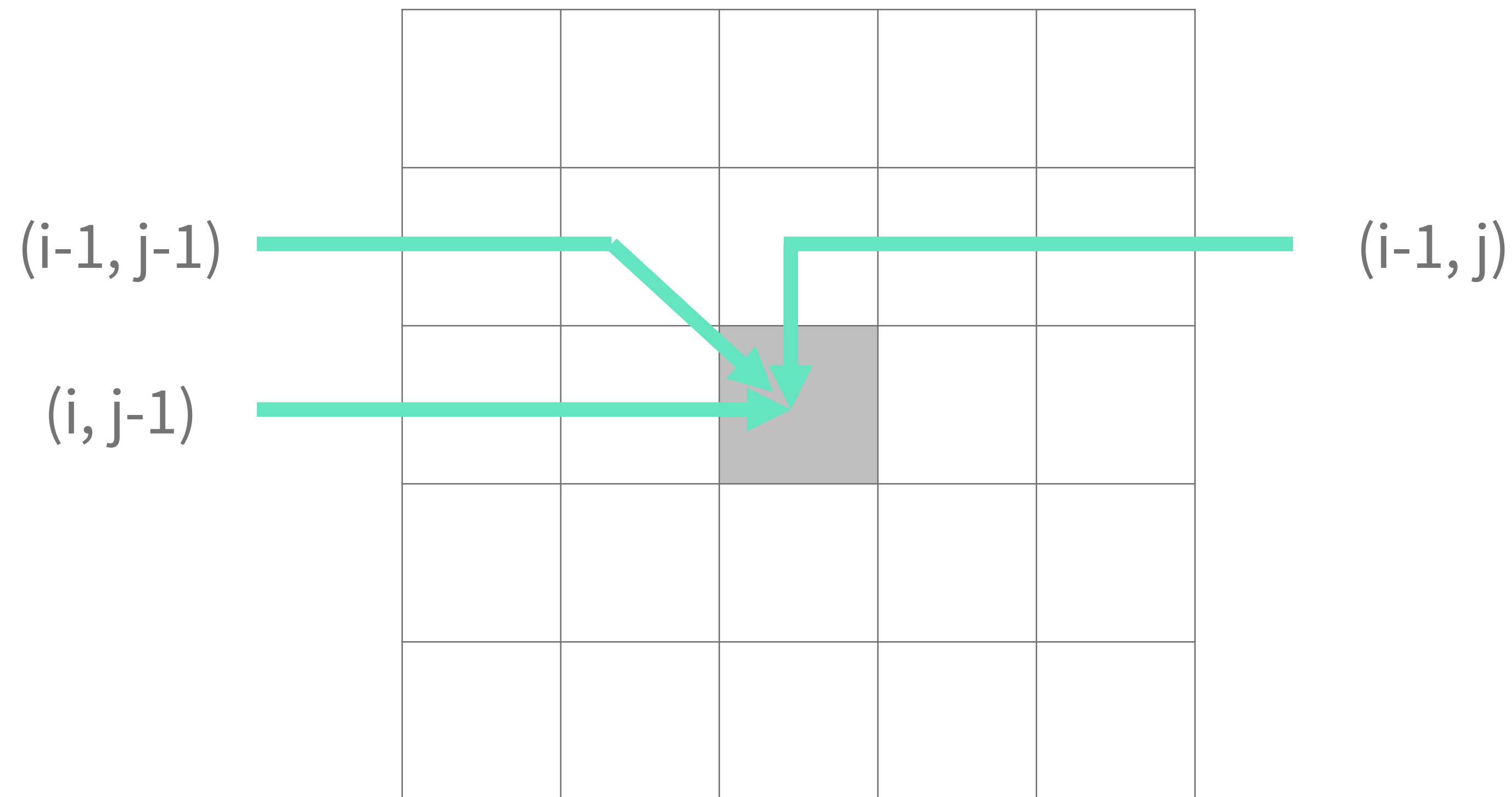
- 항상 아래와 오른쪽으로만 갈 수 있다.
- $(i,j)$ 에서 가능한 이동:  $(i+1, j)$ ,  $(i, j+1)$ ,  $(i+1, j+1)$



# 이동하기

<https://www.acmicpc.net/problem/11048>

- $D[i][j] = (i, j)$ 로 이동할 때 가져올 수 있는 최대 사탕 개수
- $D[i][j] = \text{Max}(D[i-1][j-1], D[i][j-1], D[i-1][j]) + A[i][j]$



# 이동하기

$(1,1) \sim (N,M)$

$\downarrow [n][m]$

8

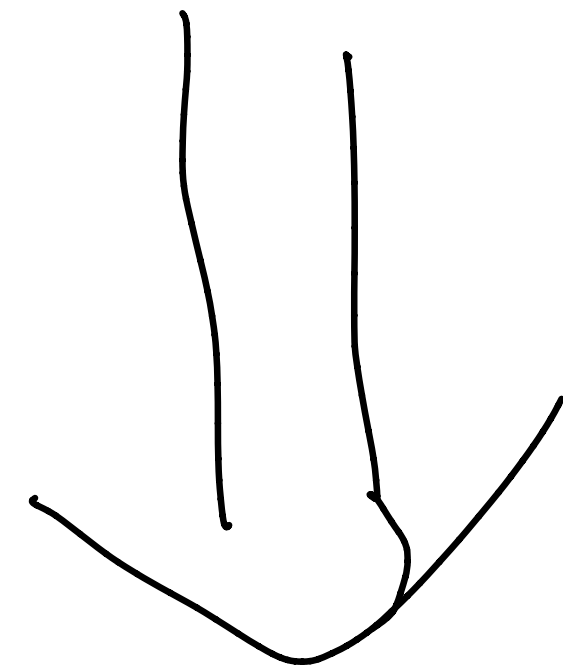
<https://www.acmicpc.net/problem/11048>

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max3(d[i-1][j], d[i][j-1], d[i-1][j-1]) + a[i][j];  
    }  
}
```

$\max(\max(a,b), c)$

$\max3(a,b,c)$   
 $a > b, a > c \quad \text{ret}$

$\max(a,b) \{$   
if  $a > b$  return  $a$ ;  
else return  $b$ ;





# 이동하기

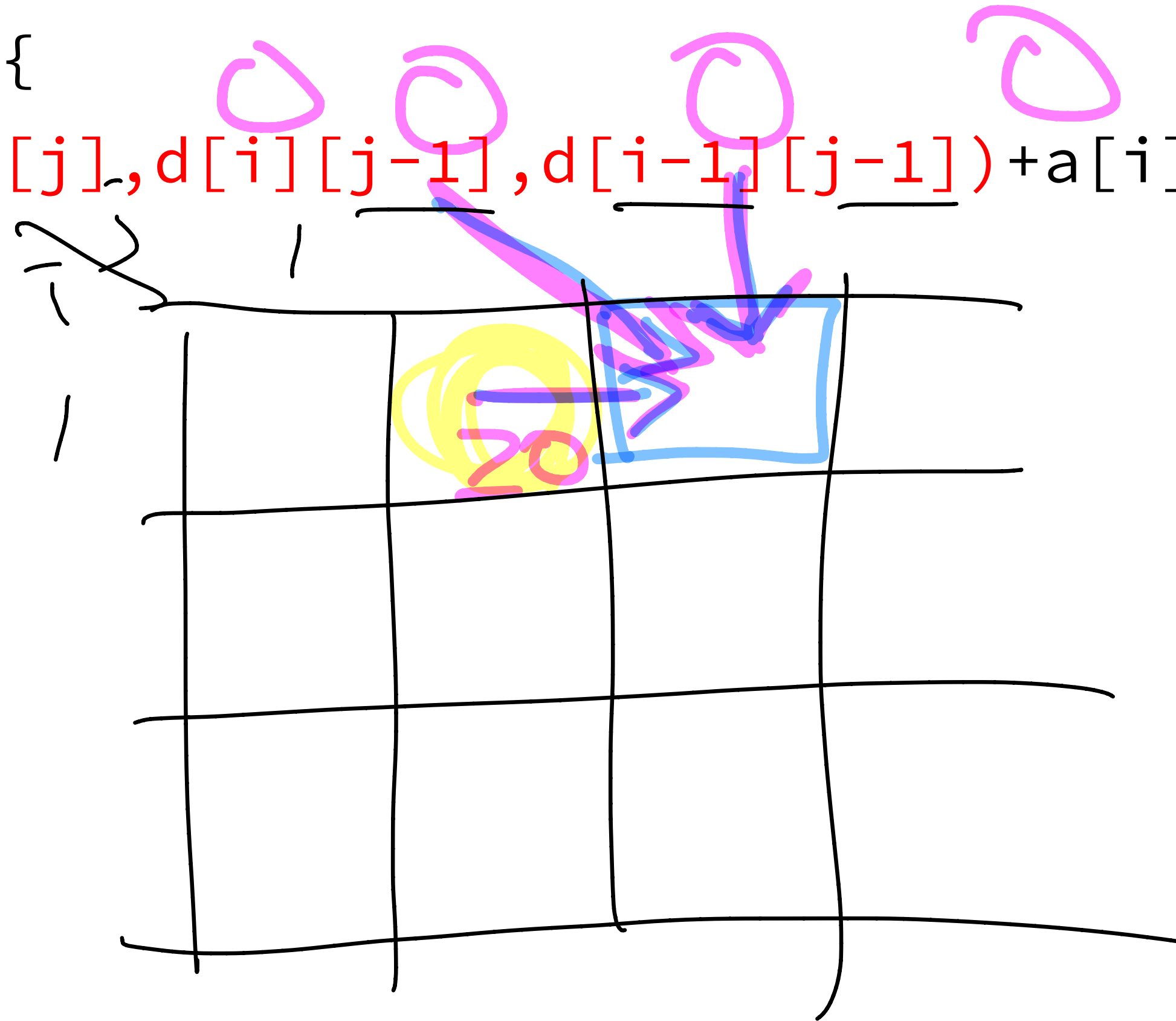
<https://www.acmicpc.net/problem/11048>

9

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max3(d[i-1][j], d[i][j-1], d[i-1][j-1]) + a[i][j];  
    }  
}
```

- $i-1, j-1$  범위 검사를 하지 않은 이유
- $i=1, j=1$ 인 경우
- $i=1$ 인 경우
- $j=1$ 인 경우

$$\boxed{A[i][j]} \geq 0$$



# 이동하기

<https://www.acmicpc.net/problem/11048>

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max3(d[i-1][j], d[i][j-1], d[i-1][j-1]) + a[i][j];  
    }  
}
```

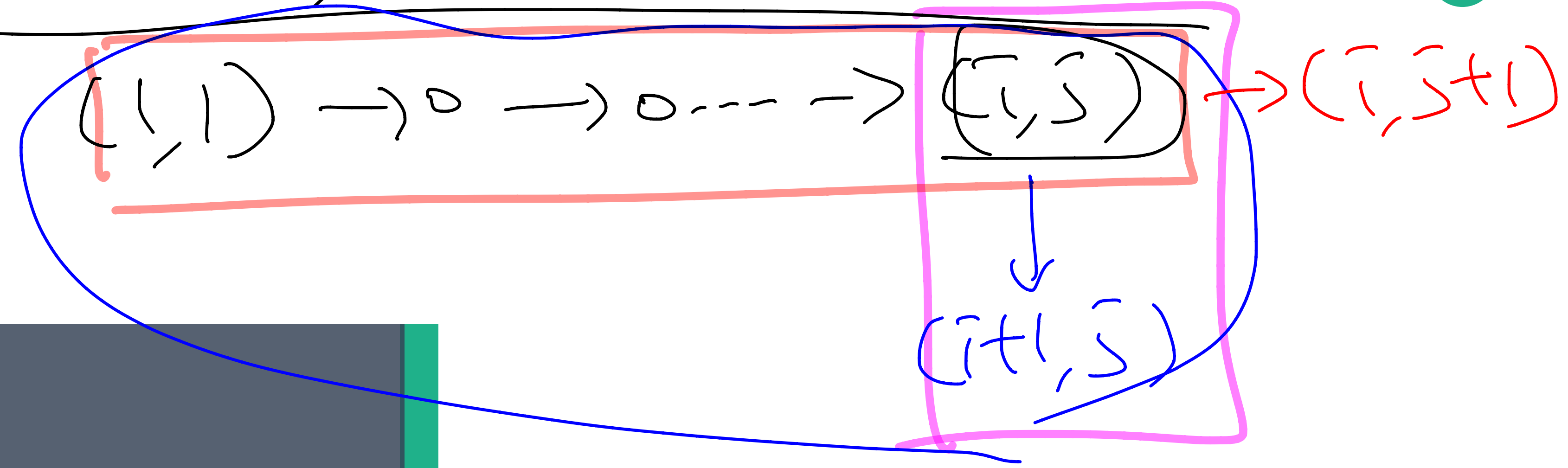
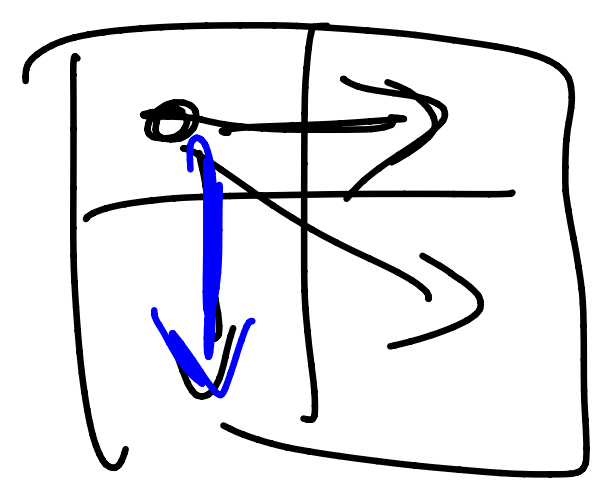
- $i-1, j-1$  범위 검사를 하지 않은 이유
- $i = 1, j = 1$ 인 경우:  $d[i-1][j], d[i][j-1], d[i-1][j-1]$ 은 0이기 때문
- $i = 1$ 인 경우:  $d[i-1][j] = 0 < d[i][j-1]$  이기 때문
- $j = 1$ 인 경우:  $d[i][j-1] = 0 < d[i-1][j]$  이기 때문

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 소스: <http://codeplus.codes/696019b79e67491da22ecbc22a806c98>

$D[i][j] = (1,1)$  시작,  $(i,j)$  도착 횟수



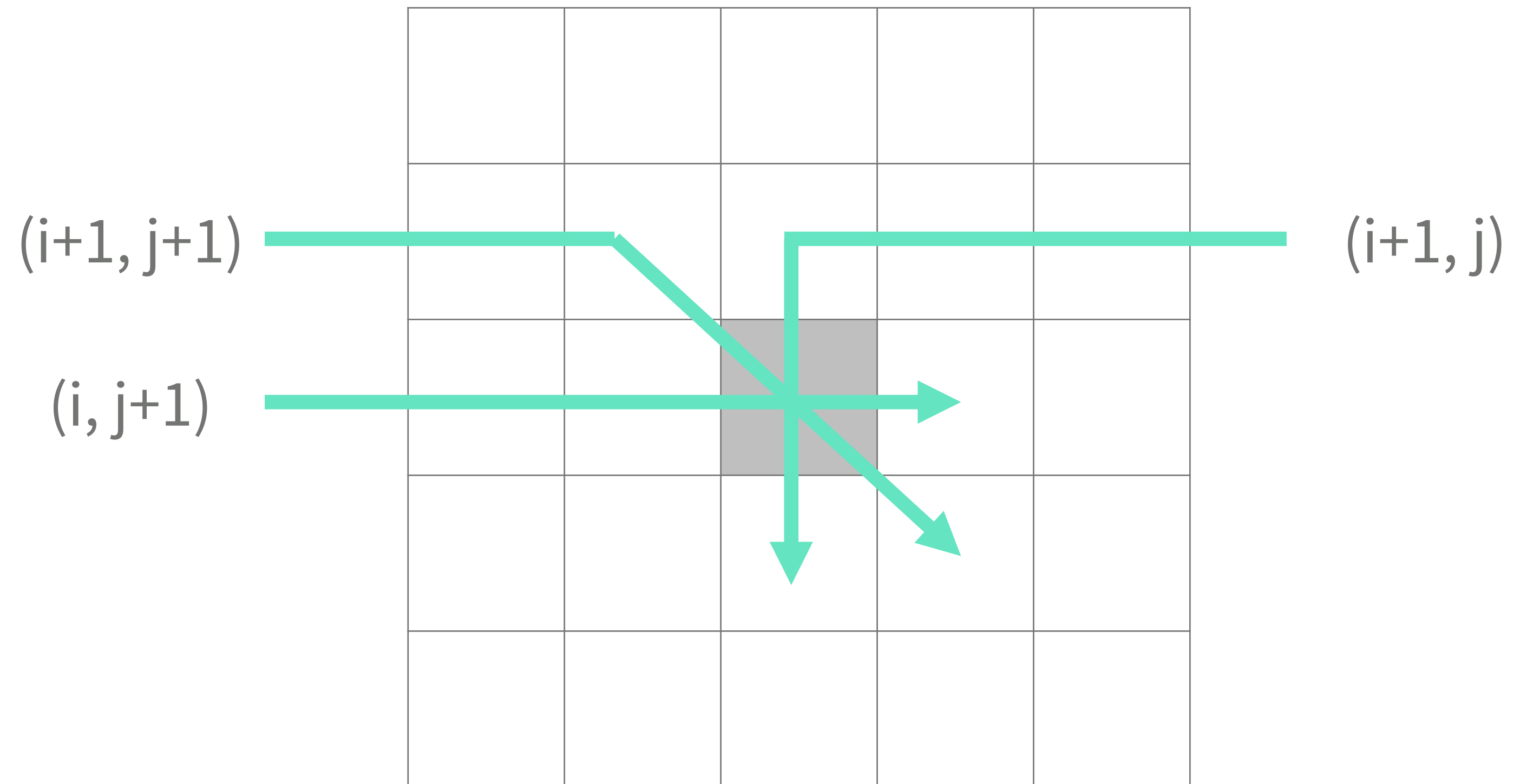
## 방법 2

$$\begin{aligned} D[i+1][j] &= \max(D[i+1][j], D[i][j] + A[i+1][j]) \\ D[i][j+1] &= \max(D[i][j+1], D[i][j] + A[i][j+1]) \end{aligned}$$

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 항상 아래와 오른쪽으로만 갈 수 있다.
- $(i,j)$ 에서 가능한 이동:  $(i+1, j)$ ,  $(i, j+1)$ ,  $(i+1, j+1)$



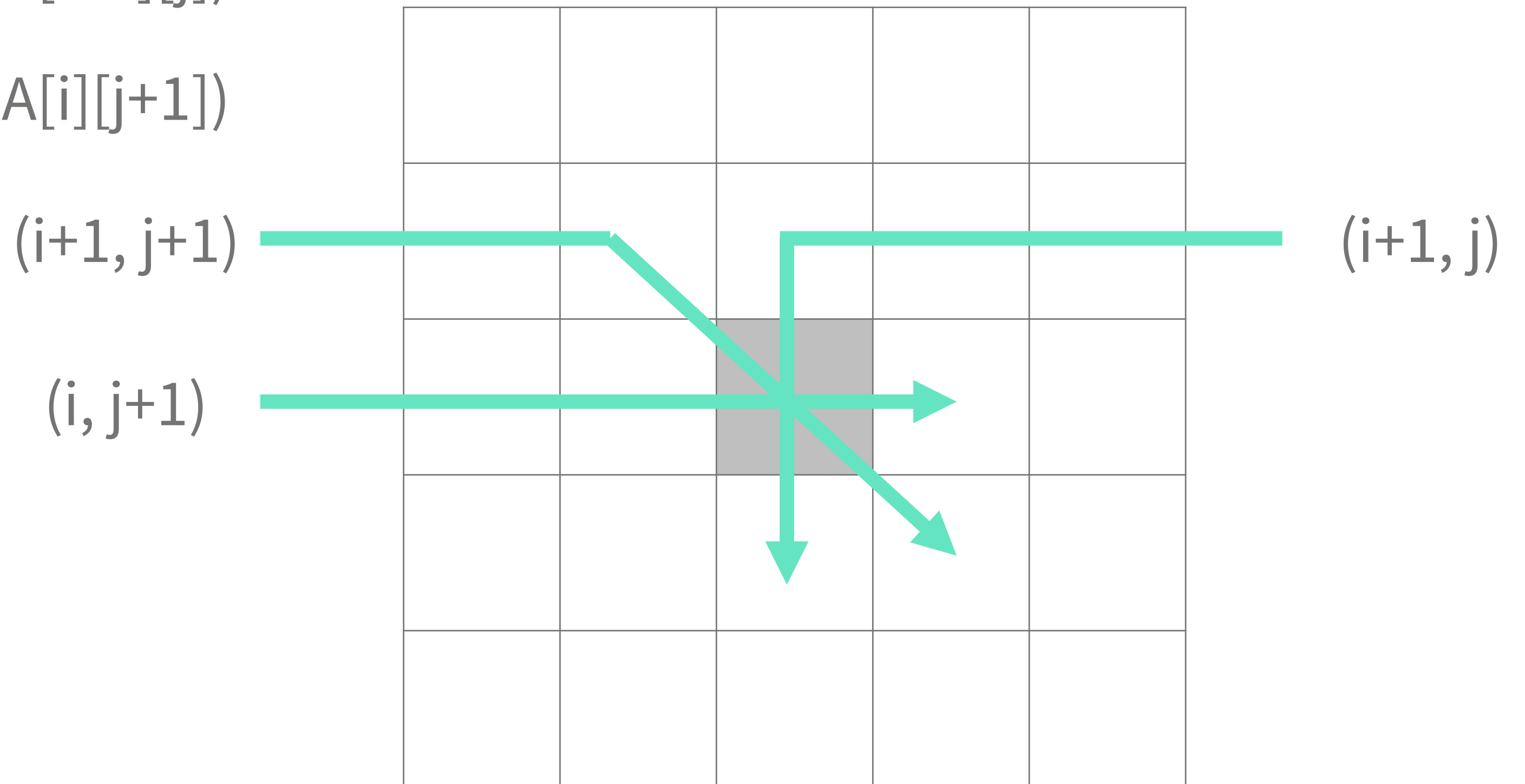
# 이동하기

남남 1로 정의 같아요!

14

<https://www.acmicpc.net/problem/11048>

- $D[i][j] = (i, j)$ 로 이동할 때 가져올 수 있는 최대 사탕 개수
- $D[i+1][j+1] = \max(D[i+1][j+1], D[i][j] + A[i+1][j+1])$
- $D[i+1][j] = \max(D[i+1][j], D[i][j] + A[i+1][j])$
- $D[i][j+1] = \max(D[i][j+1], D[i][j] + A[i][j+1])$



# 이동하기

<https://www.acmicpc.net/problem/11048>

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        if (d[i][j+1] < d[i][j] + a[i][j+1]) {  
            d[i][j+1] = d[i][j] + a[i][j+1];  
        }  
        if (d[i+1][j] < d[i][j] + a[i+1][j]) {  
            d[i+1][j] = d[i][j] + a[i+1][j];  
        }  
        if (d[i+1][j+1] < d[i][j] + a[i+1][j+1]) {  
            d[i+1][j+1] = d[i][j] + a[i+1][j+1];  
        }  
    }  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 소스: <http://codeplus.codes/f13b048d1ee64876a020827c2cfbb0be>



# 방법 3

---

# 이동하기

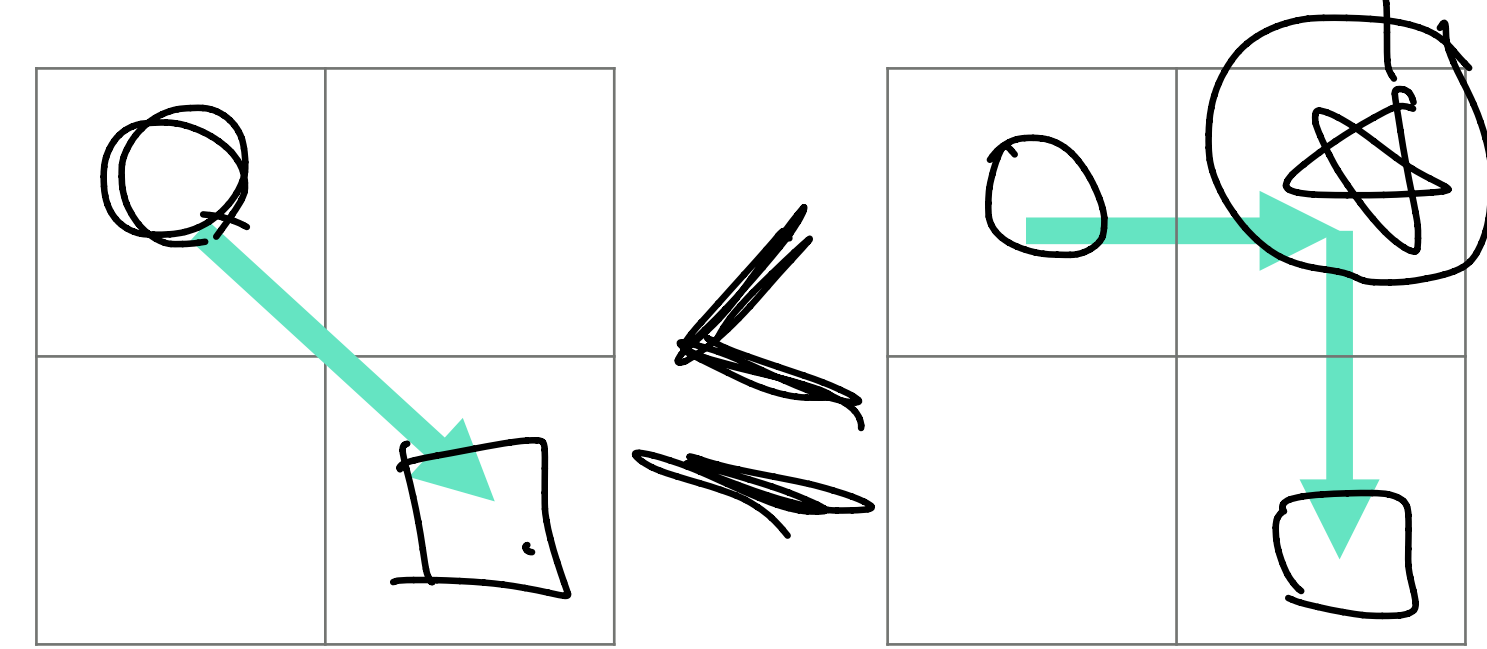
<https://www.acmicpc.net/problem/11048>

$$A[i][j] + A[i+1][j+1] \geq 0$$

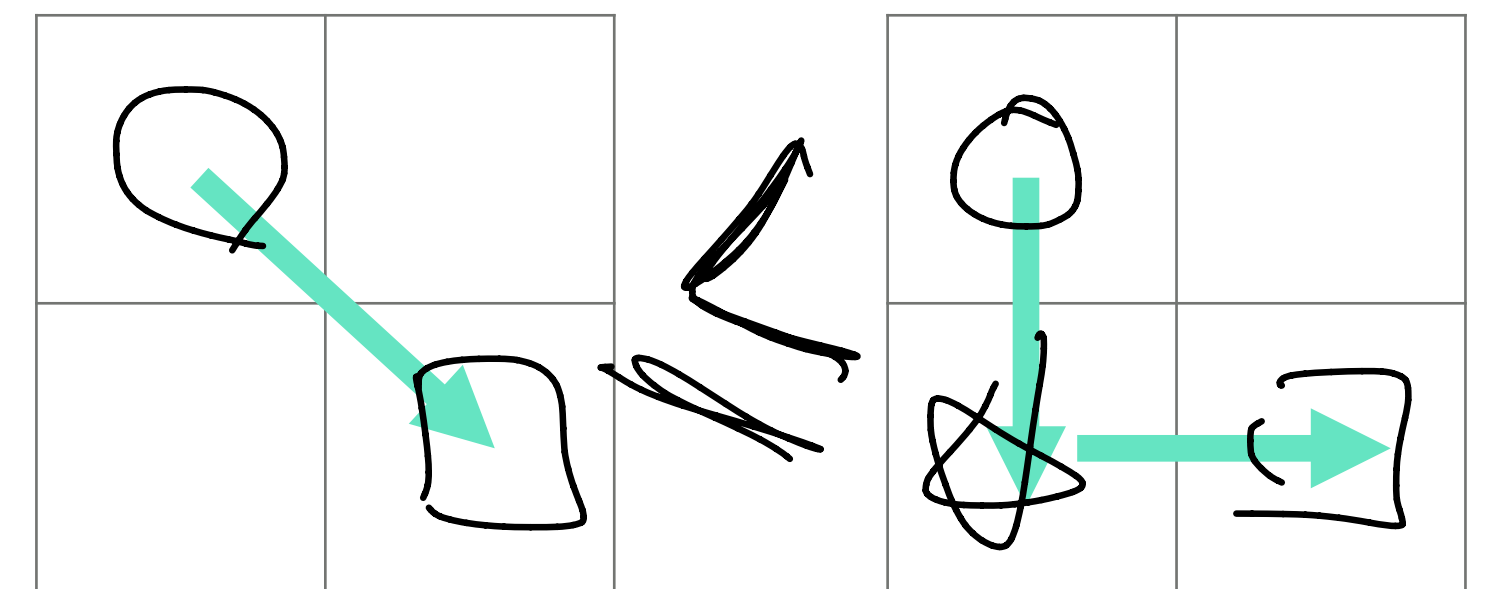
18

- 대각선 이동은 처리하지 않아도 된다
- 대각선 이동은 다른 2가지를 포함한 방법보다 항상 작거나 같다

- $A[i][j] + A[i+1][j+1] \leq A[i][j] + A[i][j+1] + A[i+1][j+1]$



- $A[i][j] + A[i+1][j+1] \leq A[i][j] + A[i+1][j] + A[i+1][j+1]$



# 이동하기

<https://www.acmicpc.net/problem/11048>

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 소스: <http://codeplus.codes/c6c6bf2e936f4532b9b7b3f67479ffe2>

# 방법 4

---

# 이동하기

<https://www.acmicpc.net/problem/11048>

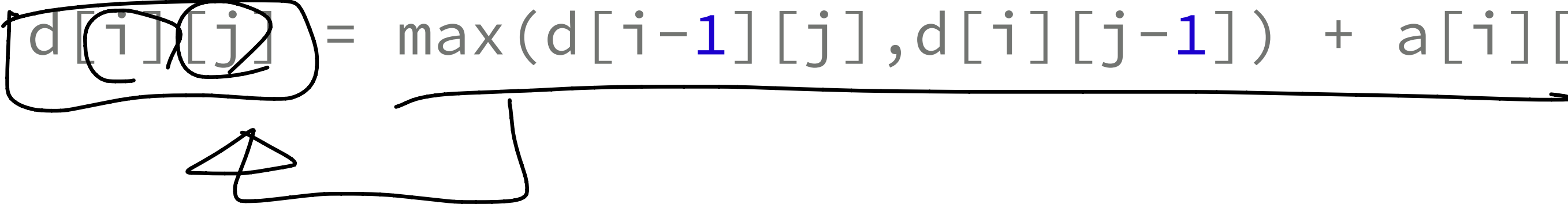
- 재귀 함수를 이용해서도 구현할 수 있다
- $D[i][j] = (i, j)$ 로 이동할 때 가져올 수 있는 최대 사탕 개수
- $D[i][j] = \max(D[i][j-1], D[i-1][j]) + A[i][j]$
- 식이 달라지는 것이 아니고 구현 방식이 달라지는 것이다

# 이동하기

<https://www.acmicpc.net/problem/11048>

// Bottom-up

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}  
cout << d[n][m] << '\n';
```



# 이동하기

<https://www.acmicpc.net/problem/11048>

// Bottom-up

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {
```

```
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];
```

```
    }
```

```
}
```

```
cout << d[n][m] << '\n';
```

// Top-down

```
int go(int i, int j) {
```

```
}
```





# 이동하기

<https://www.acmicpc.net/problem/11048>

// Bottom-up

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}
```

```
cout << d[n][m] << '\n';
```

// Top-down

```
int go(int i, int j) {  
    d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

// Bottom-up

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}  
cout << d[n][m] << '\n';
```

// Top-down

```
int go(int i, int j) {  
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

// Bottom-up

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}
```

```
cout << d[n][m] << '\n';
```

// Top-down

```
int go(int i, int j) {  
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];  
    return d[i][j];  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

// Bottom-up

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}
```

```
cout << d[n][m] << '\n';
```

// Top-down

```
int go(int i, int j) {  
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];  
    return d[i][j];  
}
```

```
cout << d[n][m] << '\n';
```

# 이동하기

29

<https://www.acmicpc.net/problem/11048>

$(1, 1) \sim (N, M)$

// Bottom-up

```
for (int i=1, i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
        d[i][j] = max(d[i-1][j], d[i][j-1]) + a[i][j];  
    }  
}
```

```
cout << d[n][m] << '\n';
```

// Top-down

```
int go(int i, int j) {  
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];  
    return d[i][j];  
}
```

```
cout << go(n, m) << '\n';
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

```
// Bottom-up
```

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=m; j++) {  
    }
```

```
}
```

```
cout << d[n][m] << '\n';
```

```
// Top-down
```

```
int go(int i, int j) {  
    if (i < 1 || j < 1) return 0;  
    d[i][j] = max(go(i-1,j), go(i,j-1)) + a[i][j];  
    return d[i][j];
```

```
}
```

```
cout << go(n,m) << '\n';
```

# 이동하기

31

<https://www.acmicpc.net/problem/11048>

```
int go(int i, int j) {  
    if (i < 1 || j < 1) {  
        return 0;  
    }
```

```
    if (d[i][j] > 0) {  
        return d[i][j];  
    }
```

```
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];  
    return d[i][j];  
}
```

A[i][j] ≥ 0

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 시간 초과를 받게 된다.
- 사탕의 개수가 0보다 크거나 같기 때문에
- $d[i][j]$ 에 들어있는 값이 0인 경우가 답을 구했는데, 0일 수도 있다.
- 따라서, 같은 문제의 정답을 여러 번 구하게 된다.



# 이동하기

<https://www.acmicpc.net/problem/11048>

- 답이 음수인 경우는 절대로 없기 때문에, 배열을 -1로 초기화 하고 사용한다.

# 이동하기

<https://www.acmicpc.net/problem/11048>

```
int go(int i, int j) {  
    if (i < 1 || j < 1) {  
        return 0;  
    }  
    if (d[i][j] >= 0) {  
        return d[i][j];  
    }  
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];  
    return d[i][j];  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 소스: <http://codeplus.codes/7ba619a0503a470abf3c2f85825b42fe>

# 방법 5

---

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 방법 1~4의 점화식은 모두 같았는데
- 구현 방식, 식을 채우는 순서만 조금씩 달랐다

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 점화식을 조금 바꿔서 세워보자
- $D[i][j] = (i, j)$ 에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- 지금까지의 점화식
- $D[i][j] = \text{ $(i, j)$ 로 이동했을 때, 가져올 수 있는 최대 사탕 개수}$

$(i, j)$  시작

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 점화식을 조금 바꿔서 세워보자
- $D[i][j] = (i, j)$ 에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- 도착( $N, M$ )으로 정해져 있는데, 시작( $i, j$ )을 이동시키는 방식
- 지금까지의 점화식
- $D[i][j] = (i, j)$ 로 이동했을 때, 가져올 수 있는 최대 사탕 개수
- 시작은  $(1, 1)$ 로 정해져 있고, 도착  $(i, j)$ 을 이동시키는 방식

# 이동하기

<https://www.acmicpc.net/problem/11048>

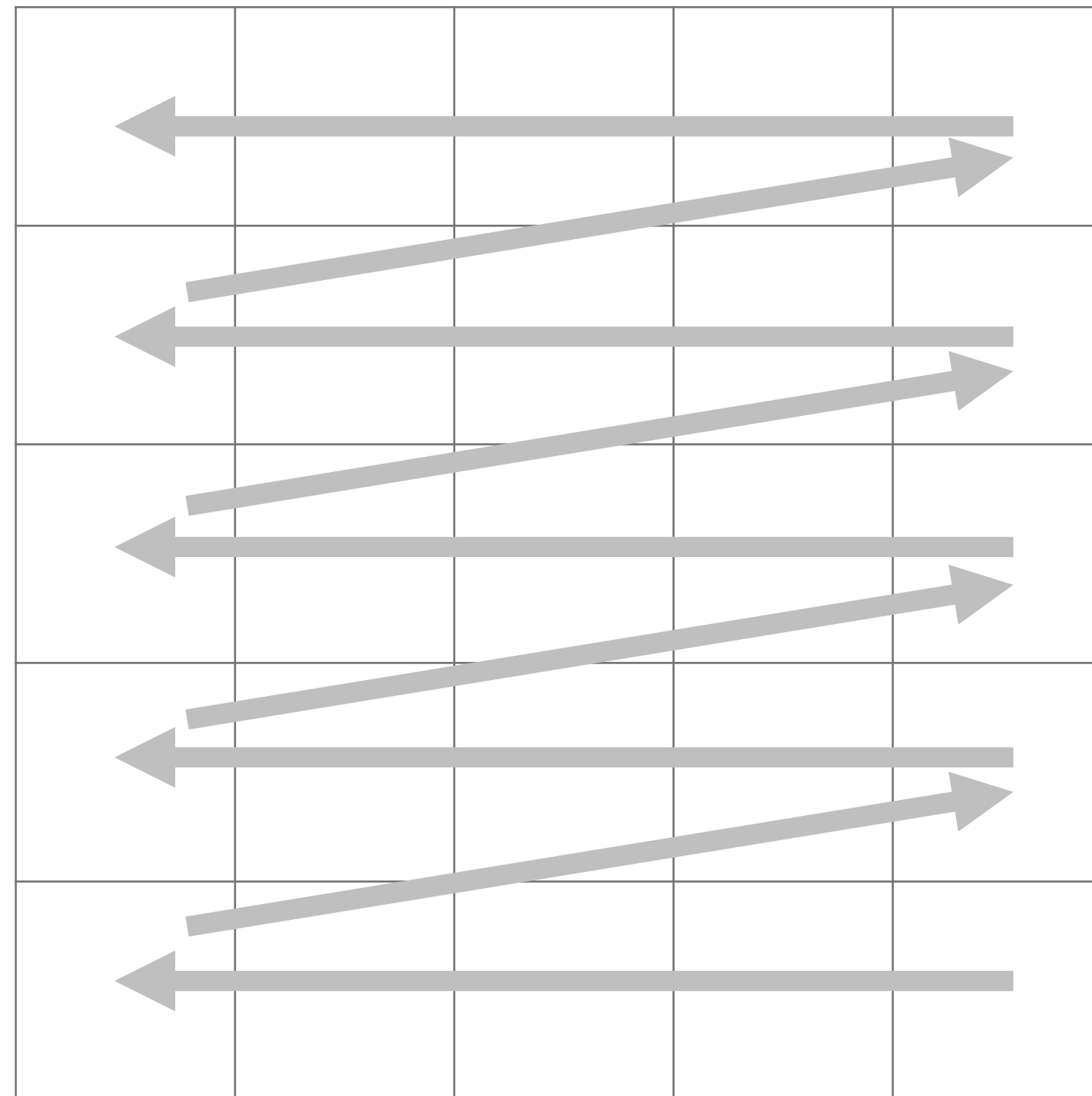
- $D[i][j]$  = (i, j)에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- $D[i][j] = \max(D[i+1][j], D[i][j+1]) + A[i][j]$



# 이동하기

<https://www.acmicpc.net/problem/11048>

- 항상 아래와 오른쪽으로만 갈 수 있다.
- $(i,j)$ 에서 가능한 이동:  $(i+1, j)$ ,  $(i, j+1)$ ,  $(i+1, j+1)$



# 이동하기

<https://www.acmicpc.net/problem/11048>

```
int go(int x, int y) {  
    if (x > n || y > m) return 0;  
    if (d[x][y] >= 0) return d[x][y];  
    d[x][y] = max(go(x+1,y), go(x,y+1)) + a[x][y];  
    return d[x][y];  
}
```

# 이동하기

<https://www.acmicpc.net/problem/11048>

- $D[i][j] = (i, j)$ 에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- $D[i][j] = \max(D[i+1][j], D[i][j+1]) + A[i][j]$
- 정답은  $D[1][1]$ 에 있다.
- 즉,  $go(1, 1)$ 을 호출해서 답을 구해야 한다.

# 이동하기

<https://www.acmicpc.net/problem/11048>

- 소스: <http://codeplus.codes/c7e44113023a4c9da7f14ddda9d437d5>

# 문제 풀기

---

# 점프

<https://www.acmicpc.net/problem/1890>

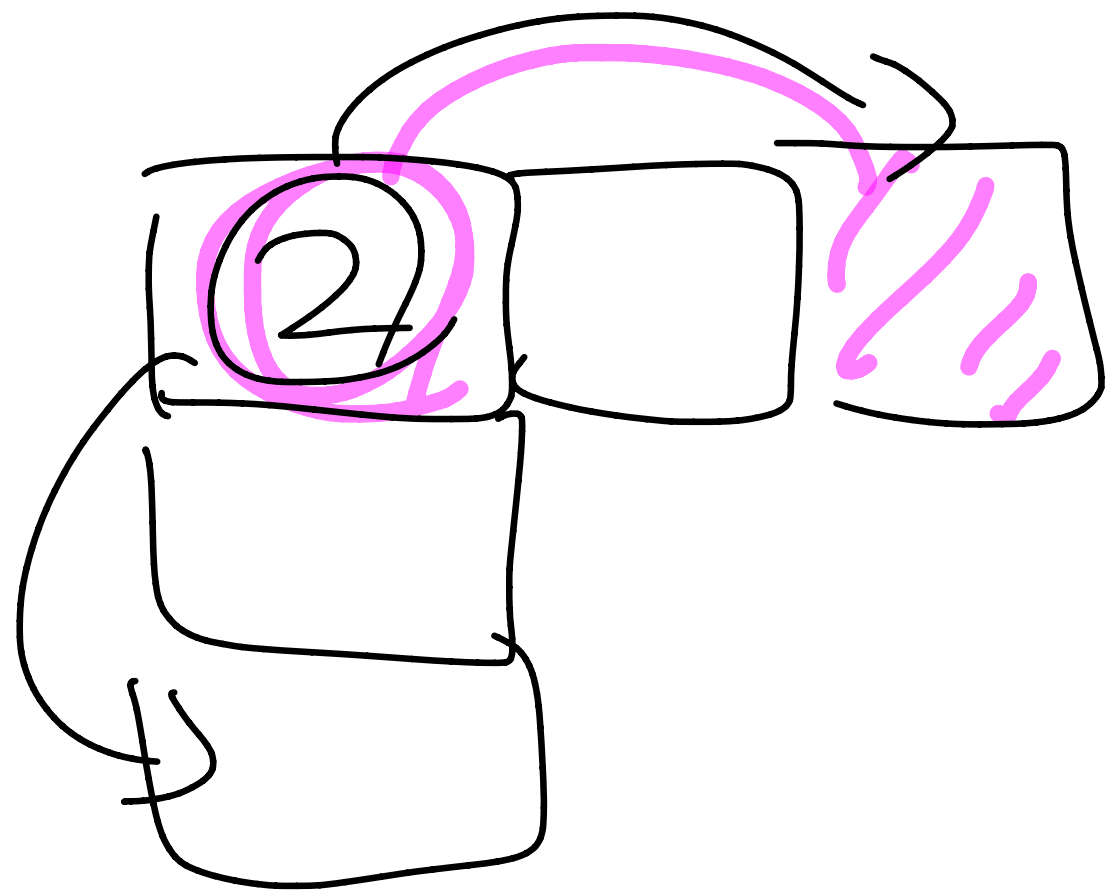
③

- $N \times N$  게임판에 수가 적혀져 있음
- 게임의 목표는 가장 왼쪽 위 칸에서 가장 오른쪽 아래 칸으로 규칙에 맞게 점프를 해서 가는 것
- 각 칸에 적혀있는 수는 현재 칸에서 갈 수 있는 거리를 의미
- 반드시 오른쪽이나 아래쪽으로만 이동해야 함
- 0은 더 이상 진행을 막는 종착점이며, 항상 현재 칸에 적혀있는 수만큼 오른쪽이나 아래로 가야 함
- 가장 왼쪽 위 칸에서 가장 오른쪽 아래 칸으로 규칙에 맞게 이동할 수 있는 경로의 개수를 구하는 문제

# 점프

<https://www.acmicpc.net/problem/1890>

- $D[i][j] = (i, j)$  칸에 갈 수 있는 경로의 개수
- $(i, j)$  칸에 올 수 있는 칸을 찾아야 한다.



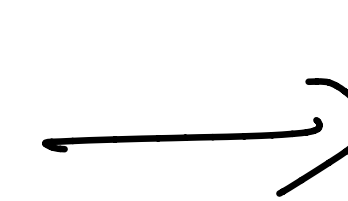
$$\bar{T} - k == A[\bar{k}][\bar{j}]$$

$$1 \leq k < \bar{T}$$

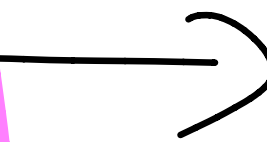


47

$$(\bar{T}, k)$$



$$(\bar{T}, j)$$



$$1 \leq k < \bar{j}$$

$$\bar{j} - k == \underline{A[\bar{k}][k]}$$

$$N^2 \times O(N)$$

$$O(N^3)$$

# 점프

<https://www.acmicpc.net/problem/1890>

- $D[i][j]$  = (i, j)칸에 갈 수 있는 경로의 개수
- (i, j)칸에 올 수 있는 칸을 찾아야 한다.
- $D[i][j] += D[i][k]$  ( $k+A[i][k] == j, 0 \leq k < j$ )
- $D[i][j] += D[k][j]$  ( $k+A[k][j] == i, 0 \leq k < i$ )



# 점프

<https://www.acmicpc.net/problem/1890>

- $D[i][j]$  = (i, j)칸에 갈 수 있는 경로의 개수
- (i, j)칸에 올 수 있는 칸을 찾아야 한다.
- $D[i][j] += D[i][k] \text{ (} k+A[i][k] == j, 0 \leq k < j \text{)}$
- $D[i][j] += D[k][j] \text{ (} k+A[k][j] == i, 0 \leq k < i \text{)}$
- 한 칸을 채우는데 필요한 복잡도:  $O(N)$
- 총 시간 복잡도 :  $O(N^3)$

# 점프

50

<https://www.acmicpc.net/problem/1890>

- 소스: <http://codeplus.codes/4e01dbf013b148c5b7d7d0fd42913bb5>

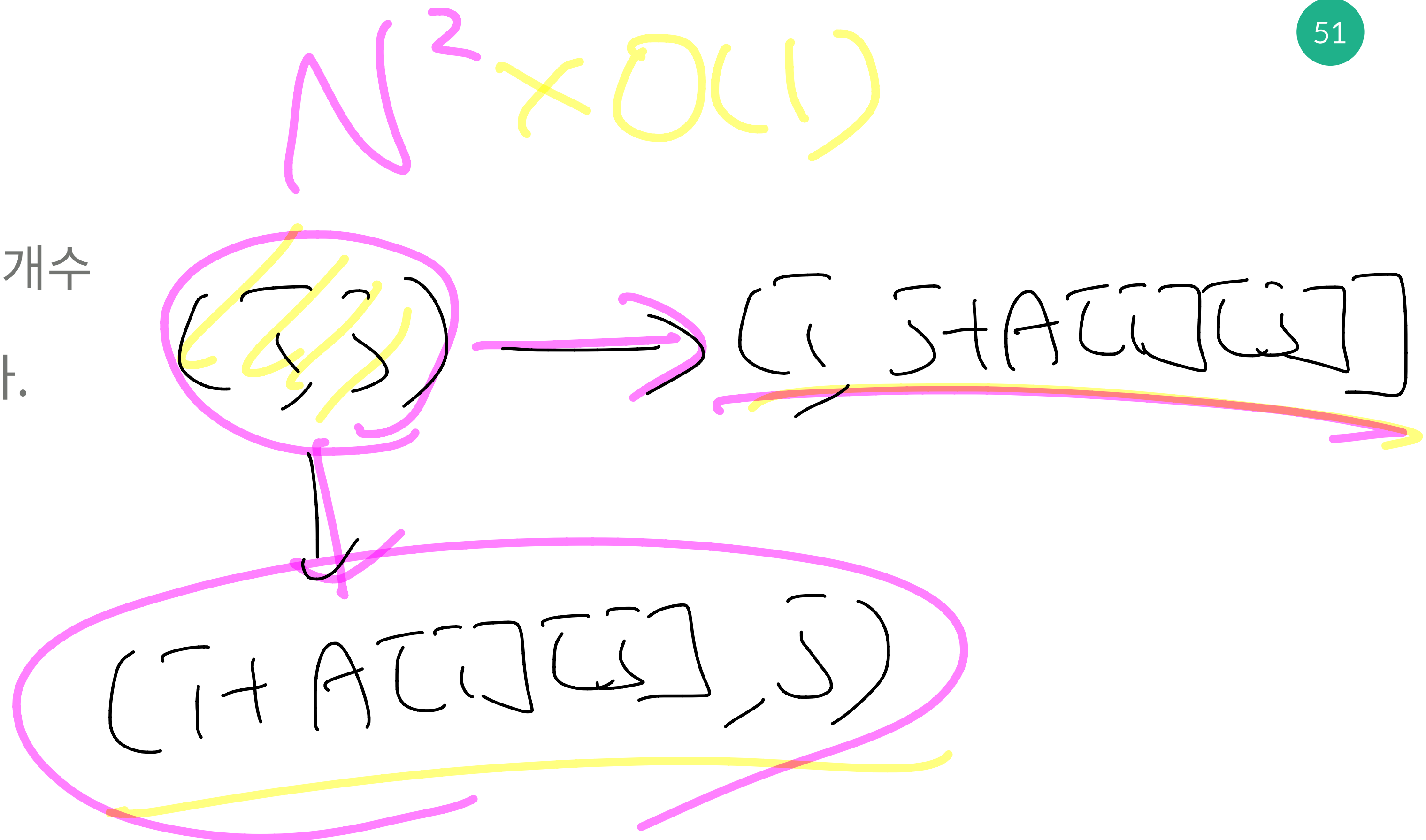
# 점프

51

<https://www.acmicpc.net/problem/1890>

- $D[i][j]$  = (i, j)칸에 갈 수 있는 경로의 개수
- (i, j)에서 갈 수 있는 칸을 찾아야 한다.
- $D[i][j+A[i][j]] += D[i][j];$
- $D[i+A[i][j]][j] += D[i][j];$

$O(N^2)$



# 점프

<https://www.acmicpc.net/problem/1890>

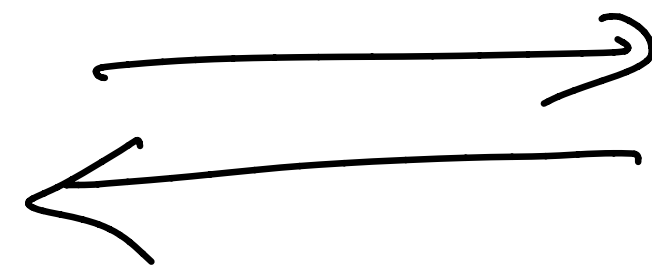
- $D[i][j] = (i, j)$ 칸에 갈 수 있는 경로의 개수
- $(i, j)$ 에서 갈 수 있는 칸을 찾아야 한다.
- $D[i][j+A[i][j]] += D[i][j];$
- $D[i+A[i][j]][j] += D[i][j];$
- 한 칸을 채우는데 필요한 복잡도:  $O(1)$
- 총 시간 복잡도 :  $O(N^2)$

# 점프

<https://www.acmicpc.net/problem/1890>

- 소스: <http://codeplus.codes/d9074b533f8c4874900cfff6c728161a>

포켓 링크



길이:  $N$

→ level

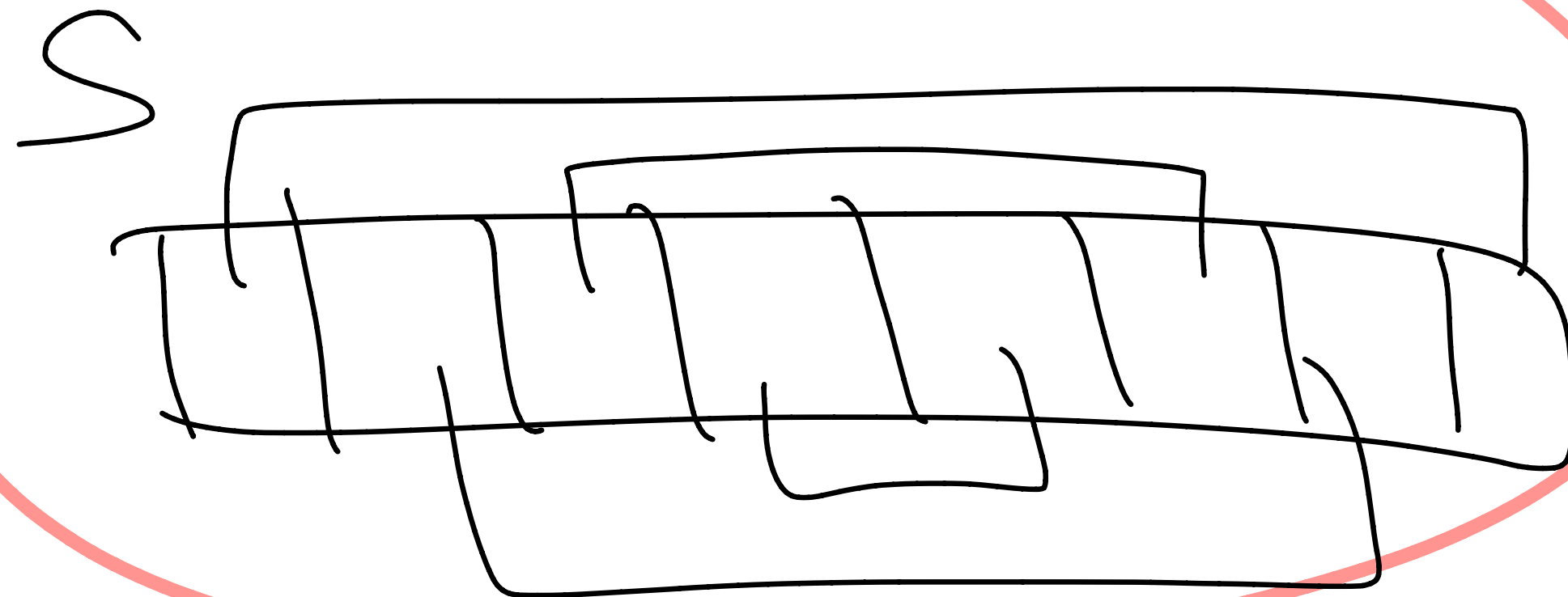
→ radar

① 어떤 문자열  $S$

$S = S_1 S_2 \dots S_N$  (같은 문자열)

$O(N)$

②



$O(N)$

# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

$$DP[i][j] = A[i] \sim A[j]$$

패턴, 1  
아니면, 0

54

- 어떤 수열의 부분 수열이 팰린드롬인지 확인하는 문제
- 팰린드롬인지 확인하는데 걸리는 시간:  $O(N)$
- 질문이  $M$ 개면  $O(MN)$ 이라는 시간이 걸림
- $1 \leq M \leq 1,000,000, 1 \leq N \leq 2,000$

1 2 1 3 1 2 1

$$A[i] = A[i+1]$$

aa  
bb

- 길이 1인 문자열  
 $DP[i][i] = \text{true}$
- 길이 2인 문자열  
 $DP[i][i+1] = \text{T/F}$

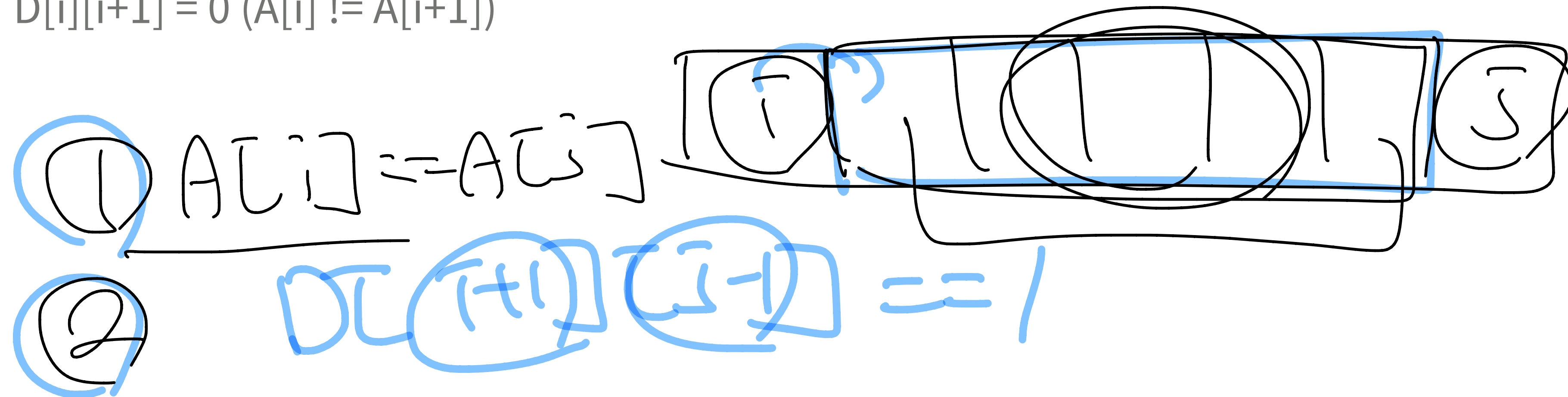
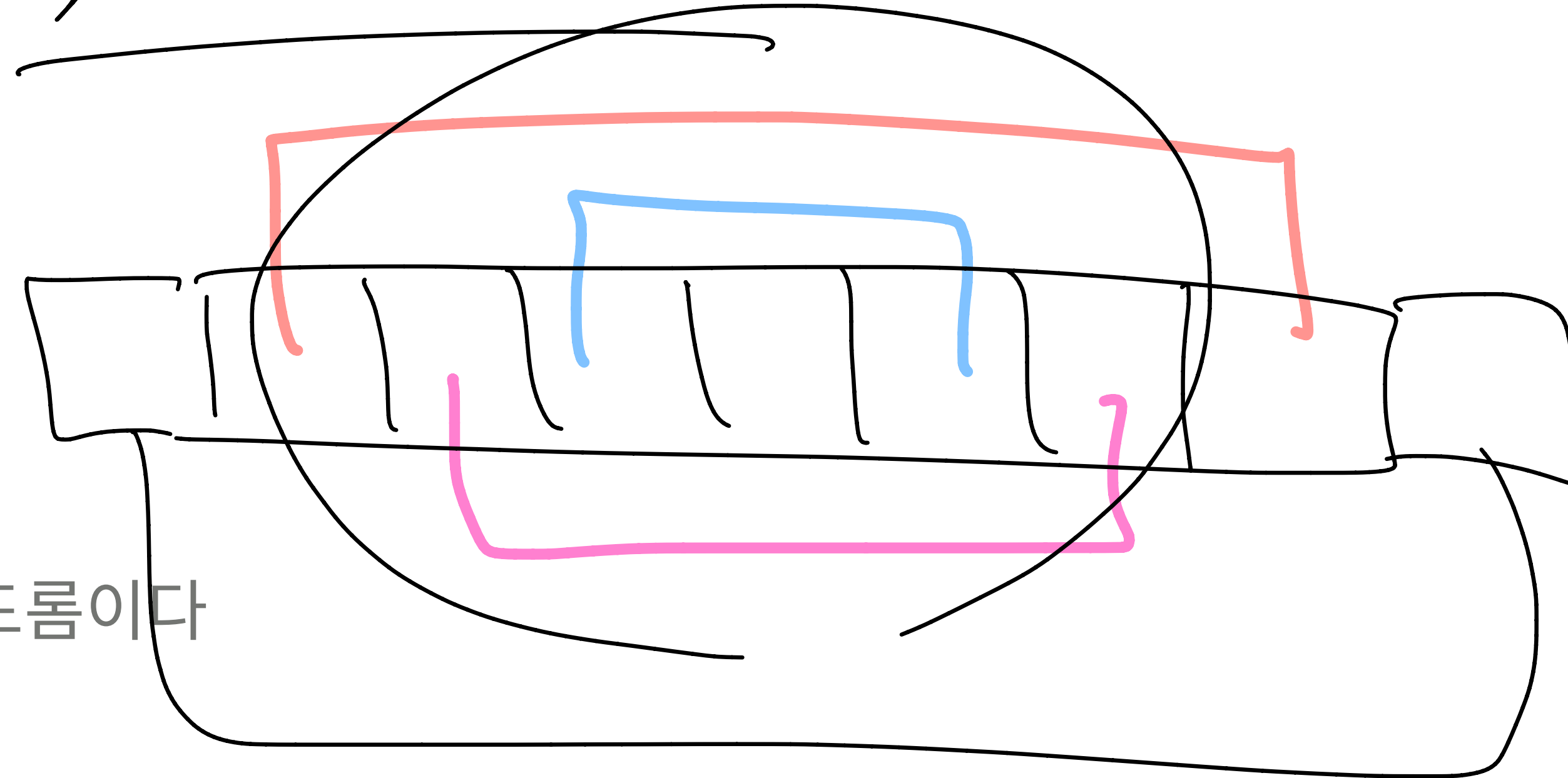
# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

55

- $D[i][j] = 1$  if  $A[i] \sim A[j]$  is a palindrome, 0 otherwise
- Length 1 substring is always a palindrome
  - $D[i][i] = 1$
- Length 2 substring is a palindrome only if the two numbers are equal
  - $D[i][i+1] = 1$  ( $A[i] == A[i+1]$ )
  - $D[i][i+1] = 0$  ( $A[i] != A[i+1]$ )

길이가 3이상



# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

- $D[i][j] = 1$  if  $A[i] \sim A[j]$  is a palindrome, otherwise 0
- 길이가 1인 부분 수열은 반드시 팰린드롬이다
  - $D[i][i] = 1$
- 길이가 2인 부분 수열은 두 수가 같을 때만 팰린드롬이다
  - $D[i][i+1] = 1$  ( $A[i] == A[i+1]$ )
  - $D[i][i+1] = 0$  ( $A[i] != A[i+1]$ )
- $A[i] \sim A[j]$ 가 팰린드롬이 되려면,  $A[i] == A[j]$  이어야 하고,  $A[i+1] \sim A[j-1]$ 이 팰린드롬이어야 한다
  - $D[i][j] = 1$  ( $A[i] == A[j] \ \&\& \ D[i+1][j-1] == 1$ )



# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

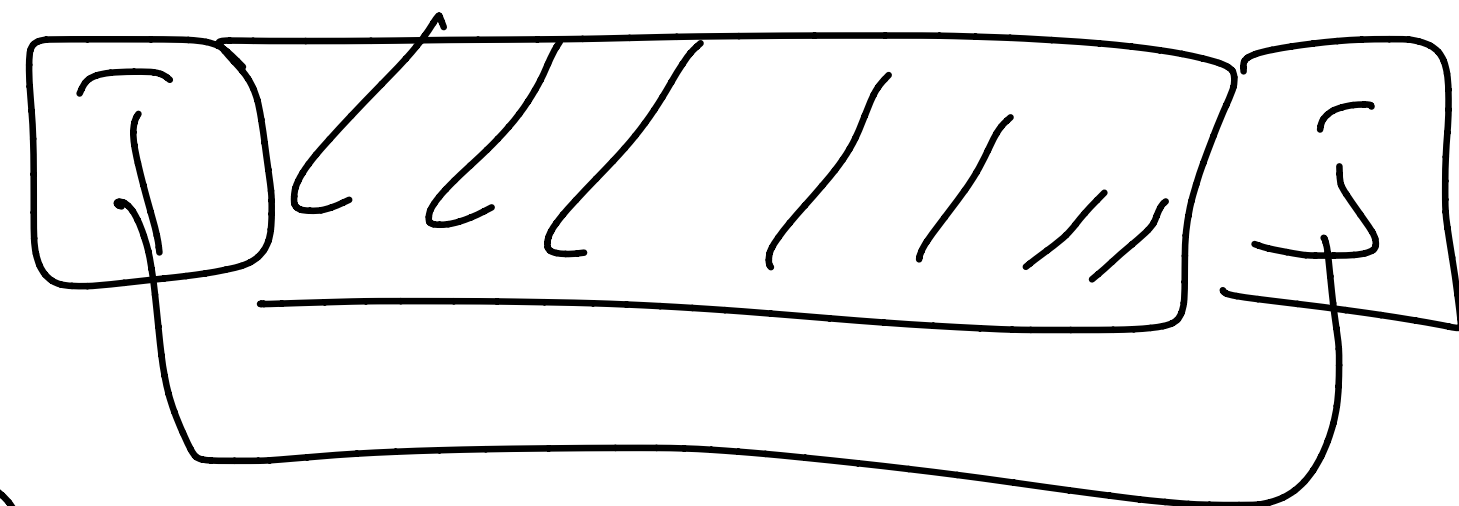
- 일반적인 방식으로 배열을 채우지 않기 때문에 재귀 호출을 사용하는 것이 좋다

# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

```
int go(int i, int j) {  
    if (i == j) {  
        return 1;  
    } else if (i+1 == j) {  
        if (a[i] == a[j]) return 1;  
        else return 0;  
    }  
    if (d[i][j] >= 0) return d[i][j];  
    if (a[i] != a[j]) return d[i][j] = 0;  
    else return d[i][j] = go(i+1, j-1);  
}
```

-1 0, 2  
0, X  
1, 0



# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

- 재귀 호출을 사용하지 않고도 풀 수 있다
- 길이가 1인  $D[i][j]$ 를 채우고
- 2인 것을 채우고
- 3인 것을 채우고
- ...
- $N-1$ 인 것을 채우는 방식을 이용하면
- for문으로도 채울 수 있다

# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

```
for (int i=1; i<=n; i++) d[i][i] = true;
for (int i=1; i<=n-1; i++) {
    if (a[i] == a[i+1]) d[i][i+1] = true;
}
for (int k=3; k<=n; k++) {
    for (int i=1; i<=n-k+1; i++) {
        int j = i+k-1;
        if (a[i] == a[j] && d[i+1][j-1]) {
            d[i][j] = true;
        }
    }
}
}
```

# 팰린드롬?

<https://www.acmicpc.net/problem/10942>

- Bottom-up 소스: <http://codeplus.codes/e9e2a159977845a7936698d2a7e26001>
- Top-down 소스: <http://codeplus.codes/bc785b9f9be54bc0ab44cd9d41c30d3b>

# 1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

$$D[n] = n\text{을 나타내는 방법의 수}$$
$$= D[n-1] + D[n-2] + D[n-3]$$

62

- 정수  $n$ 을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 문제

- $n = 4$

- 1+1+1+1

- 1+1+2

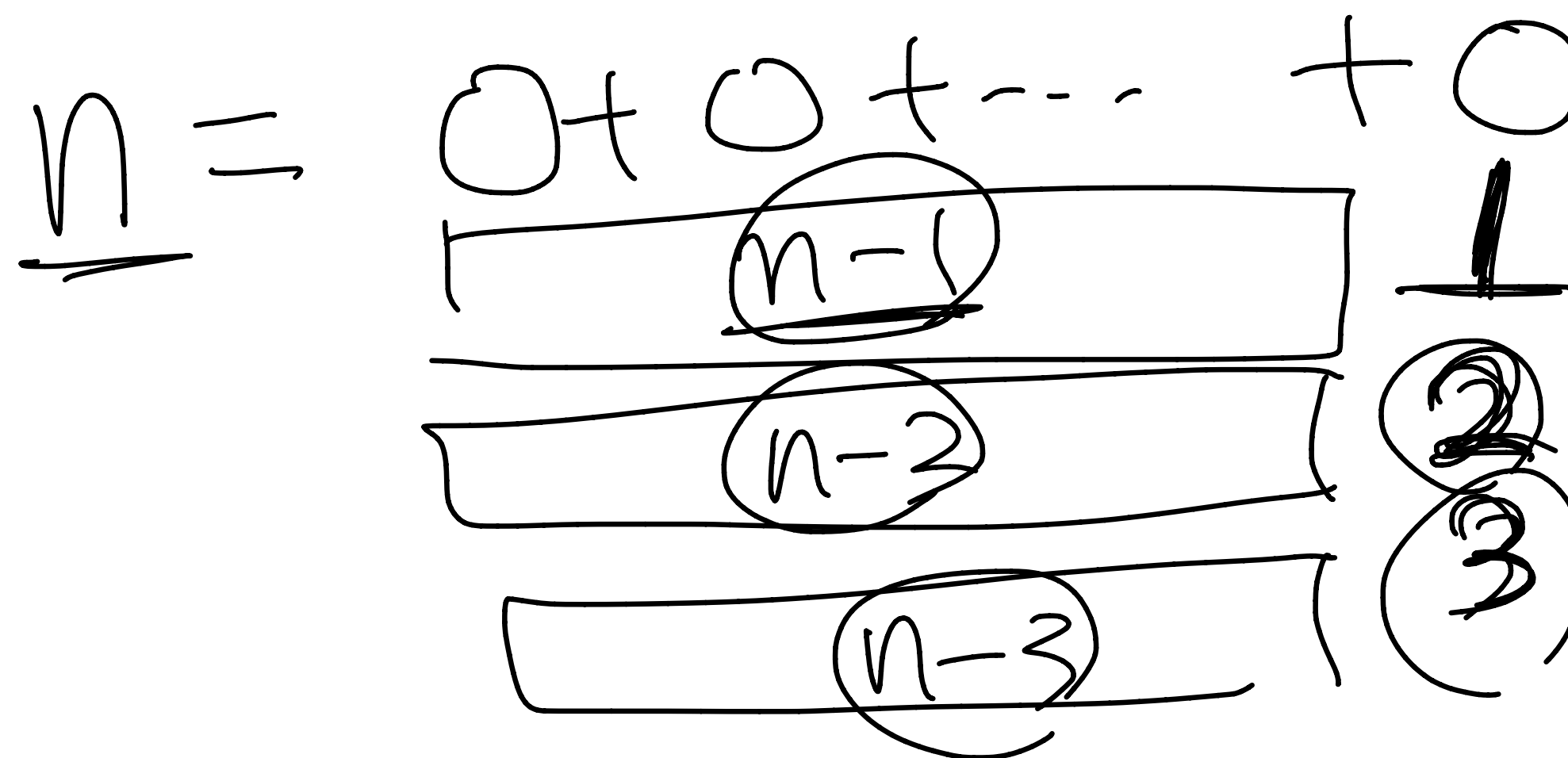
- 1+2+1

- 2+1+1

- 2+2

- 1+3

- 3+1



# 1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

- $D[i]$  =  $i$ 를 1, 2, 3의 조합으로 나타내는 방법의 수
- $D[i] = D[i-1] + D[i-2] + D[i-3]$

# 1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

64

0

1

2

3

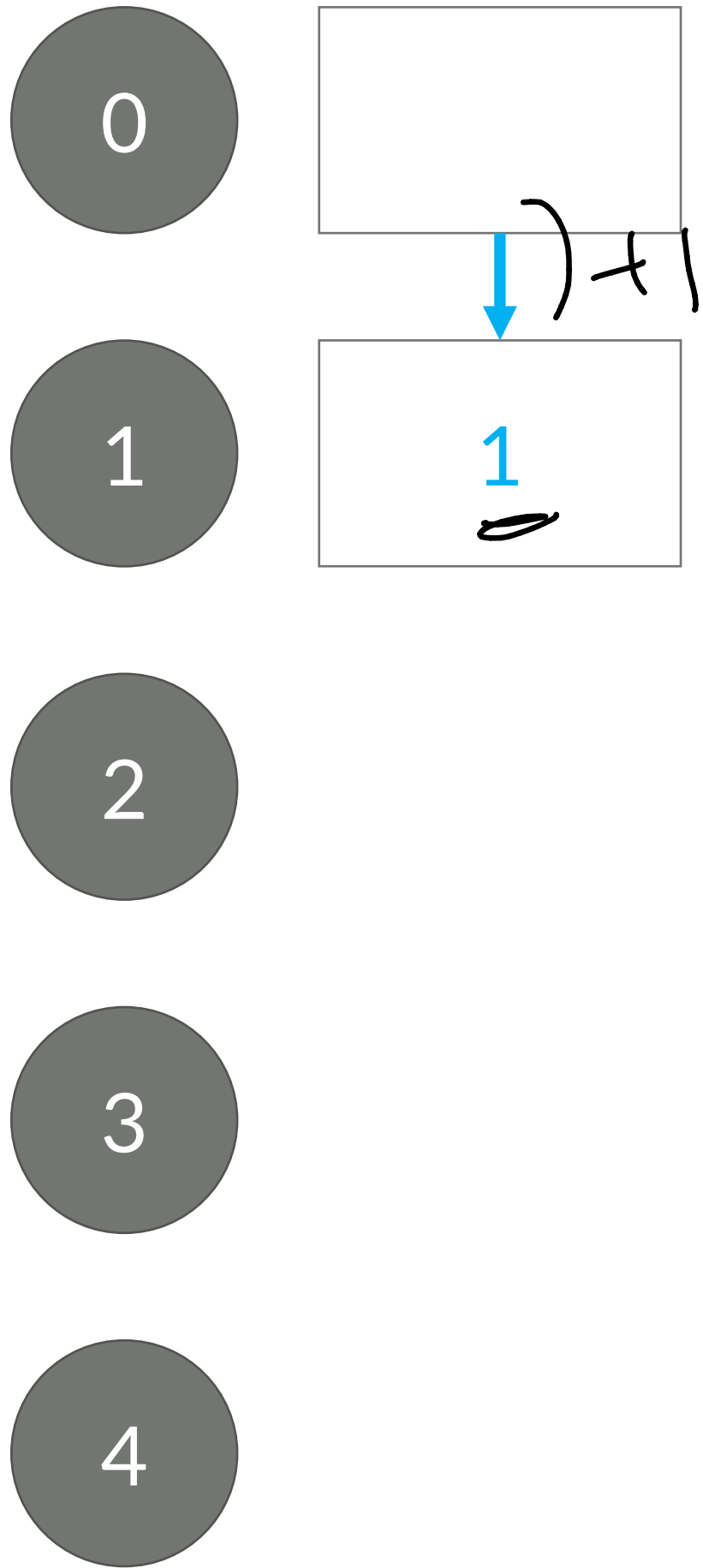
4



# 1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

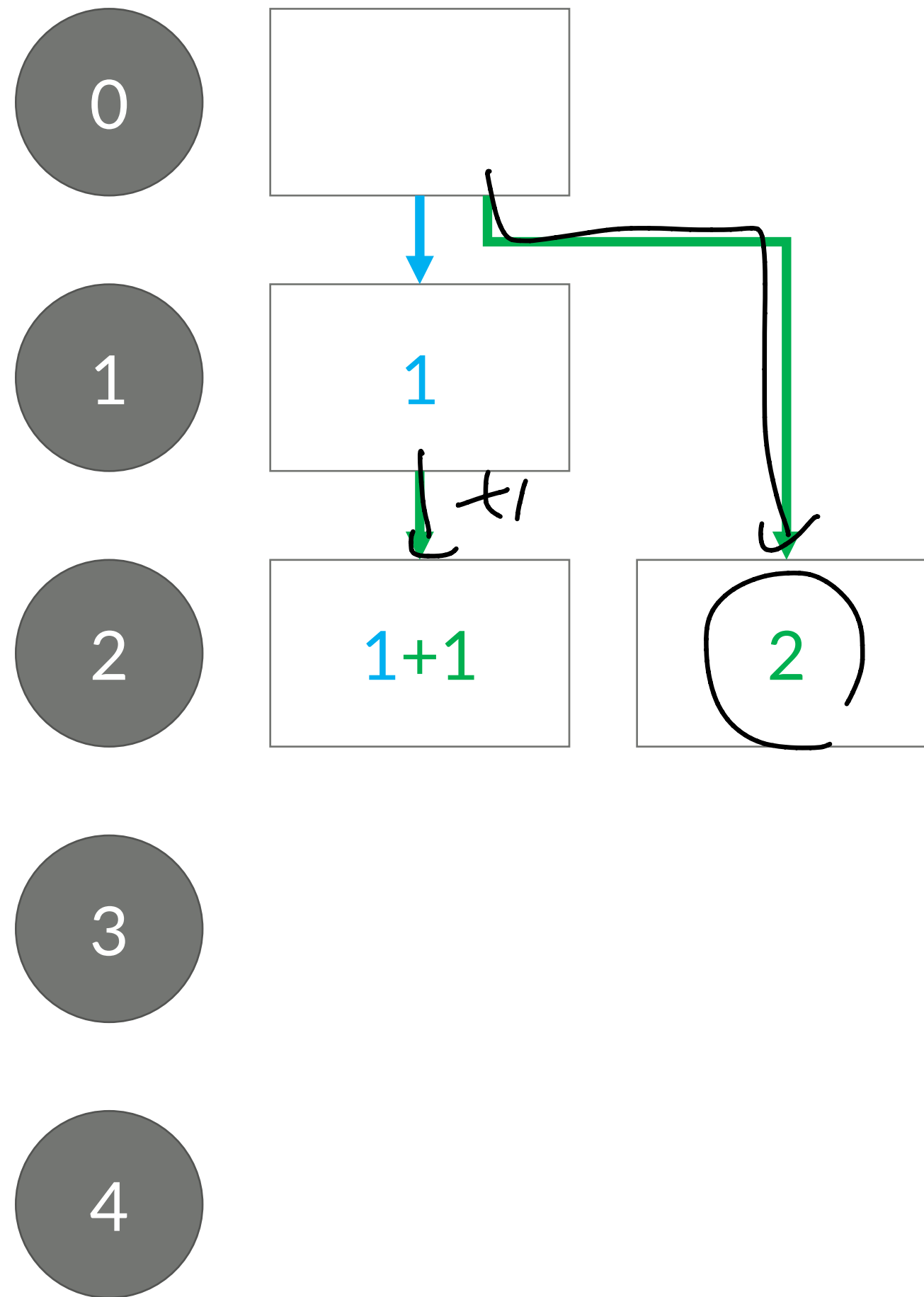
65



# 1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

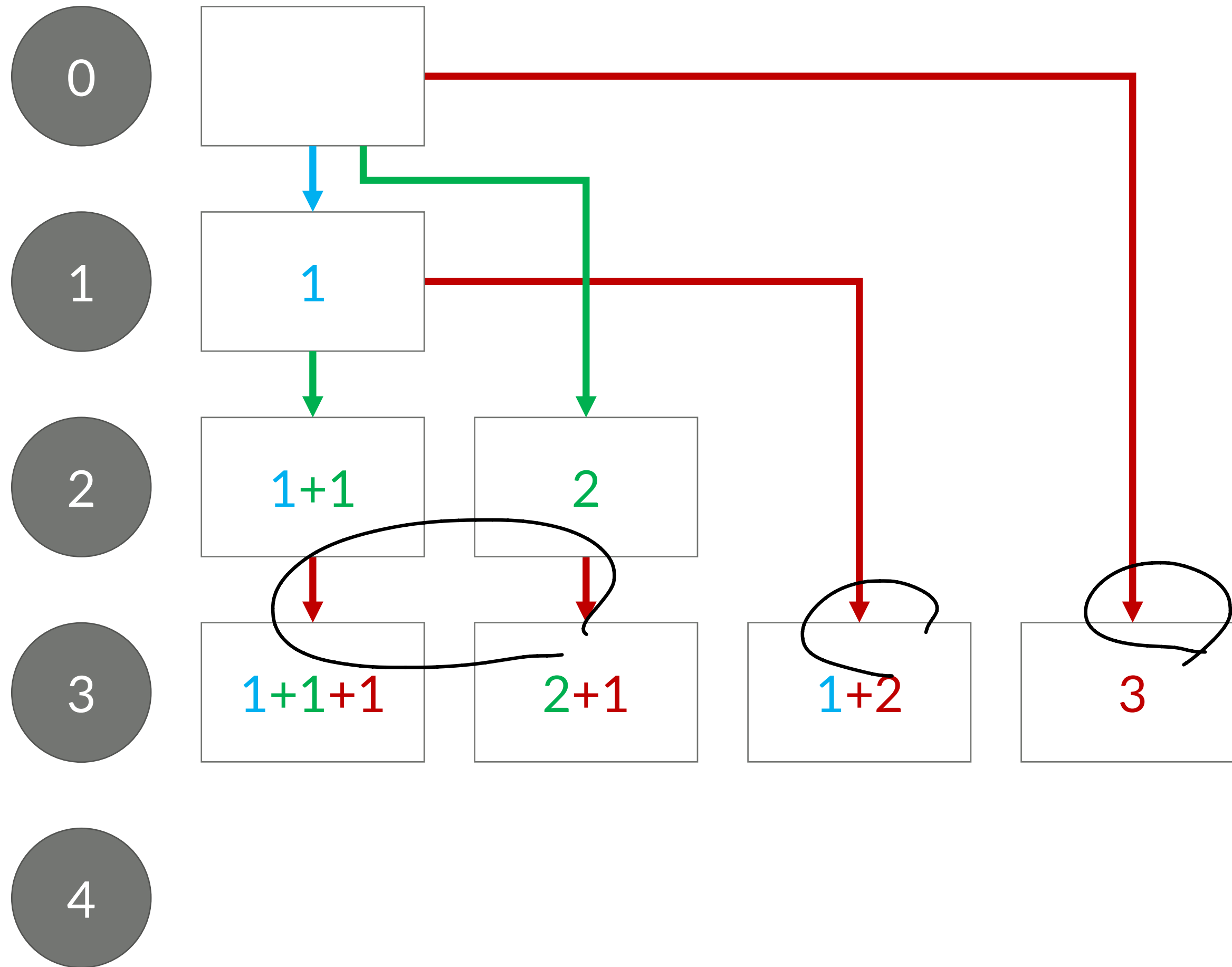
66



# 1, 2, 3 더하기

<https://www.acmicpc.net/problem/9095>

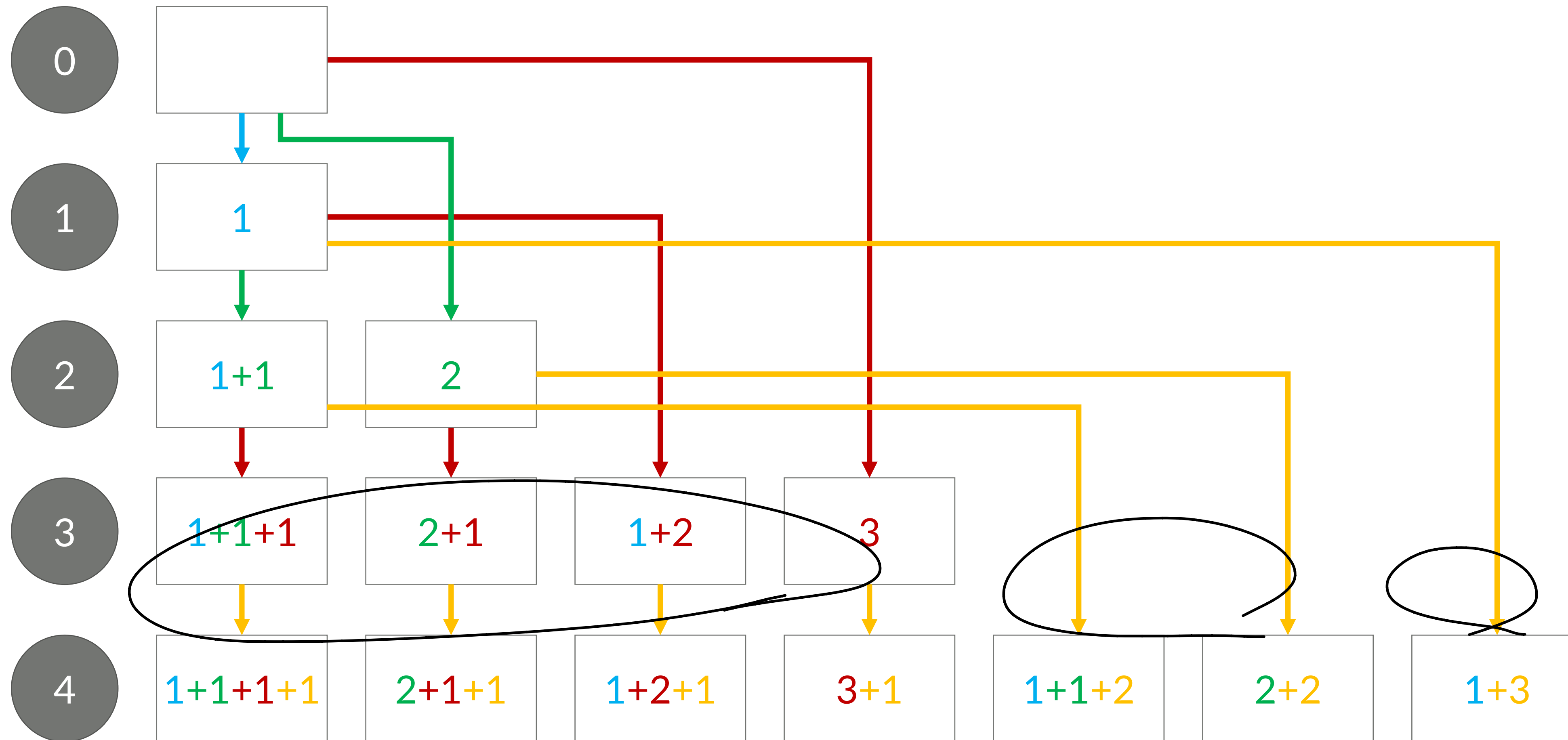
67



# 1, 2, 3 더하기

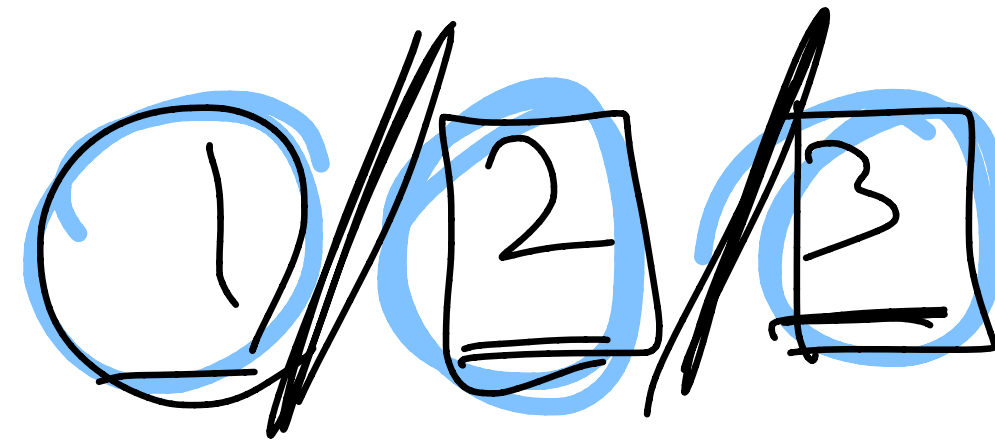
<https://www.acmicpc.net/problem/9095>

68



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>



69

- 정수  $n$ 을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 문제
- 합을 이루고 있는 수의 순서만 다른 것은 같은 것으로 친다.

•  $n = 4$

•  $1+1+1+1$

•  $2+1+1$  ( $1+1+2$ ,  $1+2+1$ )

•  $2+2$

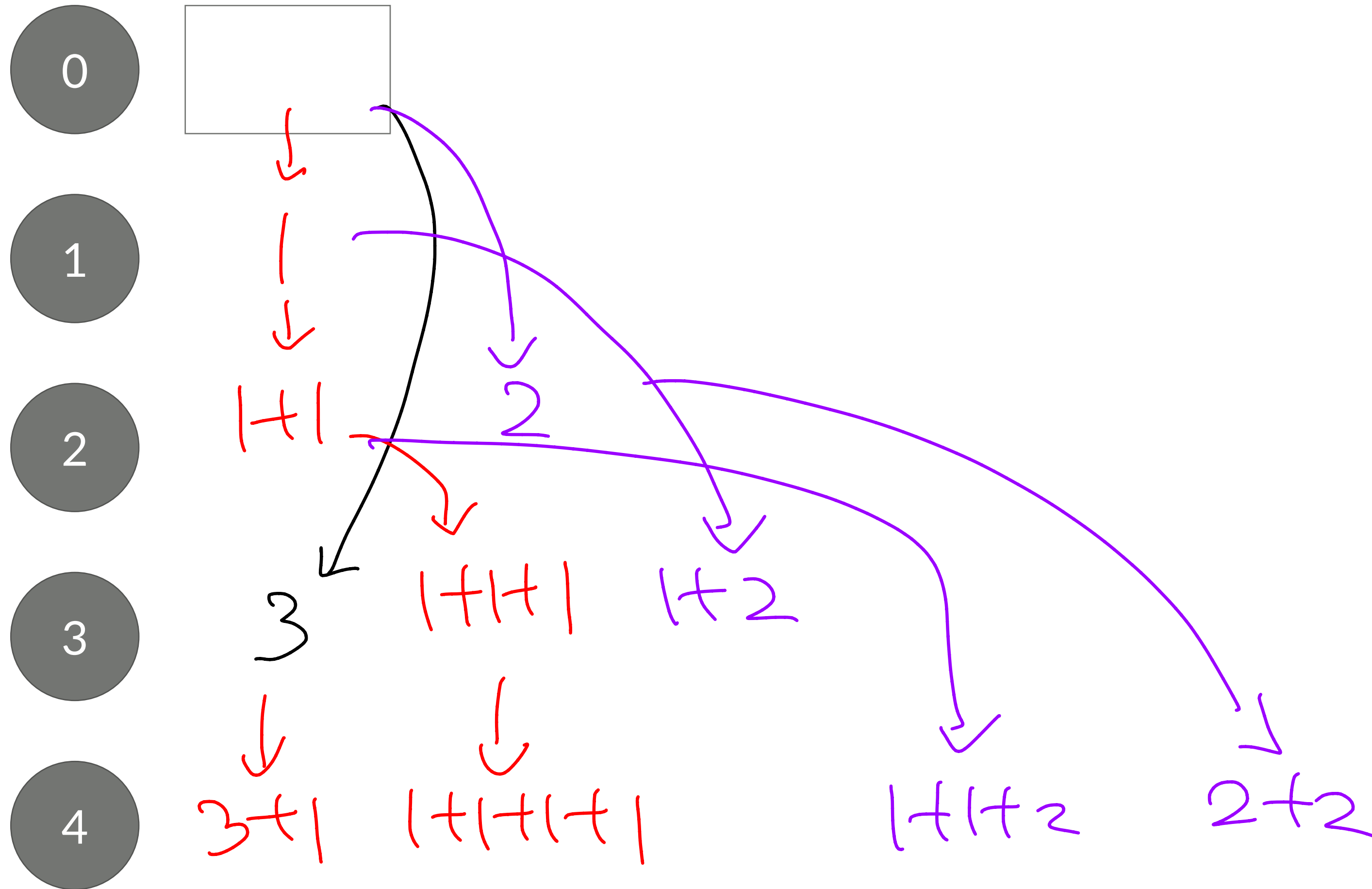
•  $1+3$  ( $3+1$ )

# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

70

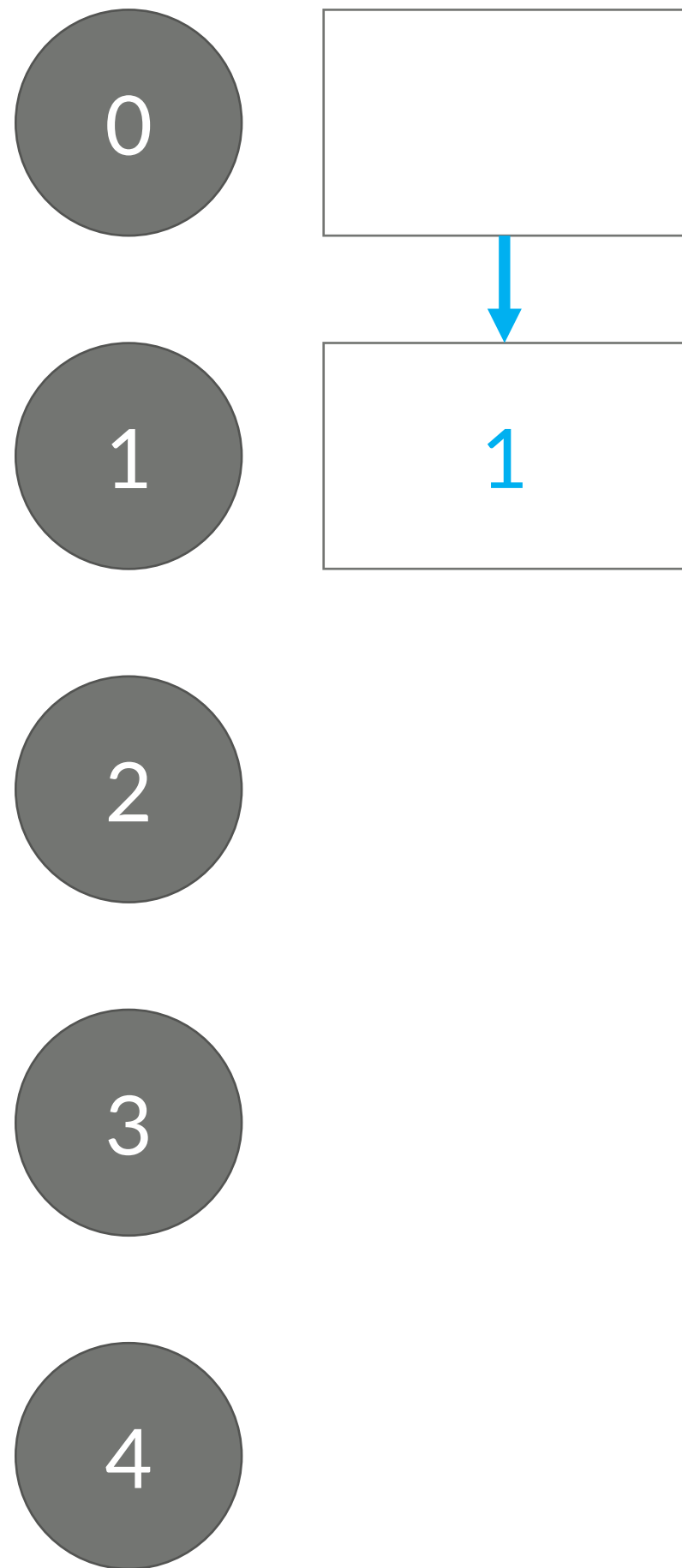
3, 1, 2



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

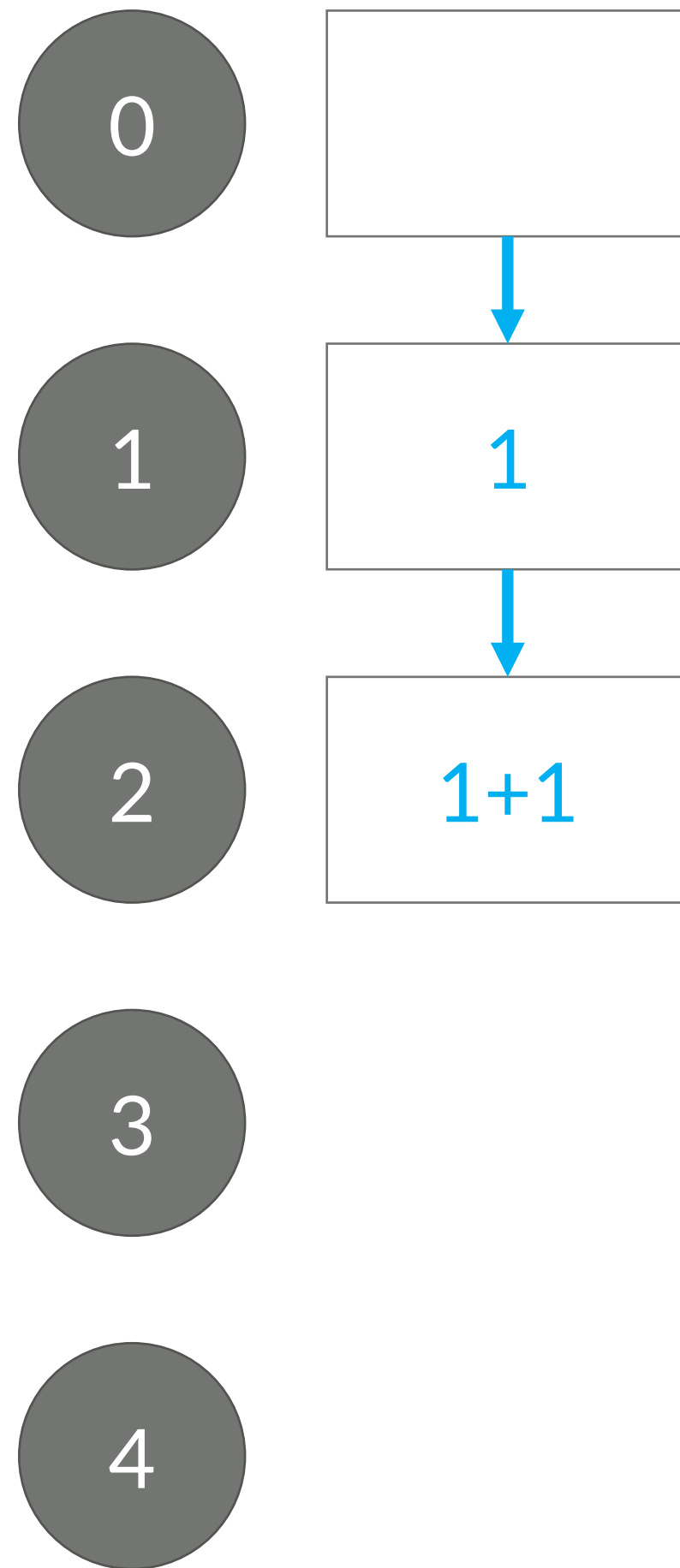
71



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

72

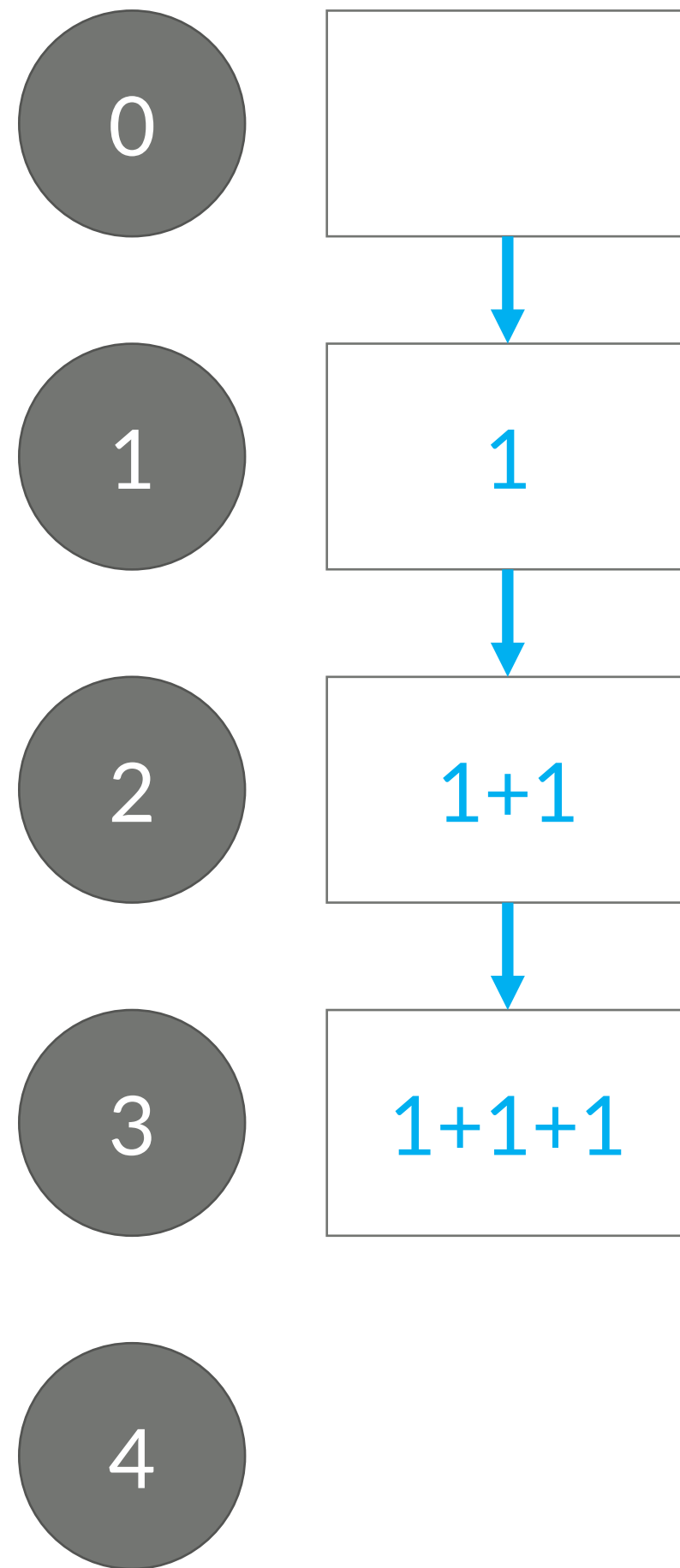




# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

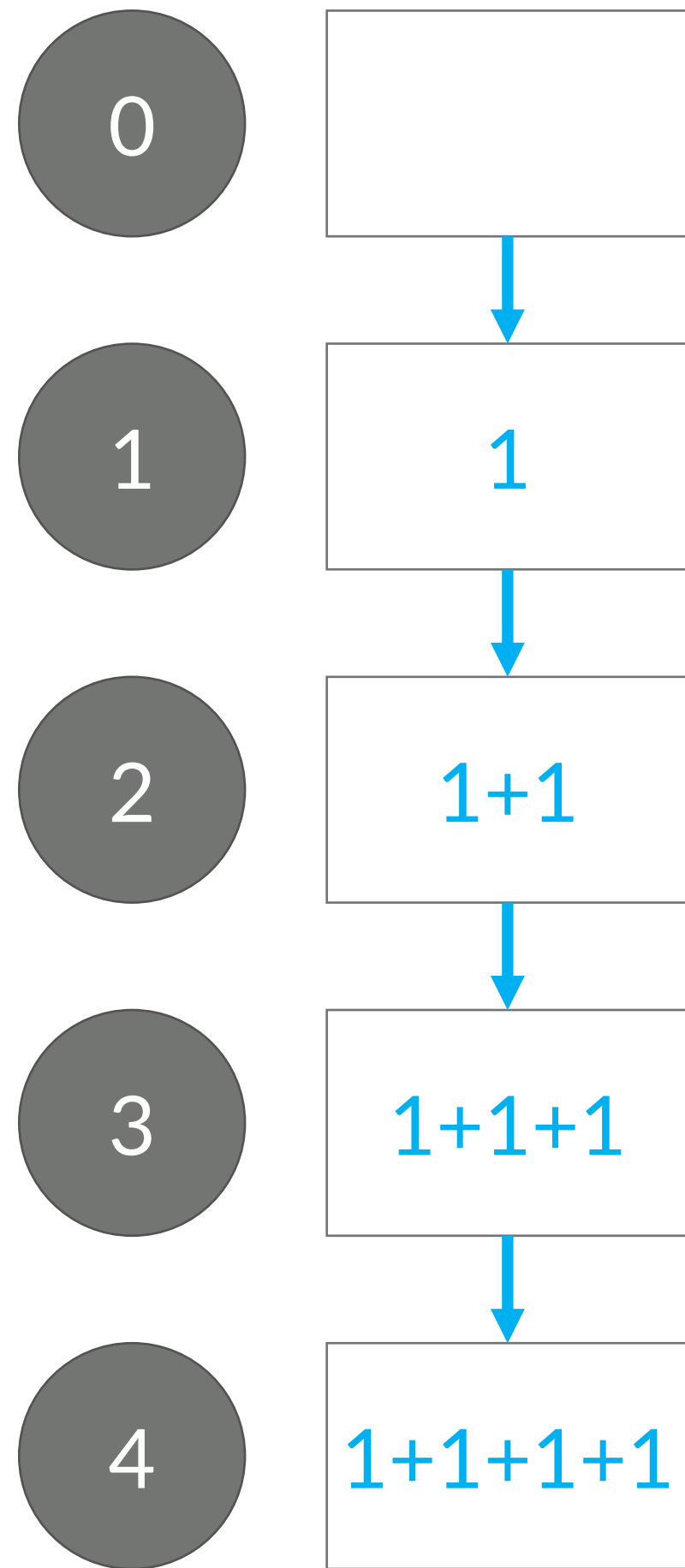
73



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

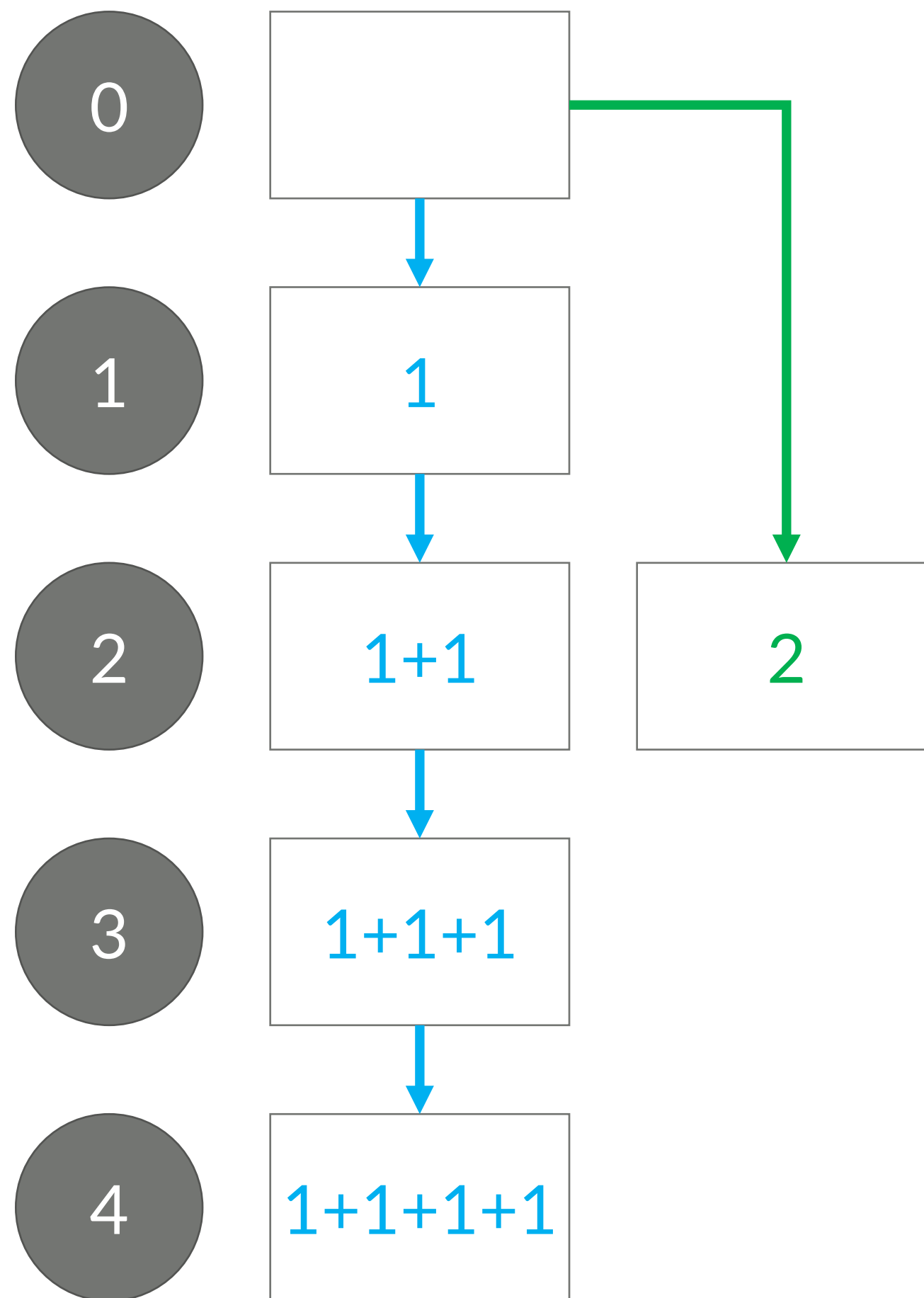
74



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

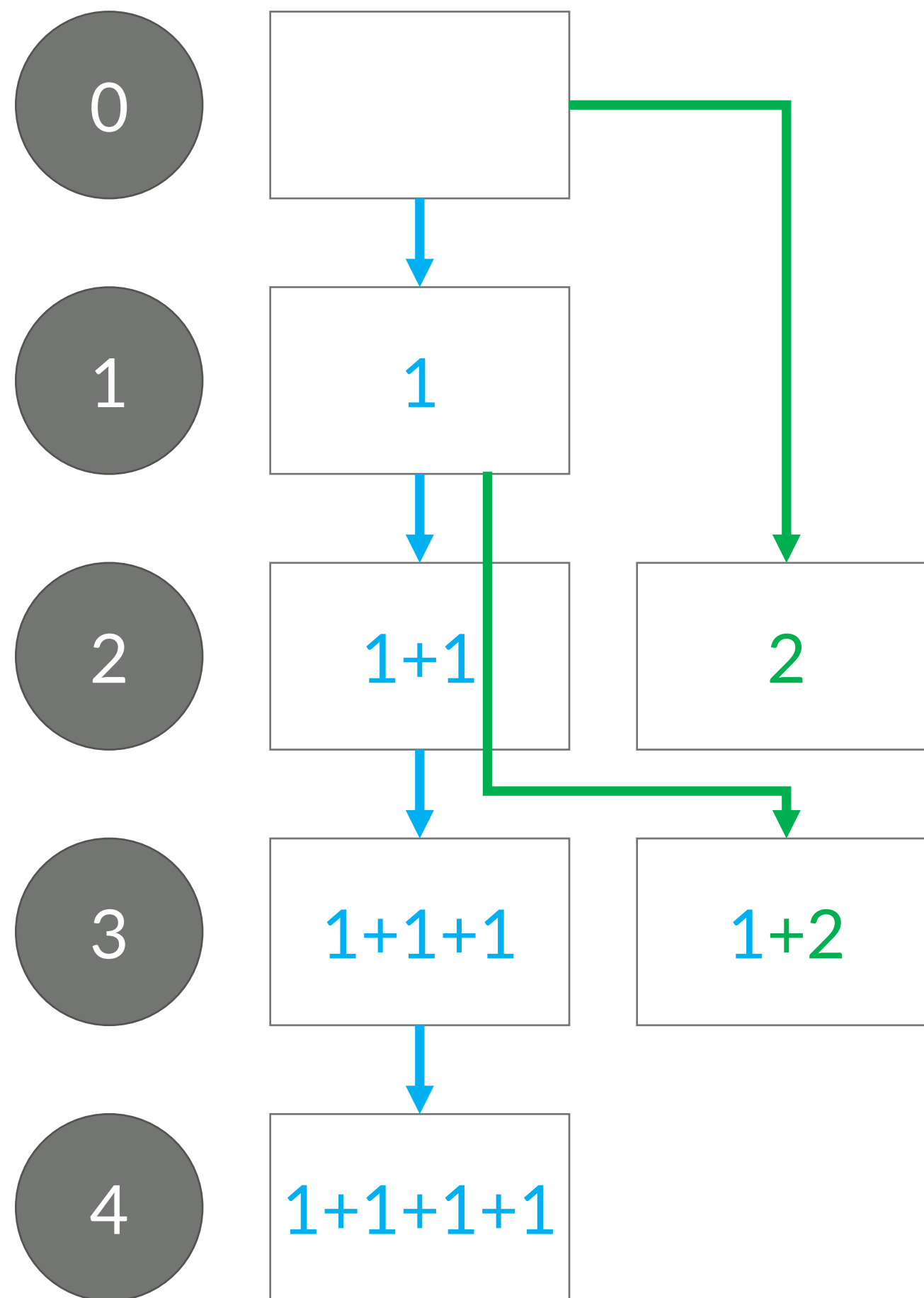
75



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

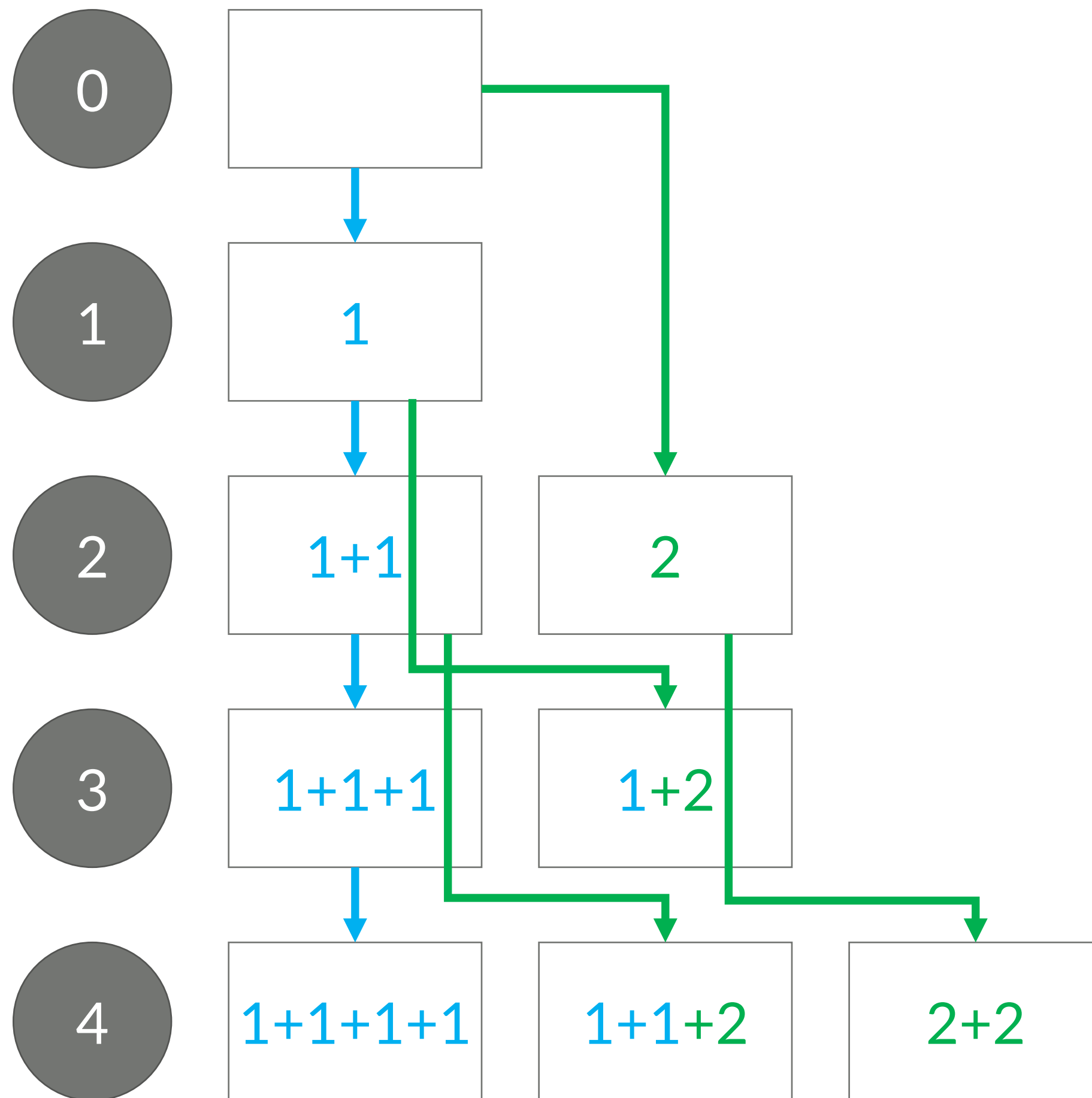
76



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

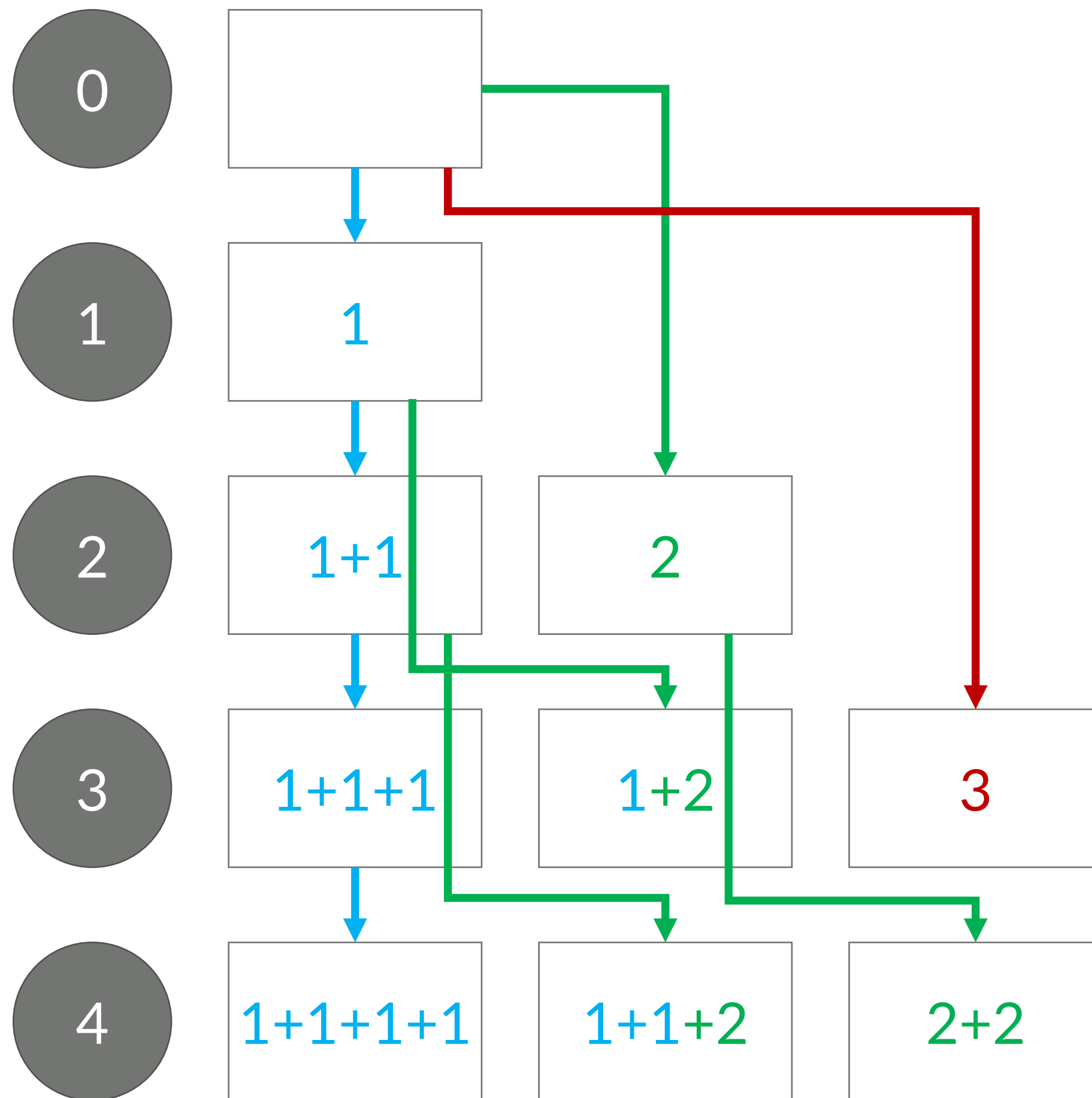
77



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

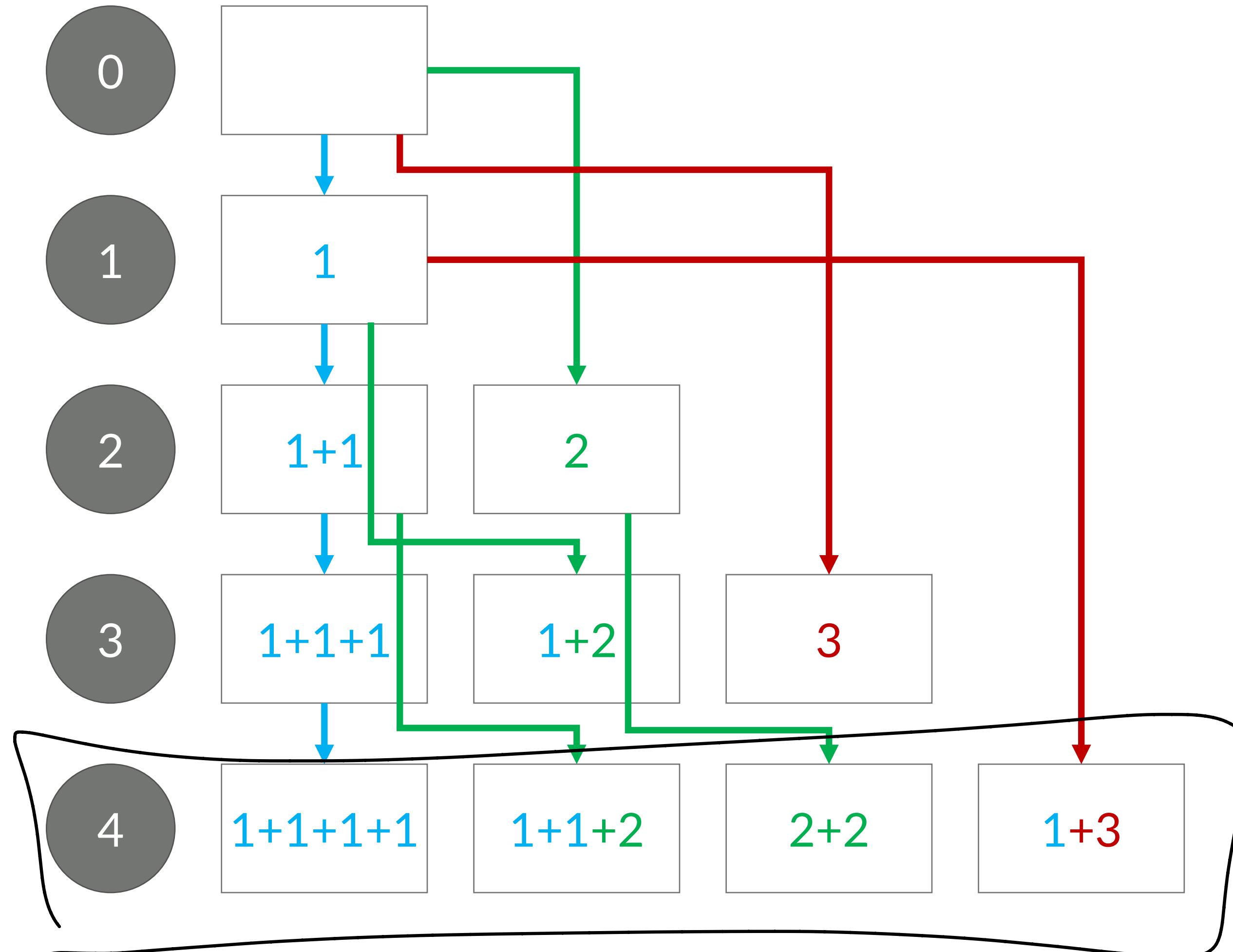
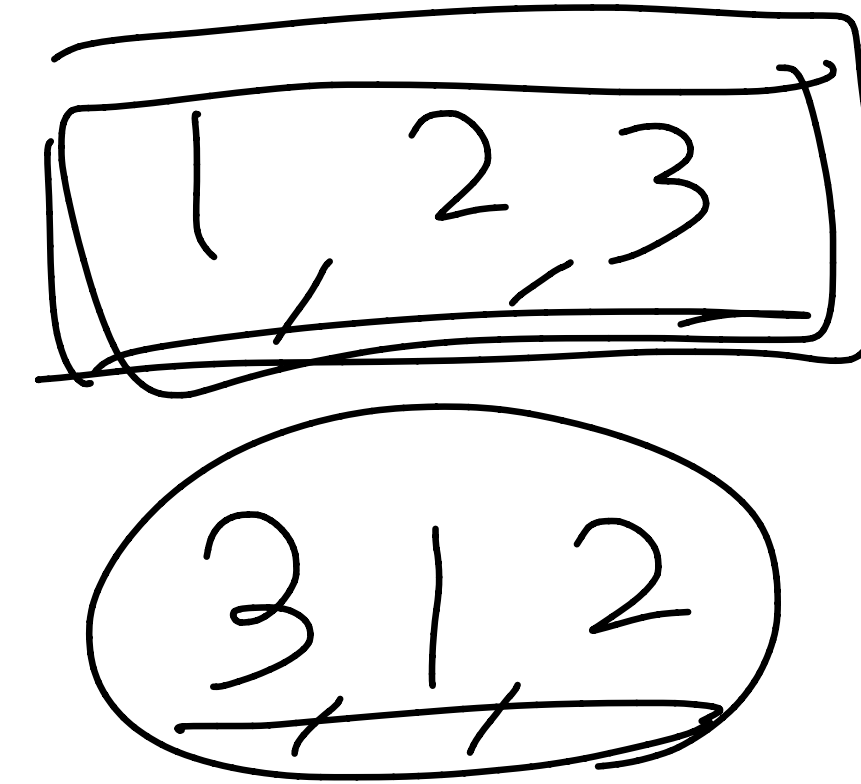
78



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

79



# 1, 2, 3 더하기

80

<https://www.acmicpc.net/problem/9095>

```
int n = 4;
int m = 3;
int nums = {1, 2, 3};
int d[n];
d[0] = 1;
for (int i=1; i<=n; i++) {
    for (int j=0; j<m; j++) {
        if (i-nums[j] >= 0) {
            d[i] += d[i-nums[j]];
        }
    }
}
```

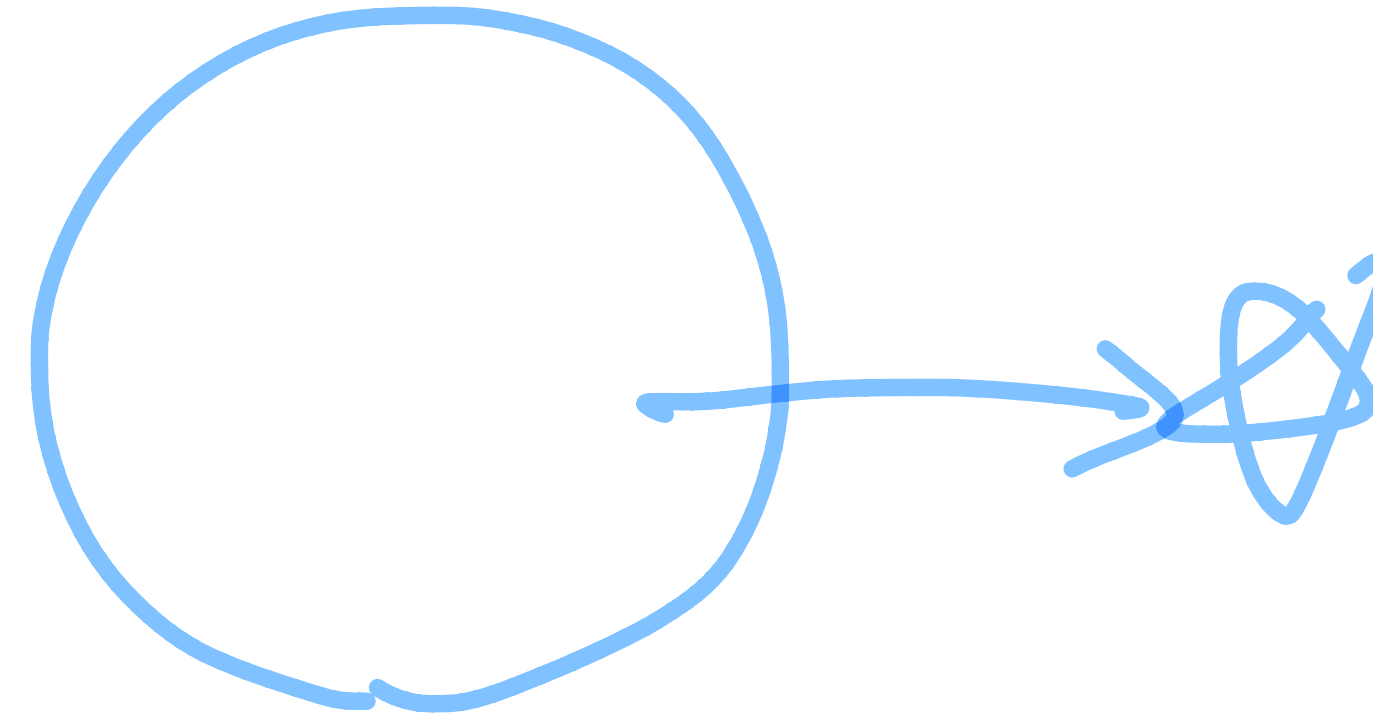
$$D[i] = D[i-1] + D[i-2] + D[i-3]$$



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

```
int n = 4;
int m = 3;
int nums = {1, 2, 3};
           2   1   3
int d[n];
d[0] = 1;
for (int j=0; j<m; j++) {
    for (int i=1; i<=n; i++) {
        if (i-nums[j] >= 0) {
            d[i] += d[i-nums[j]];
        }
    }
}
```



# 1, 2, 3 더하기 4

<https://www.acmicpc.net/problem/15989>

- 소스: <http://codeplus.codes/80f087c81027447c8dbfc4d79ad727fe>

# 동전 1

<https://www.acmicpc.net/problem/2293>

- n가지 종류의 동전이 있다
- 각각의 동전이 나타내는 가치는 다르다
- 이 동전들을 적당히 사용해서, 그 가치의 합이 k원이 되도록 하고 싶다
- 그 경우의 수를 구하시오.
- 각각의 동전은 몇 개라도 사용할 수 있다.

# 동전 1

<https://www.acmicpc.net/problem/2293>

- 1, 2, 3 더하기 4와 같은 문제이다

# 동전 1

85

<https://www.acmicpc.net/problem/2293>

- 소스: <http://codeplus.codes/3de6244b1f6841b285a645fa5243fa81>

# 동전 2

<https://www.acmicpc.net/problem/2294>

- $n$ 가지 종류의 동전이 있다
- 각각의 동전이 나타내는 가치는 다르다
- 이 동전들을 적당히 사용해서, 그 가치의 합이  $k$ 원이 되도록 하고 싶다
- 그러면서 동전의 개수가 최소가 되도록 하려고 한다
- 각각의 동전은 몇개라도 사용할 수 있다

# 동전 2

<https://www.acmicpc.net/problem/2294>

- 동전 1과 비슷한 방법으로 풀 수 있다
- $D[i]$  = i원을 만드는데 필요한 동전의 최소 개수

⊕ min

# 동전 2

<https://www.acmicpc.net/problem/2294>

- 소스: <http://codeplus.codes/e50d496acfd34689a00f83c709be546d>



# 크리보드

<https://www.acmicpc.net/problem/11058>

$D[N]$  = 버퍼를 N번, 화면 A에 최대 개수

89

①  $D[N-1] + 1$

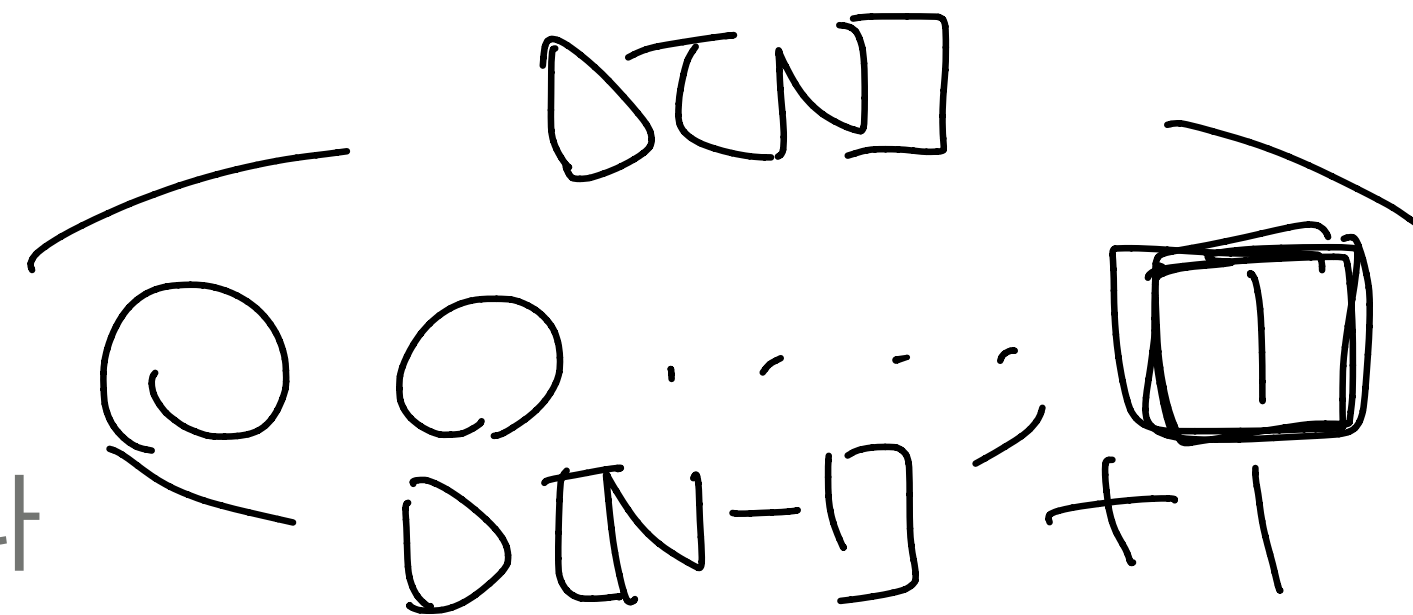
1. 화면에 A를 출력한다.

2. Ctrl-A: 화면을 전체 선택한다

3. Ctrl-C: 전체 선택한 내용을 버퍼에 복사한다

4. Ctrl-V: 버퍼가 비어있지 않은 경우에는 화면에 출력된 문자열의 바로 뒤에 버퍼의 내용을 붙여넣는다.

• 크리보드의 버튼을 총 N번 눌러서 화면에 출력된 A개수를 최대화하는 프로그램을 작성하시오.



Ctrl+A, Ctrl+C - Ctrl+V

# 크리보드

<https://www.acmicpc.net/problem/11058>

A@1>731 D[N]

N번

- Ctrl + A, Ctrl + C, Ctrl + V는 꼭 연속으로 눌러야 의미가 있다

? , ? , ?

N-3번

A의 최대 개수 :  $\boxed{D[N-3]} \times 2$

~~Ctrl + A, Ctrl + A~~

~~Ctrl + C, Ctrl + C~~

Ctrl + V, Ctrl + V

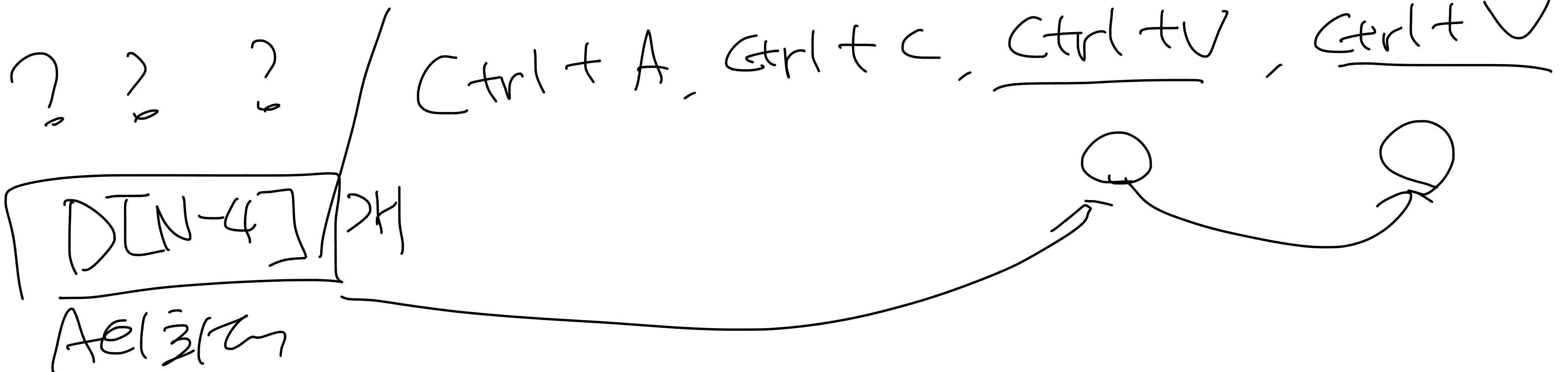
# 크리보드

<https://www.acmicpc.net/problem/11058>

$$D[N-4] \times 3$$

- $D[i]$  = 크리보드의 버튼을 총  $i$ 번 눌러서 화면에 출력된  $A$ 개수의 최대값
- 화면에  $A$ 를 출력하는 버튼을 누른 경우:
- 마지막에  $\text{Ctrl} + A$ ,  $\text{Ctrl} + C$ ,  $\text{Ctrl} + V$ 를 누른 경우:

누르



# 크리보드

<https://www.acmicpc.net/problem/11058>

- $D[i]$  = 크리보드의 버튼을 총  $i$ 번 눌러서 화면에 출력된  $A$ 개수의 최대값
- 화면에  $A$ 를 출력하는 버튼을 누른 경우:  $D[i-1] + 1$
- 마지막에  $\text{Ctrl} + A$ ,  $\text{Ctrl} + C$ ,  $\text{Ctrl} + V$ 를 누른 경우:  $D[i-3] * 2$

# 크리보드

<https://www.acmicpc.net/problem/11058>

- $D[i]$  = 크리보드의 버튼을 총  $i$ 번 눌러서 화면에 출력된  $A$ 개수의 최대값
- 화면에  $A$ 를 출력하는 버튼을 누른 경우:  $D[i-1] + 1$
- 마지막에 Ctrl + A, Ctrl + C, Ctrl + V를 누른 경우:  $D[i-3] * 2$
- 마지막에 Ctrl + A, Ctrl + C, Ctrl + V, Ctrl + V를 누른 경우:  $D[i-4] * 3$

# 크리보드

<https://www.acmicpc.net/problem/11058>

- $D[i]$  = 크리보드의 버튼을 총  $i$ 번 눌러서 화면에 출력된  $A$ 개수의 최대값
- 화면에  $A$ 를 출력하는 버튼을 누른 경우:  $D[i-1] + 1$
- 마지막에 Ctrl + A, Ctrl + C, Ctrl + V를 누른 경우:  $D[i-3] * 2$
- 마지막에 Ctrl + A, Ctrl + C, Ctrl + V, Ctrl + V를 누른 경우:  $D[i-4] * 3$
- 마지막에 Ctrl + A, Ctrl + C, Ctrl + V, Ctrl + V, Ctrl + V를 누른 경우:  $D[i-5] * 4$

# 크리보드

N62

?



95

<https://www.acmicpc.net/problem/11058>

- $D[i]$  = 크리보드의 버튼을 총  $i$ 번 눌러서 화면에 출력된  $A$ 개수의 최대값
- 화면에  $A$ 를 출력하는 버튼을 누른 경우:  $D[i-1] + 1$
- 마지막에  $\text{Ctrl} + A$ ,  $\text{Ctrl} + C$ ,  $\text{Ctrl} + V$ 를 누른 경우:  $D[i-3] * 2$
- 마지막에  $\text{Ctrl} + A$ ,  $\text{Ctrl} + C$ ,  $\text{Ctrl} + V$ ,  $\text{Ctrl} + V$ 를 누른 경우:  $D[i-4] * 3$
- 마지막에  $\text{Ctrl} + A$ ,  $\text{Ctrl} + C$ ,  $\text{Ctrl} + V$ ,  $\text{Ctrl} + V$ ,  $\text{Ctrl} + V$ 를 누른 경우:  $D[i-5] * 4$
- 마지막에  $\text{Ctrl} + A$ ,  $\text{Ctrl} + C$ 를 누르고  $\text{Ctrl} + V$ 를  $j$ 번 누른 경우:  $\boxed{D[i-(j+2)] * (j+1)}$

# 크리보드

<https://www.acmicpc.net/problem/11058>

- $D[i]$  = 크리보드의 버튼을 총  $i$ 번 눌러서 화면에 출력된  $A$ 개수의 최대값

- $D[i] = \max(D[i-1]+1, D[i-(j+2)]*(j+1)) \quad (1 \leq j \leq i-3)$

$A+1$  출력

$j$ 번 Ctrl+V 누르는 경우

$N \times N$

$O(N^2)$



# 크리보드

<https://www.acmicpc.net/problem/11058>

- 소스: <http://codeplus.codes/474326a8142c4d728253d132dc561141>

# 파일 합치기

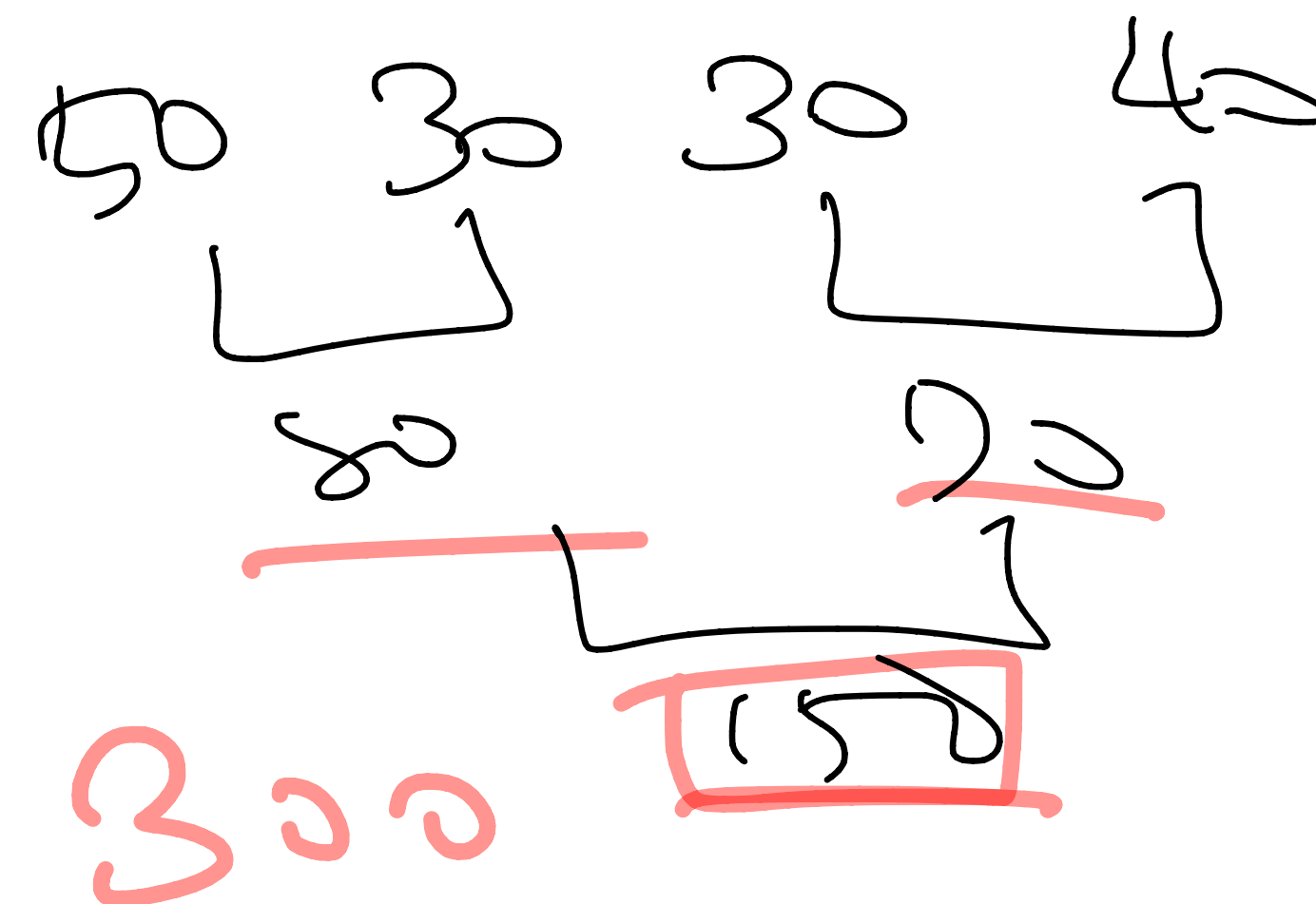
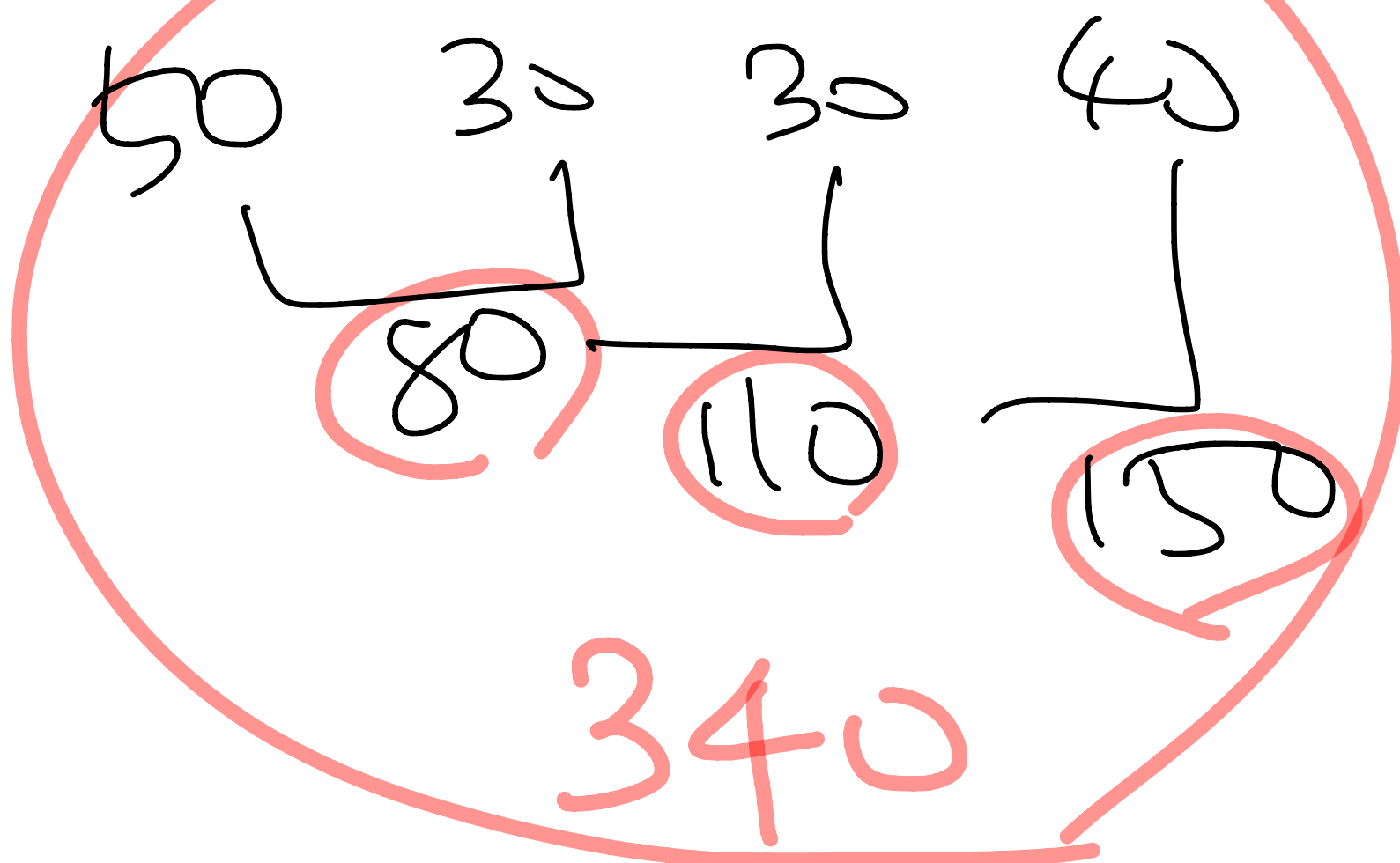
<https://www.acmicpc.net/problem/11066>

N개  $\Rightarrow$  1개  
(크기) AC

크기:  $u+v$   
 $u, v$   $\swarrow$

98

- 각 장이 쓰여진 파일을 합쳐서 최종적으로 소설의 완성본이 들어있는 한 개의 파일을 만들 ~~것~~  $\frac{C}{2}$
- 이 과정에서 두 개의 파일을 합쳐서 하나의 임시파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 소설의 여러 장들이 연속이 되도록 파일을 합쳐나가고, 최종적으로는 하나의 파일로 합친다
- 두 개의 파일을 합칠 때 필요한 비용(시간 등)이 두 파일 크기의 합이라고 가정할 때, 최종적인 한 개의 파일을 완성하는데 필요한 비용의 총 합



# 파일 합치기

<https://www.acmicpc.net/problem/11066>

- 연속된 파일만 합칠 수 있다
- 파일은 2개의 연속된 파일을 합치는 것이다

# 파일 합치기

<https://www.acmicpc.net/problem/11066>

DT[1][5] = 1번 ~ 5번 합치기

100

- 파일이 5개 있다고 하자.  $A_1 A_2 A_3 A_4 A_5$
- 이렇게 5개의 파일을 합치는 방법은 4가지가 있다.

•  $(A_1) (A_2 A_3 A_4 A_5)$

•  $(A_1 A_2) (A_3 A_4 A_5)$

•  $(A_1 A_2 A_3) (A_4 A_5)$

•  $(A_1 A_2 A_3 A_4) (A_5)$

$(1-1) / (2-5)$

$(1-2) / (3-5)$

$(1-3) / (4-5)$

$(1-4) / (5-5)$

$\rightarrow (1-5)$

$\rightarrow (1-5)$

$\rightarrow (1-5)$

$\rightarrow (1-5)$

DT[1][5]

최소 비용  
DT[1][4] + DT[2][5]  
+ 1~5 파일 크기

$\rightarrow DT[1][3] + DT[4][5] + 1-5 \text{ 파일 크기}$

# 파일 합치기

<https://www.acmicpc.net/problem/11066>

101

- $i$ 번째 부터  $j$ 번째까지 파일이 있다고 하자.  $A_i A_{i+1} \dots A_{j-1} A_j$
- 파일을 합치는 방법은 다음과 같이 나타낼 수 있다.

$(A_i A_{i+1} \dots A_k) (A_{k+1} \dots A_{j-1} A_j)$

$\min(D[i][k] + D[k+1][j]) + (\text{[}i\text{번째} \sim j\text{번째} \text{파일 크기의 합})$

$i \leq k \leq j-1$

$O(N)$

$N^2$

$D[i][j]$

$O(N^3)$

# 파일 합치기

<https://www.acmicpc.net/problem/11066>

- $D[i][j]$  =  $i$ 번째 부터  $j$ 번째까지 파일을 하나로 합치는 비용
- $i$ 번째 부터  $j$ 번째까지 파일이 있다고 하자.  $A_i A_{i+1} \dots A_{j-1} A_j$
- 파일을 합치는 방법은 다음과 같이 나타낼 수 있다.
- $(A_i A_{i+1} \dots A_k) (A_{k+1} \dots A_{j-1} A_j)$
- $D[i][k] + D[k+1][j] + \text{합치는 비용}$
- $i \leq k < j$

# 파일 합치기

103

<https://www.acmicpc.net/problem/11066>

- 소스: <http://codeplus.codes/23a0fdf5993b4f88bf7e7fb34a1c504b>

# 평범한 배낭

<https://www.acmicpc.net/problem/12865>

- N개의 물건이 있고, 각 물건은 무게  $W[i]$ 와 가치  $V[i]$ 를 갖는다.
- 배낭에는 무게 K까지만 물건을 넣을 수 있다.
- 배낭에 넣을 수 있는 물건 가치의 최댓값을 구하는 문제

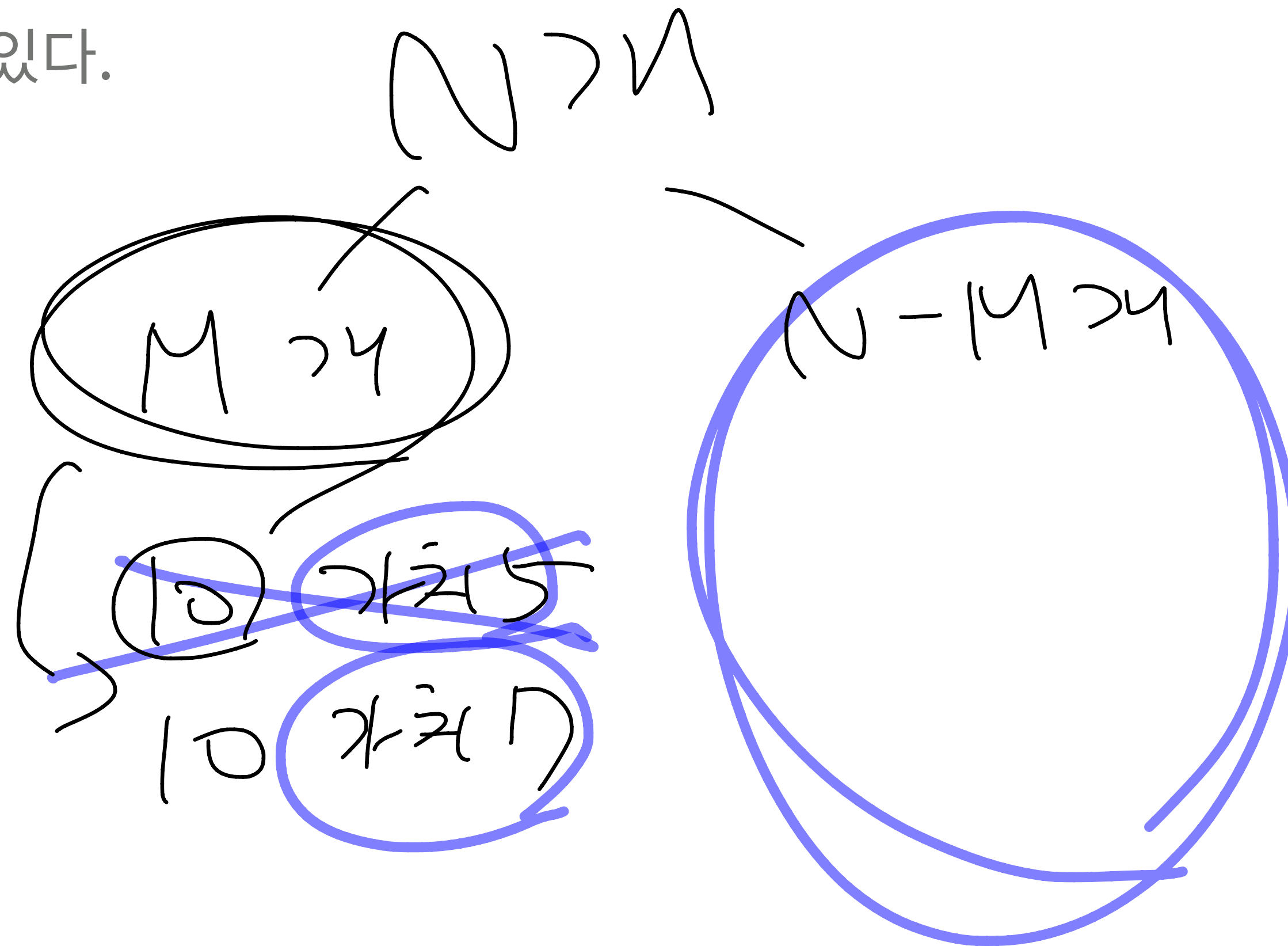


# 평범한 배낭

105

<https://www.acmicpc.net/problem/12865>

- 각각의 물건을 배낭에 넣는 경우와 넣지 않는 경우가 있다.
- 그렇다면, 총  $2^N$  가지의 경우가 있다.
- $1 \leq N \leq 100$



# 평범한 배낭

106

<https://www.acmicpc.net/problem/12865>

- 배낭의 무게  $K \leq 100,000$  이다.
- 각각의 무게에 대해서, 넣을 수 있는 가장 큰 가치를 알면 된다.

# 평범한 배낭

107

<https://www.acmicpc.net/problem/12865>

- $D[i][j]$  =  $i$ 번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이  $j$ 일 때, 가치의 최댓값

# 평범한 배낭

108

<https://www.acmicpc.net/problem/12865>

•  $D[i][j]$  =  $i$ 번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이  $j$ 일 때, 가치의 최댓값

•  $i$ 번째 물건을 가방에 넣지 않은 경우

•  $i$ 번째 물건을 가방에 넣은 경우

$$D[i-1][S]$$

$$D[i-1][S - w[i]] + v[i]$$

# 평범한 배낭

109

<https://www.acmicpc.net/problem/12865>

- $D[i][j]$  =  $i$ 번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이  $j$ 일 때, 가치의 최댓값
- $i$ 번째 물건을 가방에 넣지 않은 경우:  $D[i-1][j]$
- $i$ 번째 물건을 가방에 넣은 경우

# 평범한 배낭

110

<https://www.acmicpc.net/problem/12865>

- $D[i][j]$  =  $i$ 번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이  $j$ 일 때, 가치의 최댓값
- $i$ 번째 물건을 가방에 넣지 않은 경우:  $D[i-1][j]$
- $i$ 번째 물건을 가방에 넣은 경우:  $D[i-1][j-w[i]] + v[i]$

# 평범한 배낭

111

<https://www.acmicpc.net/problem/12865>

Tab, 무게(S)

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=k; j++) {  
        d[i][j] = d[i-1][j];  
        if (j-w[i] >= 0) {  
            d[i][j] = max(d[i][j], d[i-1][j-w[i]]+v[i]);  
        }  
    }  
}
```

# 평범한 배낭

112

<https://www.acmicpc.net/problem/12865>

- 소스: <http://codeplus.codes/ecb361131f39438482922f5096d25949>

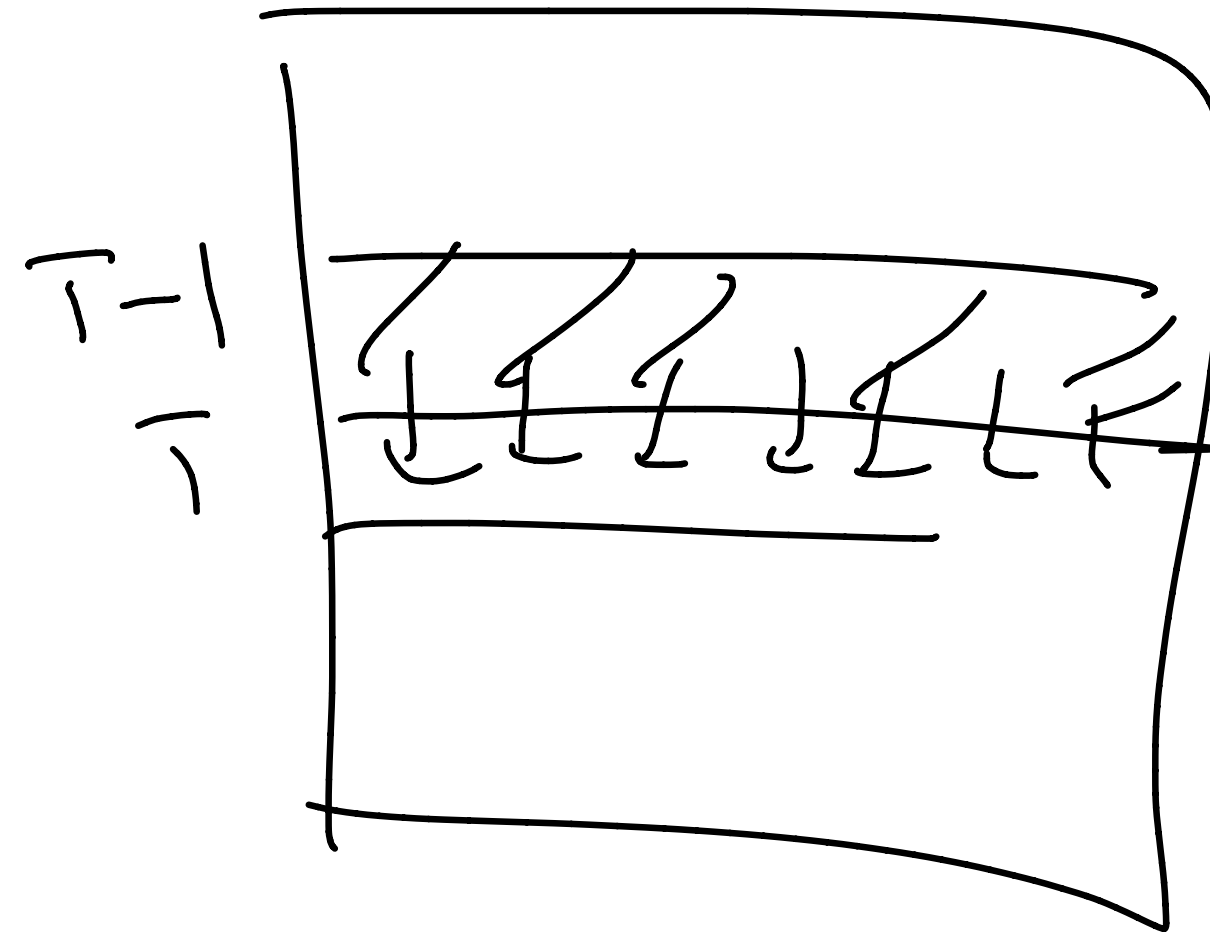


# 평범한 배낭

113

<https://www.acmicpc.net/problem/12865>

- $D[i][j]$  =  $i$ 번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이  $j$ 일 때, 가치의 최댓값
  - $i$ 번째 물건을 가방에 넣지 않은 경우:  $D[i-1][j]$
  - $i$ 번째 물건을 가방에 넣은 경우:  $D[i-1][j-w[i]] + v[i]$
  - $D[i][j] = \max(D[i-1][j], D[i-1][j-w[i]] + v[i])$  이기 때문에
  - 1차원 다이나믹으로도 해결할 수 있다.



# 평범한 배낭

<https://www.acmicpc.net/problem/12865>

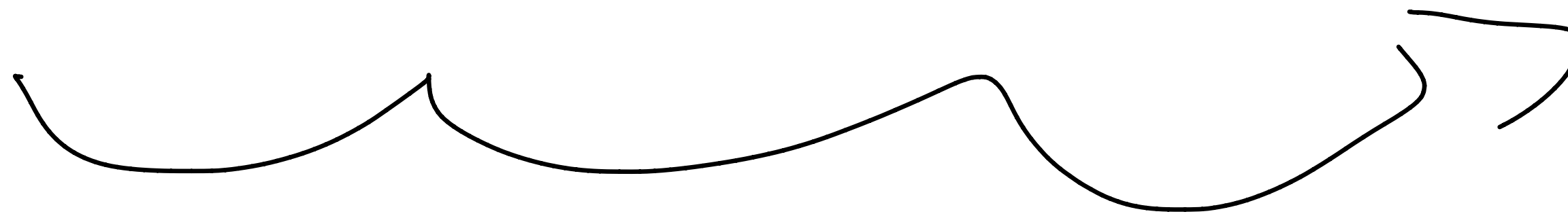
- $D[i][j]$  =  $i$ 번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이  $j$ 일 때, 가치의 최댓값
- $i$ 번째 물건을 가방에 넣지 않은 경우:  $D[i-1][j]$
- $i$ 번째 물건을 가방에 넣은 경우:  $D[i-1][j-w[i]] + v[i]$
- $D[i][j] = \max(D[i-1][j], D[i-1][j-w[i]] + v[i])$  이기 때문에
- 1차원 다이나믹으로도 해결할 수 있다.
- $D[j] = \max(D[j], D[j-w[i]] + v[i])$

# 평범한 배낭

115

<https://www.acmicpc.net/problem/12865>

```
for (int i=1; i<=n; i++) {  
    for (int j=1; j<=k; j++) {  
        if (j-w[i] >= 0) {  
            d[j] = max(d[j], d[j-w[i]]+v[i]);  
        }  
    }  
}
```



# 평범한 배낭

<https://www.acmicpc.net/problem/12865>

- 앞 페이지의 소스와 같은 방법은 물건을 중복해서 사용하게 된다.
- 중복 없이 배열을 채워야 한다.

# 평범한 배낭

<https://www.acmicpc.net/problem/12865>

- 앞 페이지의 소스와 같은 방법은 물건을 중복해서 사용하게 된다.
- 중복 없이 배열을 채워야 한다.
- 배열 채우는 순서를 뒤에서부터 채우면 된다.

# 평범한 배낭

118

<https://www.acmicpc.net/problem/12865>

```
for (int i=1; i<=n; i++) {  
    for (int j=k; j>=1; j--) {  
        if (j-w[i] >= 0) {  
            d[j] = max(d[j], d[j-w[i]]+v[i]);  
        }  
    }  
}
```

# 평범한 배낭

119

<https://www.acmicpc.net/problem/12865>

- 소스: <http://codeplus.codes/d45a11cb1f2e411c9a0f6617036a0e4c>

# 기타리스트

<https://www.acmicpc.net/problem/1495>

120

첫 볼륨:  $S$

$N \leq 100$

연주해야 하는 곡의 개수  $N$ 개

가능한 볼륨의 범위  $0 \sim M$

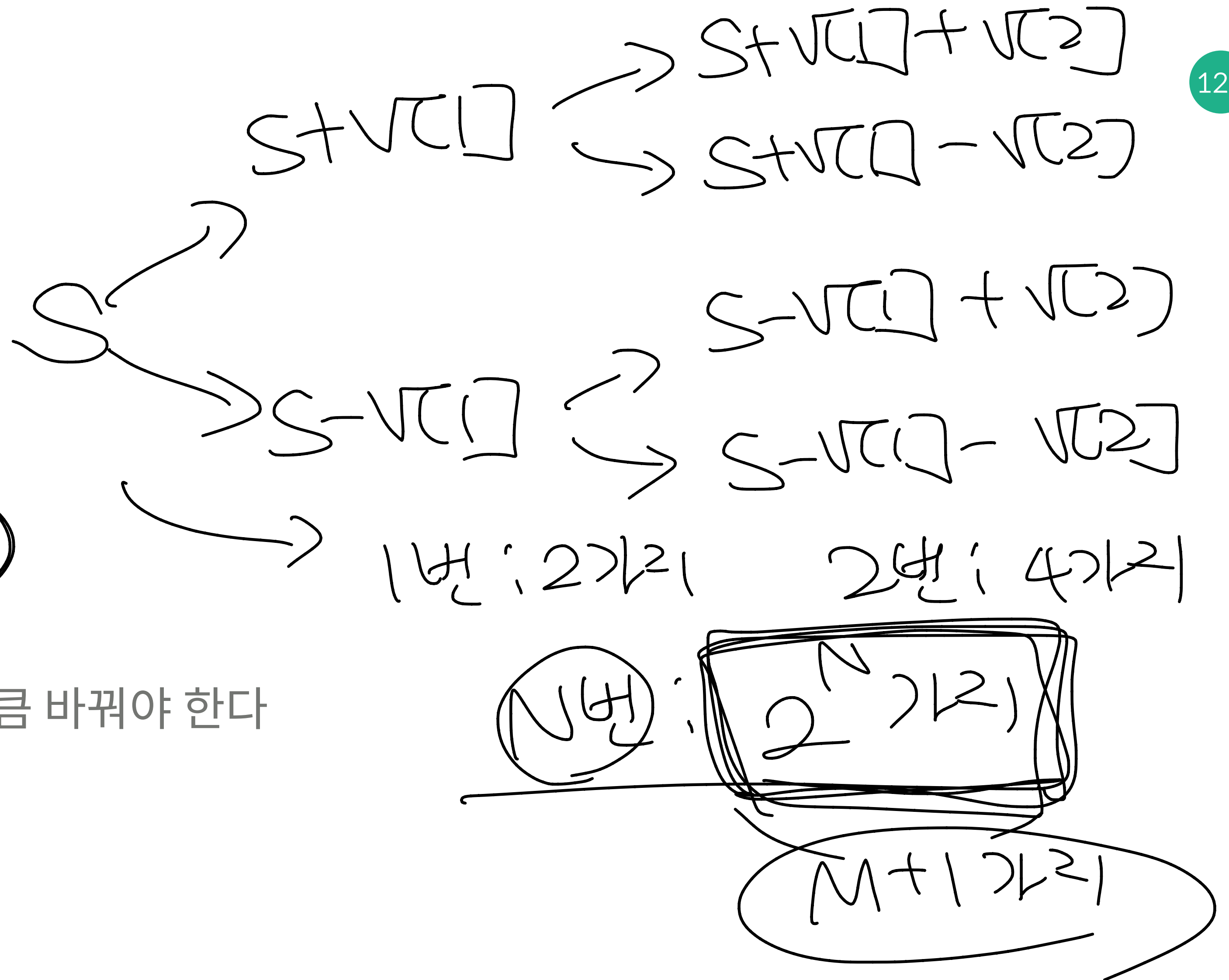
$M+1$

$i$ 번 곡을 연주하기 전에 볼륨을  $V[i]$ 만큼 바꿔야 한다

$i$ 번 곡을 연주하기 직전 볼륨이  $P$ 라면

$i$ 번 곡은  $P+V[i]$  또는  $P-V[i]$ 로 연주해야 한다

마지막 곡을 연주할 수 있는 볼륨 중 최대값





- $D[i][j]$  = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0

- $N = 3, S = 5, M = 10$

[illegible]

- $D[i][j]$  = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- $N = 3, S = 5, M = 10$
- $V[1] = 2, V[2] = 3, V[3] = 1$

[illegible]

- $D[i][j]$  = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- $N = 3, S = 5, M = 10$
- $V[1] = 2, V[2] = 3, V[3] = 1$

[illegible]

# 기타리스트

O-M

124

<https://www.acmicpc.net/problem/1495>

- $D[i][j]$  =  $i$ 번 곡을 볼륨  $j$ 로 연주할 수 있으면 1 없으면 0
- $N = 3, S = 5, M = 10$
- $V[1] = 2, V[2] = 3, V[3] = 1$

	0	1	2	3	4	5	6	7	8	9	10
0						1					
1				1				1			
2	1	X	X	X	1	X	1	X	X	X	1
3											

Handwritten annotations on the table:

- A vertical line is drawn to the left of the first column (index 0).
- Gray arrows indicate the path of maximum volume: from (2,0) to (1,3) to (0,5) to (1,7) to (2,10).
- Handwritten '1's are placed in the cells (2,0), (1,3), (0,5), (1,7), and (2,10), which correspond to the '1's in the gray path.
- Handwritten 'X's are placed in the cells (2,1), (2,2), (2,3), (2,4), (2,6), (2,8), and (2,9).
- Handwritten arrows point from the '1's in the gray path to the 'X's in the same row, indicating that those states are not optimal.
- Handwritten '1's are also present in the cells (1,4), (1,6), and (1,9).

# 기타리스트

125

<https://www.acmicpc.net/problem/1495>

- $D[i][j]$  =  $i$ 번 곡을 볼륨  $j$ 로 연주할 수 있으면 1 없으면 0
- $N = 3, S = 5, M = 10$
- $V[1] = 2, V[2] = 3, V[3] = 1$

가능이ㄴ

	0	1	2	3	4	5	6	7	8	9	10
0						1					
1				1				1			
2	1				1		1				1
3		1		1		1		1		1	

Handwritten arrows and a circle highlight the path from (2,0) to (3,1) to (3,3) to (3,5) to (3,7) to (3,9), indicating a valid sequence of volume adjustments.

# 기타리스트

<https://www.acmicpc.net/problem/1495>

- $D[i][j]$  = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- $D[i][j]$  가 1이면
- $D[i+1][j+V[i+1]]$ 와  $D[i+1][j-V[i+1]]$  을 1로 만들 수 있다.

---

# 기타리스트

127

<https://www.acmicpc.net/problem/1495>

- 소스: <http://codeplus.codes/e915910025124f3a99d2907de26572f7>

# 1학년

128

<https://www.acmicpc.net/problem/5557>

$N-2$

$N \leq 100$

- 마지막 두 숫자 사이에 '='을 넣고, 나머지 숫자 사이에는 '+' 또는 '-'를 넣어 등식을 만든다
- 예를 들어, "83248724088"에서 등식 "8+3-2-4+8-7-2-4-0+8=8"을 만들 수 있다

$N-1$   
+ + + + +  
+ + + + +  
N개

$2^N$  가지

0-20



# 1학년

129

<https://www.acmicpc.net/problem/5557>

- $D[i][j]$  =  $i$ 까지 수를 사용해서  $j$ 를 만드는 방법의 수

$$D[i][j - A[i]]$$

$$D[i][j + A[i]]$$

$i-1$ 까지  $j - A[i]$

0 ~ 20

$$A[i] \pm \dots + A[i] = j$$

$$= j$$

$$A[i] \pm \dots - A[i] = j$$

$$= j$$

$i-1$ 까지  $j + A[i]$

# 1학년

130

<https://www.acmicpc.net/problem/5557>

- $D[i][j] = D[i-1][j-A[i]] + D[i-1][j+A[i]]$

# 1학년

131

<https://www.acmicpc.net/problem/5557>

- 소스: <http://codeplus.codes/4ec2fc8789784450b4b964aefdb76046>

# 괄호

<https://www.acmicpc.net/problem/10422>



- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제

# 괄호

133

<https://www.acmicpc.net/problem/10422>

- 길이  $L$ 이 주어졌을 때, 길이가  $L$ 인 올바른 괄호 문자열의 개수를 구하는 문제

올바른 괄호 문자열

# 괄호

134

<https://www.acmicpc.net/problem/10422>

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까?

(

올바른 괄호 문자열

# 괄호

135

<https://www.acmicpc.net/problem/10422>

- 길이  $L$ 이 주어졌을 때, 길이가  $L$ 인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다

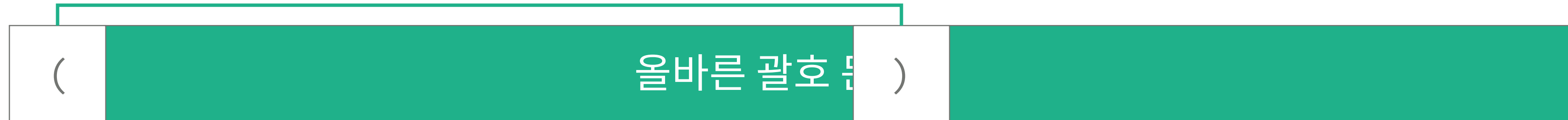


# 괄호

136

<https://www.acmicpc.net/problem/10422>

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다 (i번째 글자)

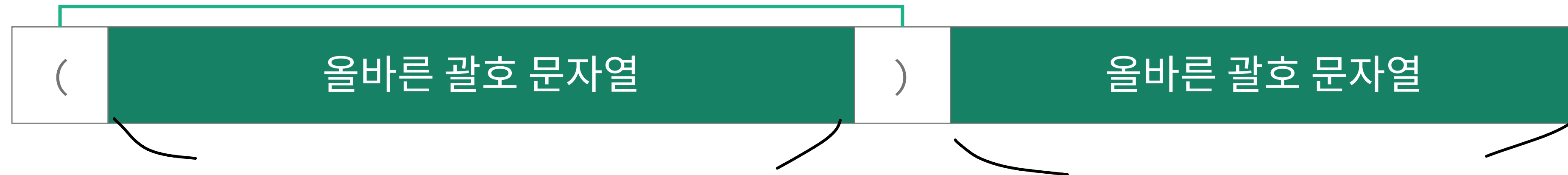




# 괄호

<https://www.acmicpc.net/problem/10422>

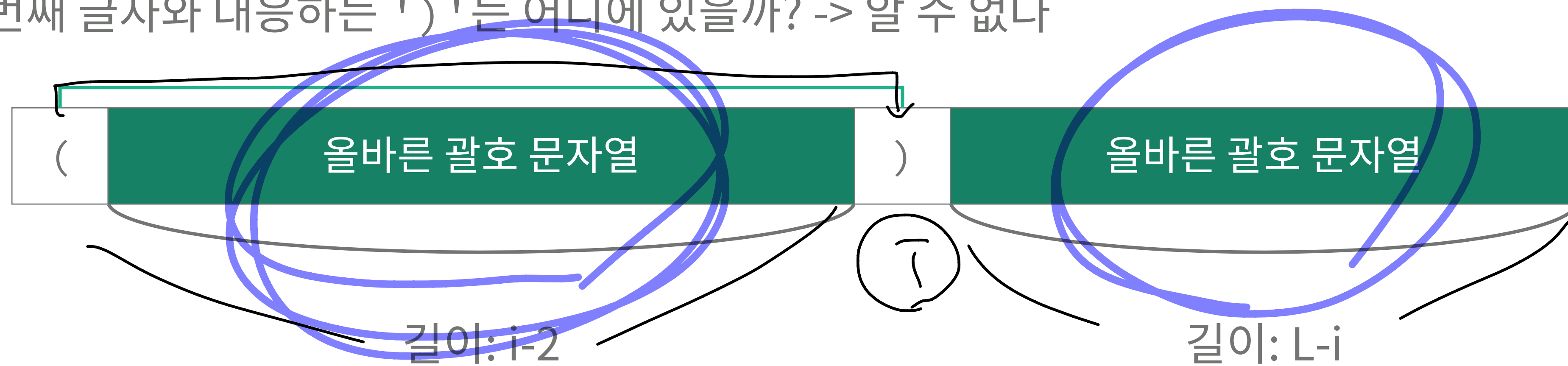
- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다



# 괄호

<https://www.acmicpc.net/problem/10422>

- 길이  $L$ 이 주어졌을 때, 길이가  $L$ 인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다



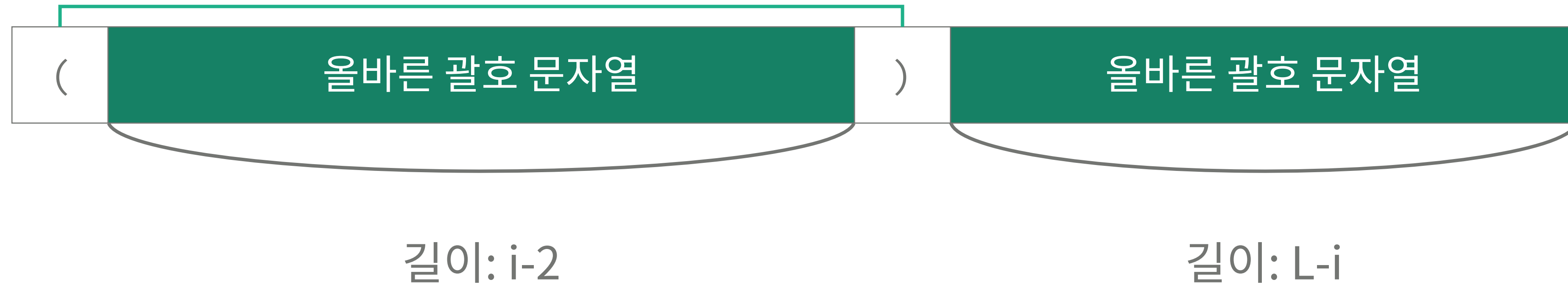
# 괄호

<https://www.acmicpc.net/problem/10422>

$N^2$

139

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다



- $D[L]$  = 길이가 L인 올바른 괄호 문자열의 개수
- $D[L] = \sum (D[i-2] * D[L-i])$

# 괄호

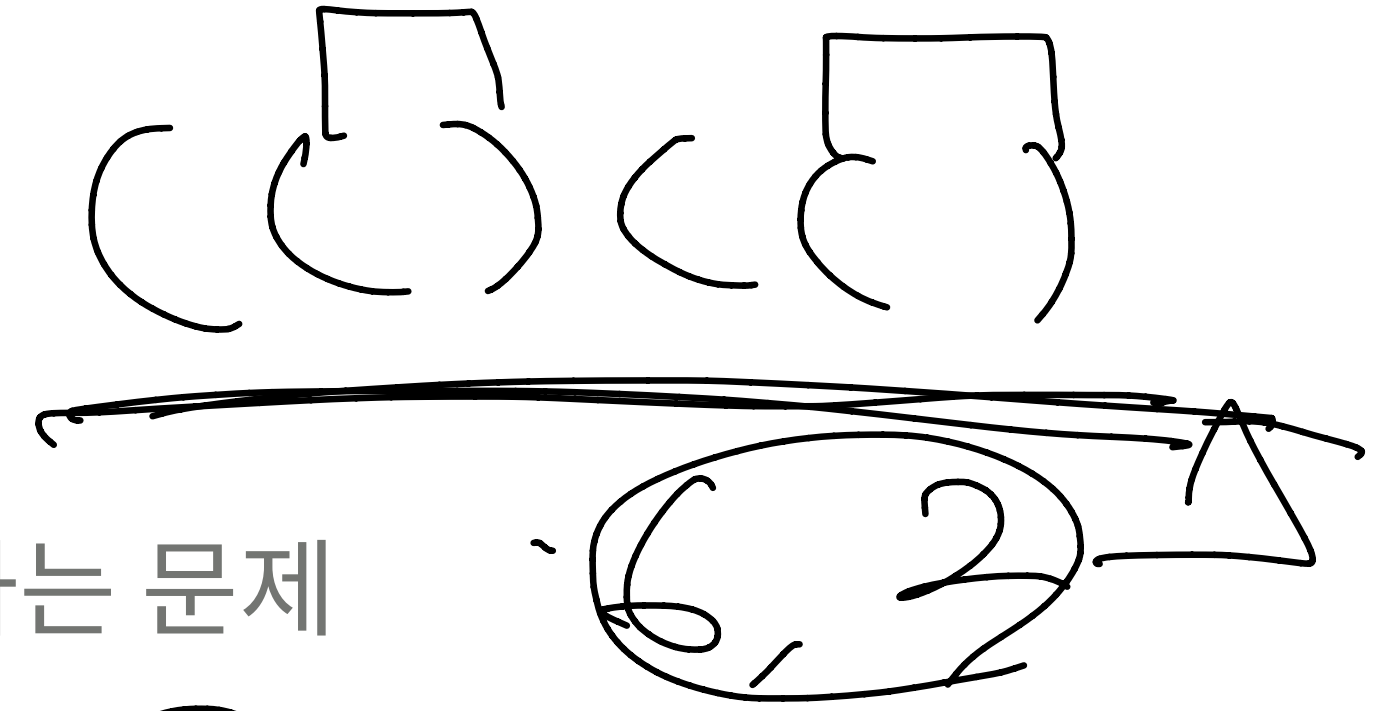
140

<https://www.acmicpc.net/problem/10422>

- 소스: <http://codeplus.codes/4c5f733557224147a47661c8212ef3ce>

# 괄호

( ( ( ) ) )



141

<https://www.acmicpc.net/problem/10422>

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- $D[N][0]$  = 길이가 N인 괄호 문자열, 짜이 맞지 않는 여는 괄호의 개수 0개
- 길이가 L인 올바른 괄호 문자열:  $D[L][0]$

괄호 문자열

# 괄호

142

<https://www.acmicpc.net/problem/10422>

- 길이  $L$ 이 주어졌을 때, 길이가  $L$ 인 올바른 괄호 문자열의 개수를 구하는 문제
- $D[N][O]$  = 길이가  $N$ 인 **괄호 문자열**, 짝이 맞지 않는 여는 괄호의 개수:  $O$ 개

괄호 문자열

)

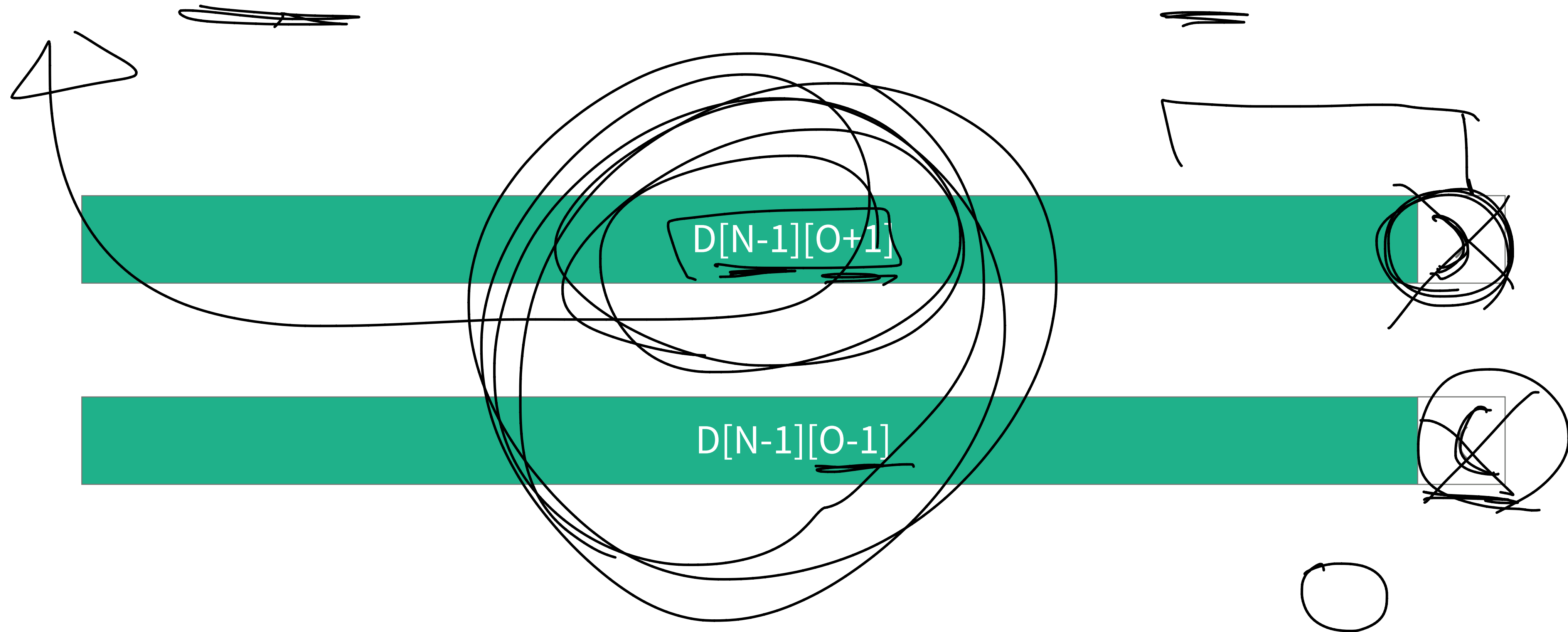
괄호 문자열

(

# 괄호

<https://www.acmicpc.net/problem/10422>

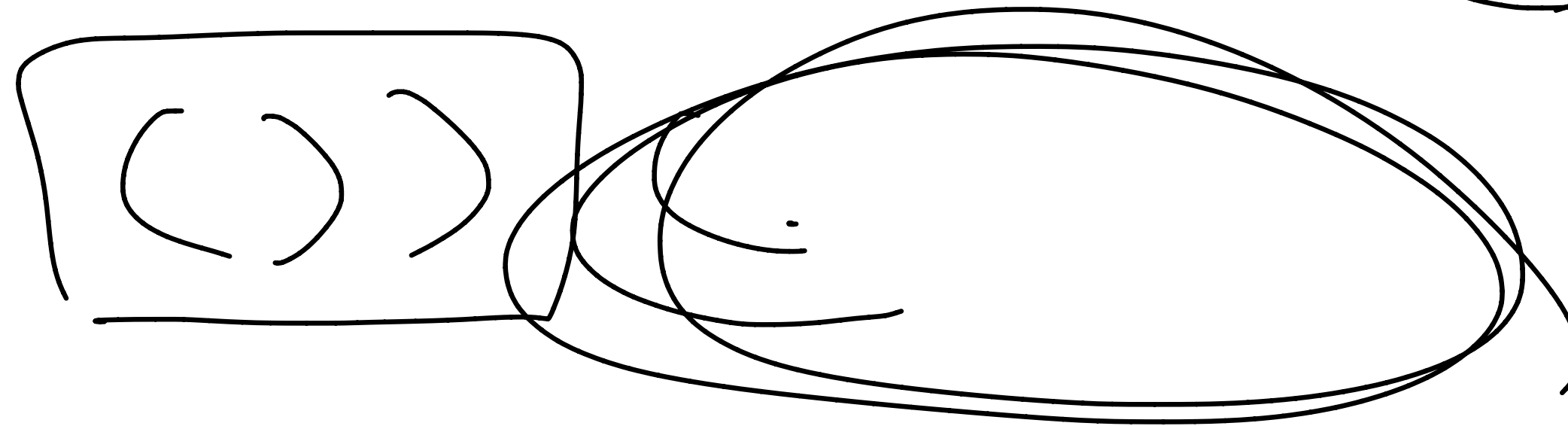
- 길이  $L$ 이 주어졌을 때, 길이가  $L$ 인 올바른 괄호 문자열의 개수를 구하는 문제
- $D[N][O]$  = 길이가  $N$ 인 괄호 문자열, 짝이 맞지 않는 여는 괄호의 개수:  $O$ 개



# 괄호

<https://www.acmicpc.net/problem/10422>

- 길이  $L$ 이 주어졌을 때, 길이가  $L$ 인 올바른 괄호 문자열의 개수를 구하는 문제
- $D[N][O]$  = 길이가  $N$ 인 **괄호 문자열**, 짝이 맞지 않는 여는 괄호의 개수:  $O$ 개
- $O < 0$ 이 되면 절대로 올바른 괄호 문자열을 만들 수 없기 때문에,  $O \geq 0$ 에 대해서만 구한다





# 괄호

145

<https://www.acmicpc.net/problem/10422>

- 소스: <http://codeplus.codes/5110acc7f2aa438bbb77f47721d21707>

끝

---

# 코드 플러스

147

<https://code.plus>

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 [codeplus@startlink.io](mailto:codeplus@startlink.io) 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.