

# 그리디 알고리즘

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

---

# 그리디 알고리즘

---

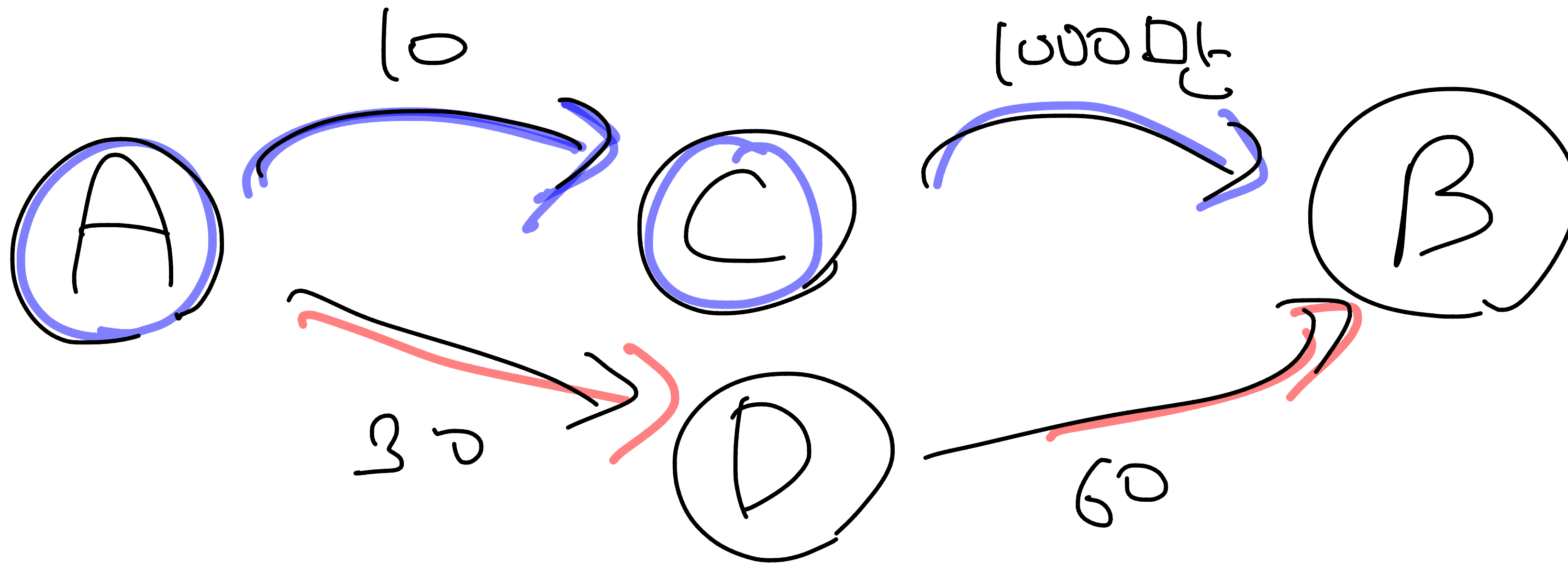
# 그리디 알고리즘

Greedy Algorithm

기존

3

- 결정해야 할 때, 그 순간에 가장 좋다고 생각하는 것을 선택하면서 답을 찾아가는 알고리즘
- 그 때 그 때는 최적일지도 모르지만, 최종적으로는 답이 최적이 아닐 수도 있다.



# 거스름돈 문제

Greedy Algorithm

10000원

2430원

- ~~1, 5, 10, 50, 100, 500원 동전을 매우 많이 가지고 있고~~
- 1000, 5000, 10000, 50000원 지폐를 매우 많이 가지고 있을 때
- N원을 거슬러주는 문제
- 이 때, 사용하는 지폐와 동전의 개수를 최소화 해야 한다.

가장 큰 액면가를 가진 지폐나 동전부터 거슬러 주자.

~~10원 x 151~~

증명

771

7570원

5000원 x 124

2570원

1000원 x 271

570원

500원 x 127

170원

50, 10 x 2      327

# 거스름돈 문제

5

Greedy Algorithm

- 7570원을 거슬러 받아야 한다
- 5000원 1장
- $7570 - 5000 = 2570$
- 1000원 2장
- $2570 - 2000 = 570$
- 500원 1개
- $570 - 500 = 70$
- 50원 1개
- $70 - 50 = 20$
- 10원 2개

# 거스름돈 문제

Greedy Algorithm

- 동전이 1, 4, 5원 있는 경우
- 12원을 거슬러주는 경우에는
- 5원 2개
- $12 - 10 = 2$
- 1원 2개
- 총 4개가 필요하다
- 하지만, 정답은 4원 3개이다

$$\underline{4 \times 3}$$

$$\begin{array}{r} \overset{\times 4}{12} \\ \underline{5 \times 2 = 10} \\ 2 \end{array}$$

$$\underline{2} \\ 1 \times 2 = 2$$

4개

(12-24)

# 그리디 알고리즘

Greedy Algorithm

- 언제 그리디 알고리즘을 쓰나?
- 지금 이 순간 가장 좋은 경우를 선택하는 것이 항상 최적인 경우에
- 그래서 쉽다

17517082

10 x 17517

505050  
10

# 그리디 알고리즘

Greedy Algorithm

8

- 언제 그리디 알고리즘을 쓰나?
- 지금 이 순간 가장 좋은 경우를 선택하는 것이 항상 최적인 경우에
- ~~그래서 쉽다~~
- 가장 어렵다
- 그것이 왜 최적이 되는지를 증명해야하기 때문



# 동전 0

<https://www.acmicpc.net/problem/11047>

- 준규가 가지고 있는 동전은 총  $N$ 종류이고, 각각의 동전을 매우 많이 가지고 있다.
- 동전을 적절히 사용해서 그 가치의 합을  $K$ 로 만드려고 한다. 이 때 필요한 동전 개수의 최소값을 구하는 프로그램을 작성하시오.
- $N$ 개의 줄에 동전의 가치  $A_i$ 가 오름차순으로 주어진다. ( $1 \leq A_i \leq 1,000,000$ ,  $A_1 = 1$ ,  $i \geq 2$ 인 경우에  $A_i$ 는  $A_{i-1}$ 의 배수)

# 동전 0

<https://www.acmicpc.net/problem/11047>

- 가치가  $A_i$ 인 동전을  $A_{i+1}/A_i$ 개보다 적게 사용한 것이 정답이다. ( $i < N$ )
- $A_i$ 를  $A_{i+1}/A_i$ 개 사용했다면  $A_{i+1}$  1개로 변경하면 더 최소가 된다.
- 다음을 증명하면 그리디 알고리즘을 증명할 수 있다.
- 모든 정답 중에서  $A_i$ 를 사용하지 않고 만든 것 중  $K$ 의 최댓값은  $A_i-1$ 이다.
- 다음 페이지부터 증명

# 동전 0

<https://www.acmicpc.net/problem/11047>

- $A_i$ 는 최대  $A_{i+1}/A_i$ 개 사용할 수 있기 때문에,  $A_i$ 를 사용하지 않고 만들 수 있는 최대 금액은 다음과 같다.
- $(A_2/A_1 - 1)A_1 + (A_3/A_2 - 1)A_2 + \cdots + (A_i/A_{i-1} - 1)A_{i-1}$
- 식을 정리하면
- $(A_2 - A_1) + (A_3 - A_2) + \cdots + (A_i - A_{i-1})$
- 더 정리하면
- $A_i - A_1 = A_i - 1$  이다.
- 따라서,  $K \geq A_i$ 면  $A_i$ 가 꼭 포함되어야 한다.  $A_i$ 가 포함되었다고 해도  $K < A_{i+1}$  이면 최소를 만들 수 없다.

# 동전 0

<https://www.acmicpc.net/problem/11047>

- 앞의 증명을 이용해서 그리디 방법이 맞음을 알 수 있다.
- $A_i \leq K < A_{i+1}$  라면  $A_i$ 가 꼭 하나 포함되어야 하고
- 이제  $K - A_i$ 를 최소로 만들면 된다.
- $K - A_i$ 를 최소로 만드는 것도 그리디 알고리즘을 이용할 수 있다.

# 동전 0

<https://www.acmicpc.net/problem/11047>

- 소스: <http://codeplus.codes/58e2ebdd73724a0c882afab4063acfd9>

# 회의실 배정 $O(N \log N)$ $O(N)$

<https://www.acmicpc.net/problem/1931>

- 한 개의 회의실이 있는데 이를 사용하고자 하는 n개의 회의들에 대하여 회의실 사용표를 만들려고 한다  
↓ ↓
- 각 회의 i에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 최대 수의 회의를 찾아라
- 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다
- 회의의 시작시간과 끝나는 시간이 같을 수도 있다
- 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

# 회의실 배정

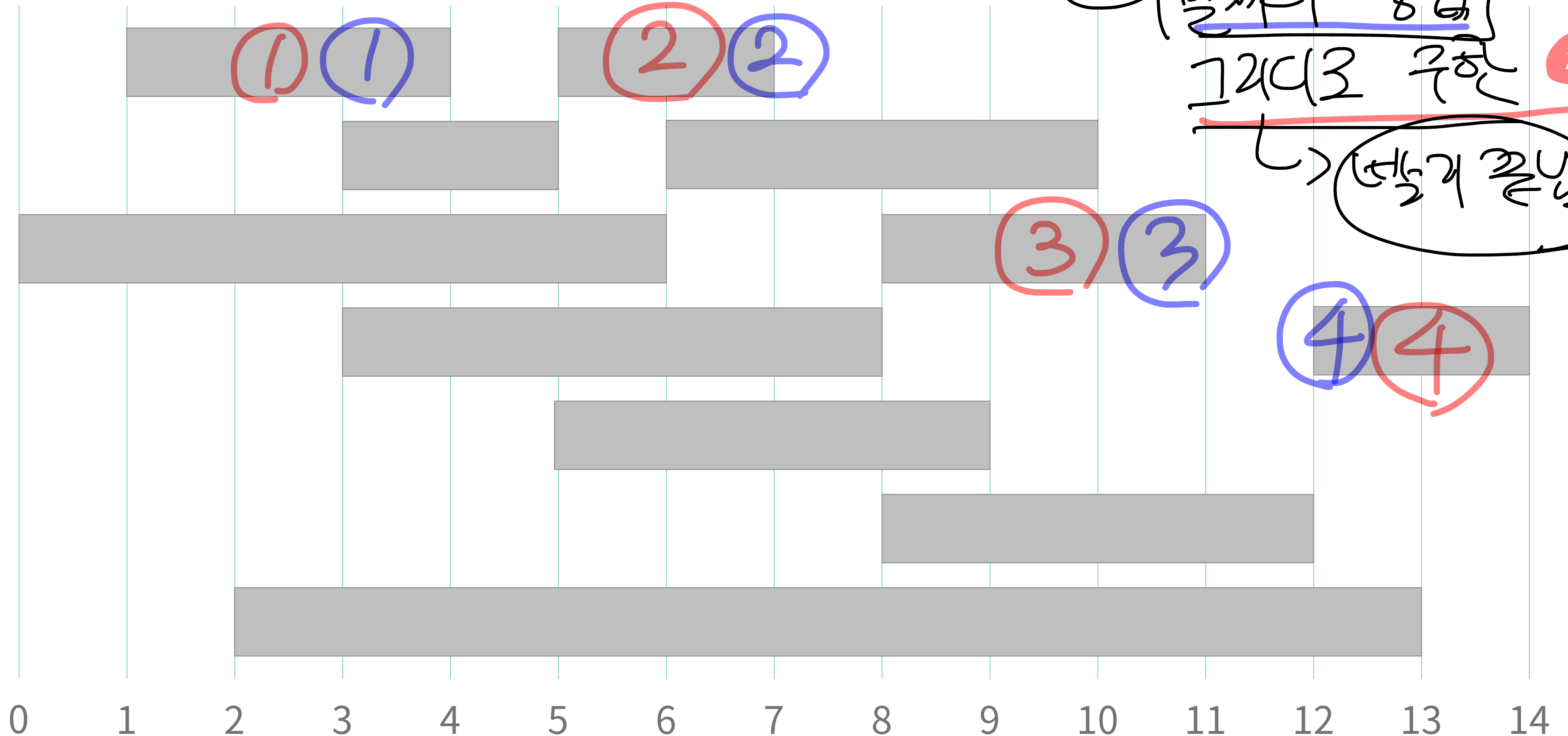
<https://www.acmicpc.net/problem/1931>

A — 22

47-1

15

12개의 정답  
그리고 3 구간  
→ 이렇게 끝남



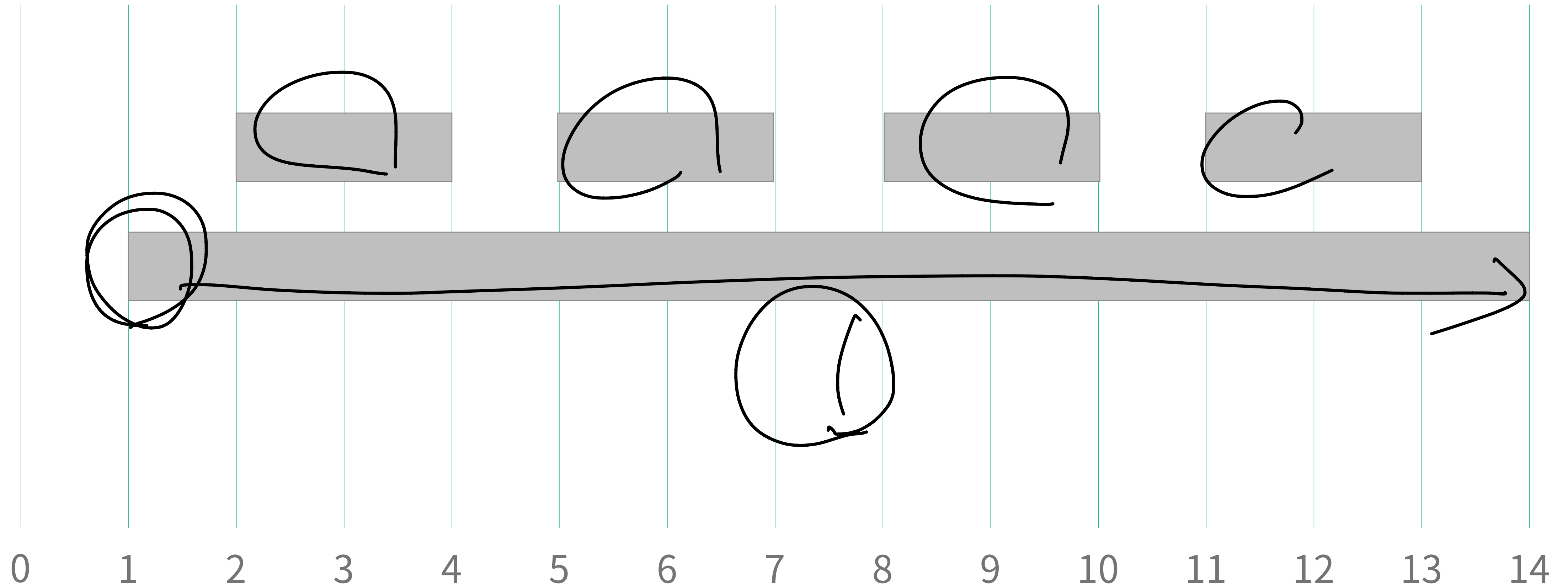
# 회의실 배정

16

<https://www.acmicpc.net/problem/1931>

- 일찍 시작하는 회의를 배정한다

4

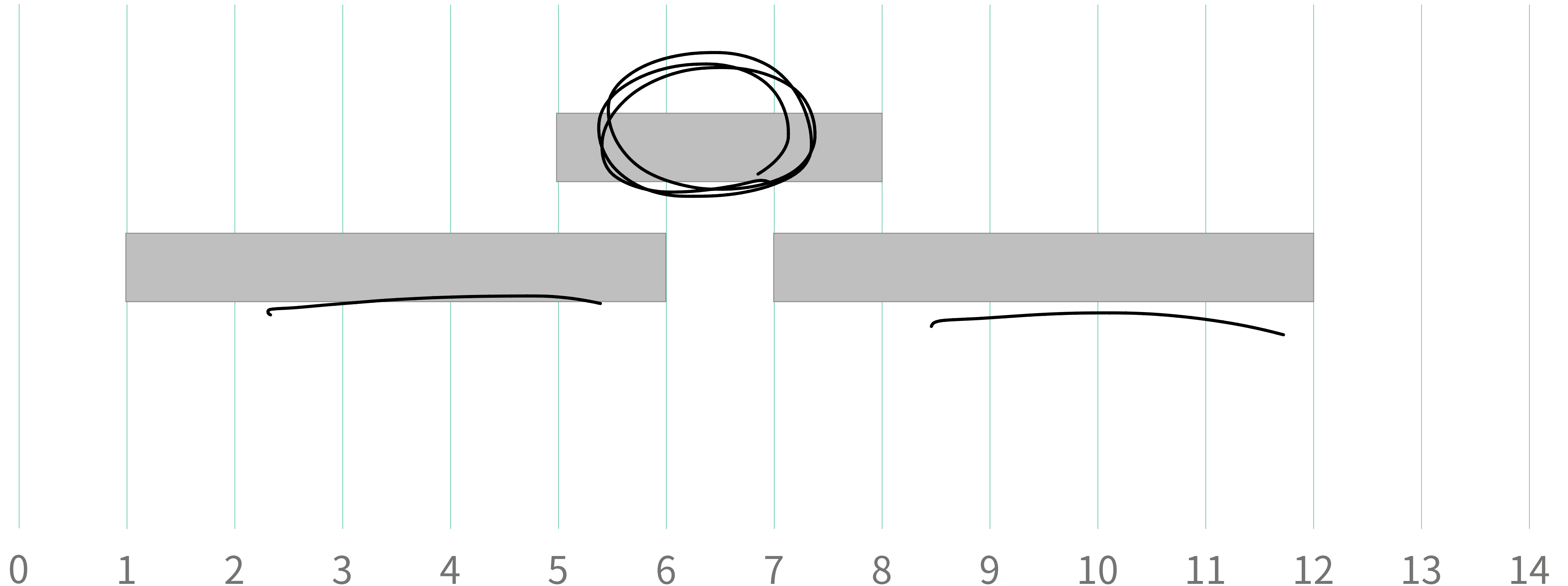
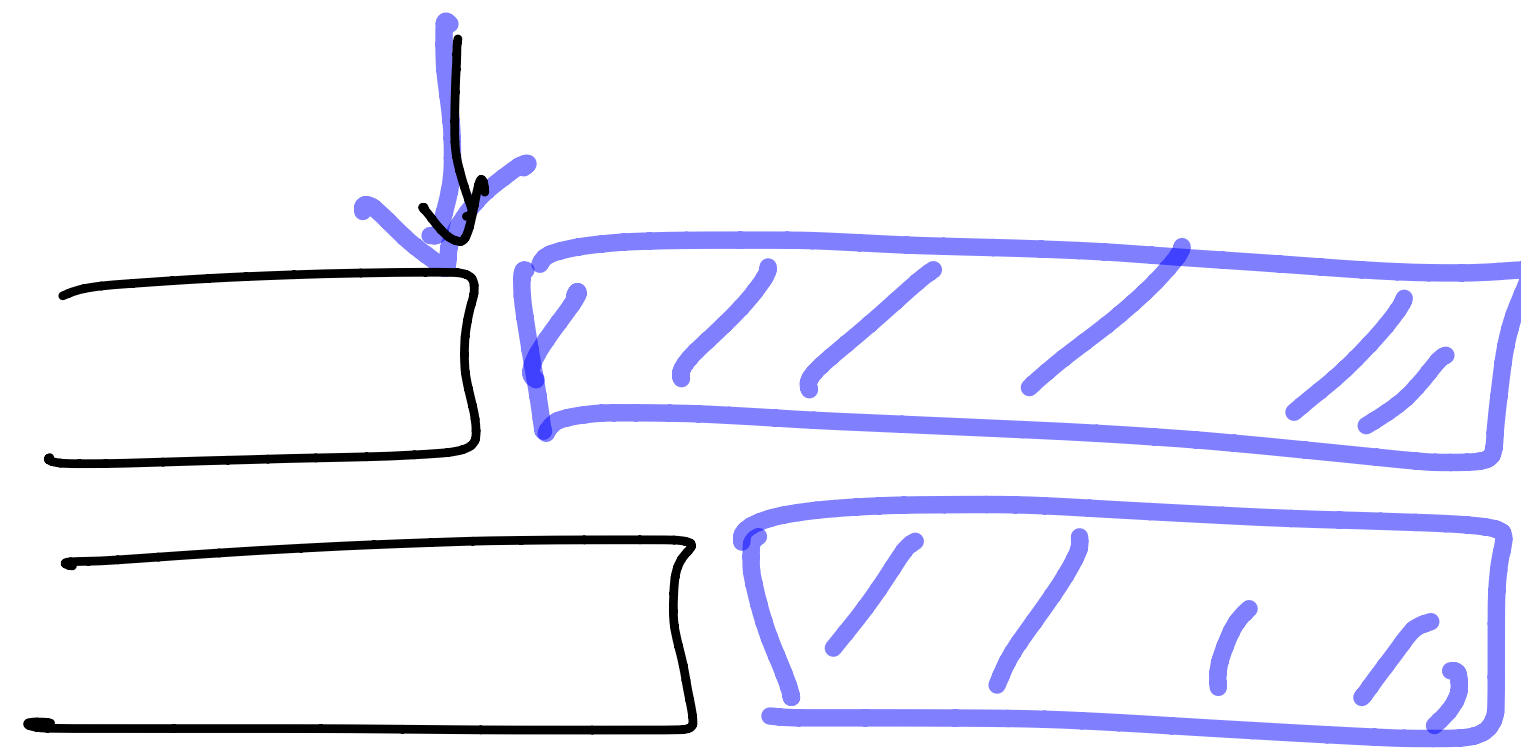




# 회의실 배정

<https://www.acmicpc.net/problem/1931>

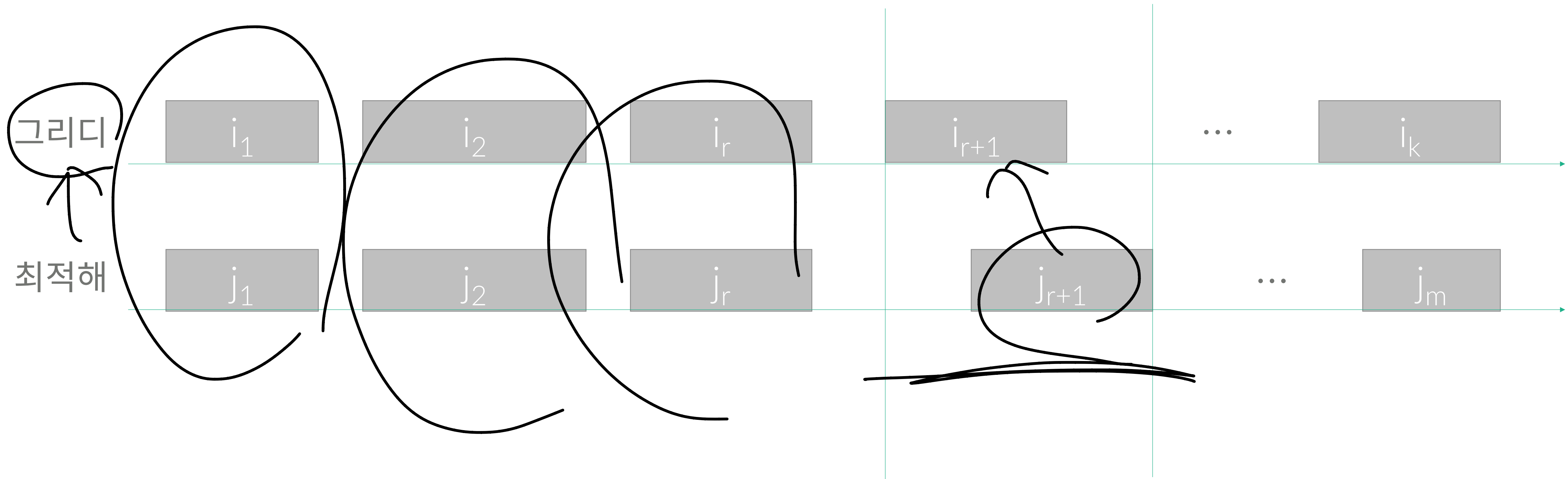
- 짧은 회의를 먼저 배정한다.



# 회의실 배정

<https://www.acmicpc.net/problem/1931>

- $i_1, i_2, i_3, \dots, i_k$ 를 그리디 알고리즘으로 선택한 정답
- $j_1, j_2, j_3, \dots, j_m$ 을  $i_1 = j_1, i_2 = j_2, \dots, i_r = j_r$  중에서  $r$ 이 가장 큰 최적해라고 하자



# 회의실 배정

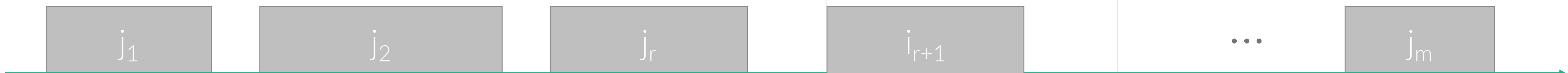
<https://www.acmicpc.net/problem/1931>

- $j_{r+1}$ 보다 일찍 끝나는  $i_{r+1}$ 이 있다고 했을 때  $j_{r+1}$ 을  $i_{r+1}$ 로 바꿔도 정답이다.

그리디



최적해



# 회의실 배정

20

<https://www.acmicpc.net/problem/1931>

- 소스: <http://codeplus.codes/b2ceb977823f4d0a9425f565183d47ad>

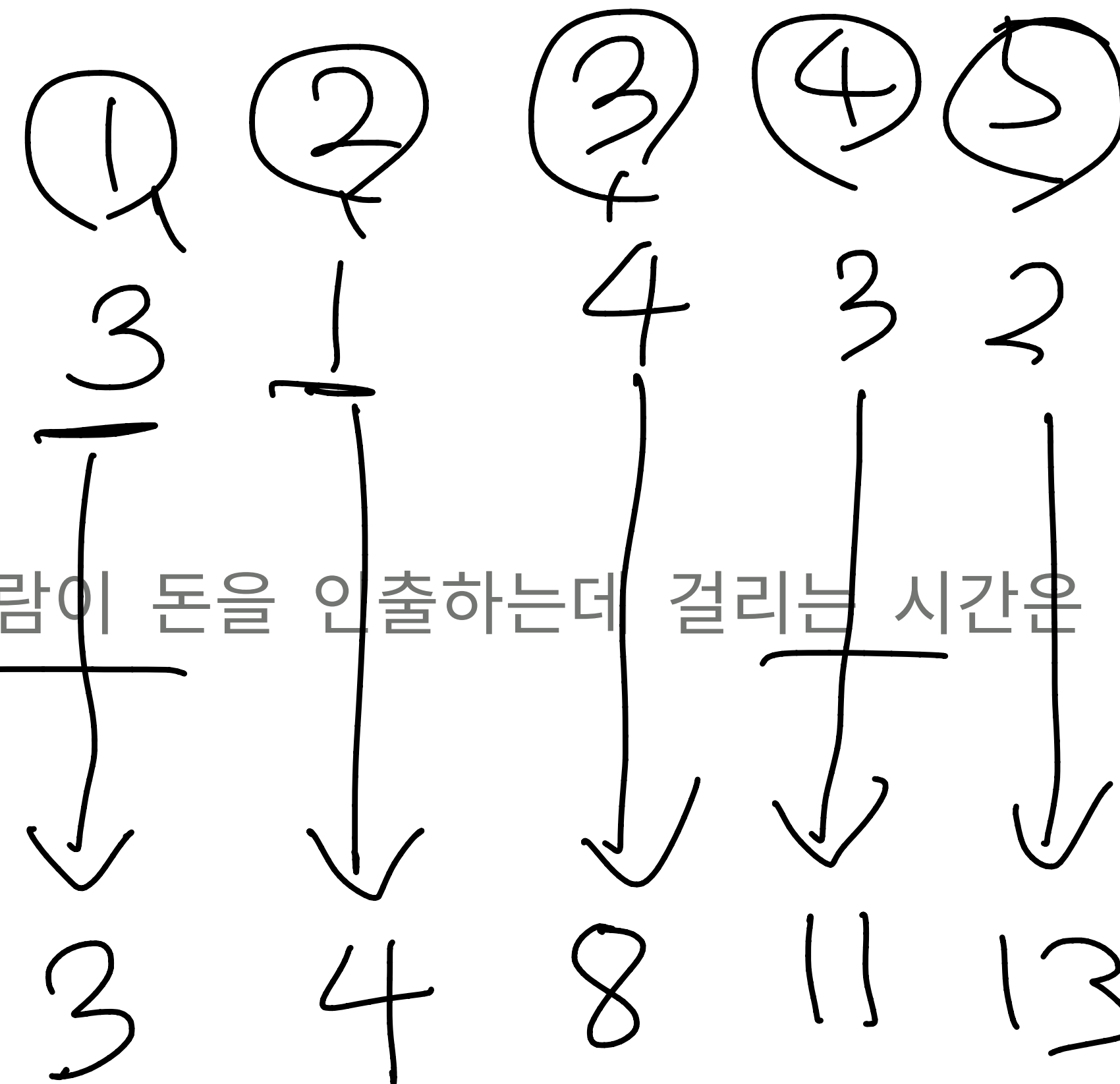
# ATM

<https://www.acmicpc.net/problem/11399>

합/최소

- 인하은행에는 ATM이 1대밖에 없다
- 지금 이 ATM앞에 N명의 사람들이 줄을 서있다
- 사람은 1번부터 N번까지 번호가 매겨져 있으며, i번 사람이 돈을 인출하는데 걸리는 시간은  $P_i$ 분이다.

ATM



$$3 + 4 + 8 + 11 + 13 = 15 + 24 = 39$$

$P_i$  분  
합/최소  
ATM ① ④ ⑤ ③ ②

3 3 2 4 1

3 6 8 12 13

합:  $17 + 25 = 42$

# ATM

<https://www.acmicpc.net/problem/11399>

- 사람들이 줄을 서는 순서에 따라서, 돈을 인출하는데 필요한 시간의 합이 달라지게 된다
- 예를 들어, 총 5명이 있고,  $P1 = 3$ ,  $P2 = 1$ ,  $P3 = 4$ ,  $P4 = 3$ ,  $P5 = 2$  인 경우를 생각해보자.
- $[1, 2, 3, 4, 5]$  순서로 줄을 선다면, 1번 사람은 3분만에 돈을 뽑을 수 있다.
- 2번 사람은 1번 사람이 돈을 뽑을 때 까지 기다려야 하기 때문에,  $3+1 = 4$ 분이 걸리게 된다.
- 3번 사람은 1번, 2번 사람이 돈을 뽑을 때까지 기다려야 하기 때문에, 총  $3+1+4 = 8$ 분이 필요하게 된다.
- 4번 사람은  $3+1+4+3 = 11$ 분, 5번 사람은  $3+1+4+3+2 = 13$ 분이 걸리게 된다.
- 이 경우에 각 사람이 돈을 인출하는데 필요한 시간의 합은  $3+4+8+11+13 = 39$ 분이 된다.

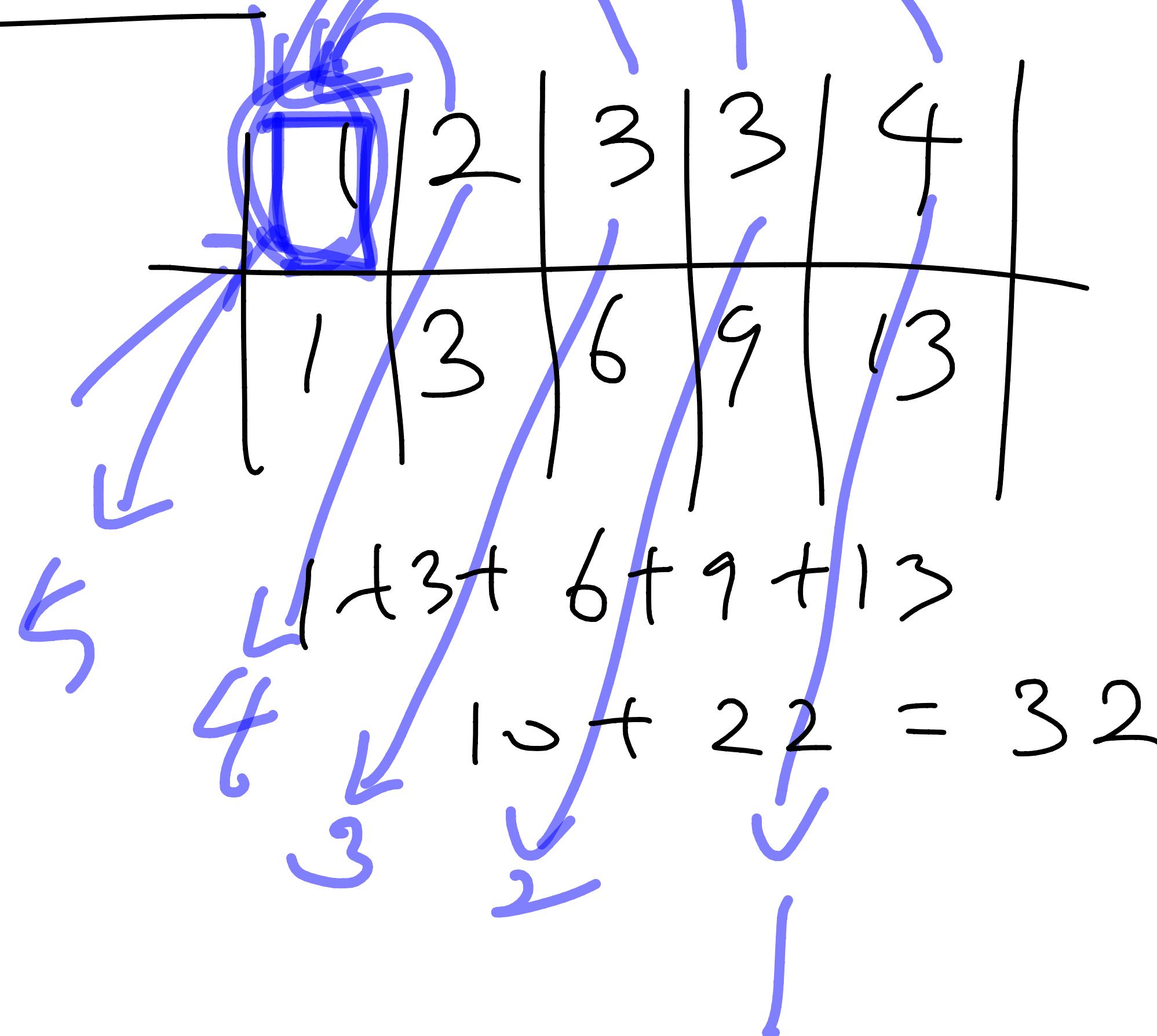
# ATM

23

<https://www.acmicpc.net/problem/11399>

3 1 4 3 2

- 줄을 서 있는 사람의 수  $N$ 과 각 사람이 돈을 인출하는데 걸리는 시간  $P_i$ 가 주어졌을 때, 각 사람이 돈을 인출하는데 필요한 시간의 합의 최소값을 구하는 문제



# ATM

<https://www.acmicpc.net/problem/11399>

- 기다리는 시간이 짧은 사람부터 ATM을 인출하는 것이 좋다.





# ATM

오름차순

25

<https://www.acmicpc.net/problem/11399>

$$p_1 \leq p_2 \leq p_3 \dots \leq p_n$$

- $p_1, p_2, p_3, \dots, p_n$
- $p_1 \leq p_2 \leq p_3 \leq \dots \leq p_n$  이 정답이라고 가정
- 총 돈을 인출하는데 걸리는 시간의 합

- $S = p_1 + (p_1 + p_2) + (p_1 + p_2 + p_3) + \dots + (p_1 + p_2 + \dots + p_n)$

- $S = n \times p_1 + (n-1) \times p_2 + \dots + p_n$

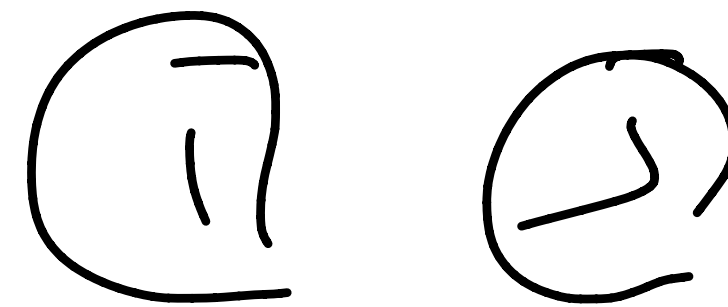
# ATM

<https://www.acmicpc.net/problem/11399>

26

작은 리 먼 안됨

- $p_1 \leq p_2 \leq p_3 \leq \dots \leq p_n$  이 정답이라면, 중간에  $i < j$ 인  $p_i$ 와  $p_j$ 의 순서를 바꿨을 때, 더 커져야 한다.

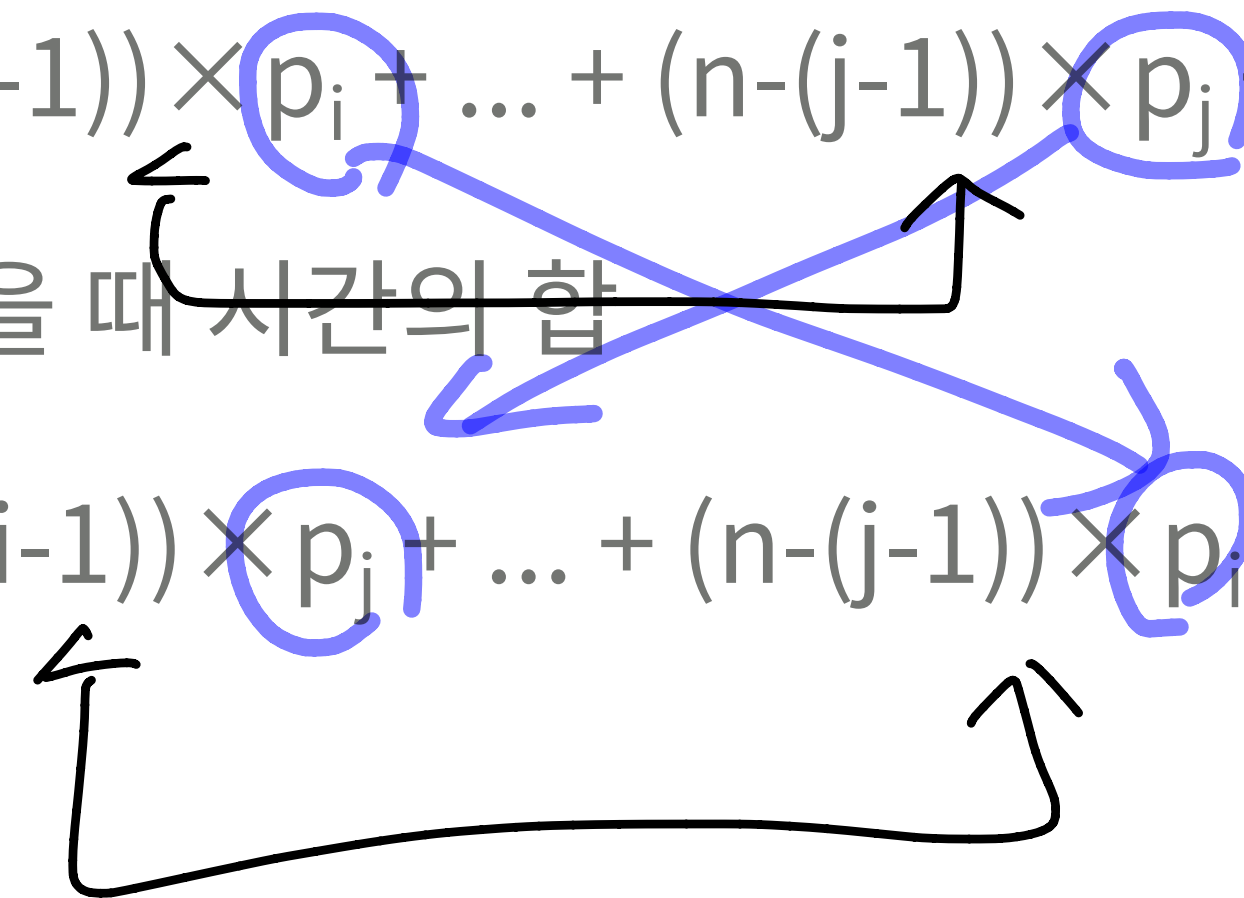


거대한 것

- 총 돈을 인출하는데 걸리는 시간의 합
- $S = n \times p_1 + \dots + (n - (i - 1)) \times p_i + \dots + (n - (j - 1)) \times p_j + \dots + p_n$

$p_i$ 와  $p_j$ 의 순서를 바꿨을 때 시간의 합

- $S' = n \times p_1 + \dots + (n - (i - 1)) \times p_j + \dots + (n - (j - 1)) \times p_i + \dots + p_n$



$$S \leq S'$$
$$S - S' \leq 0$$

# ATM

<https://www.acmicpc.net/problem/11399>

- $S$ 가 정답이기 때문에,  $S \leq S'$ 를 만족해야 한다.
- $S \leq S'$  라면  $S - S' \leq 0$  이 되어야 한다.

# ATM

<https://www.acmicpc.net/problem/11399>

28

$$[i < j]$$

$$\underline{P_i \leq P_j}$$

- $S - S' = (n-i+1) \times p_i + (n-j+1) \times p_j - (n-i+1) \times p_j - (n-j+1) \times p_i$
- $= (n-i+1-n+j-1)p_i + (n-j+1-n+i-1)p_j$
- $= (-i+j)p_i + (-j+i)p_j$
- $= (j-i)p_i + (i-j)p_j$
- $= -(i-j)p_i + (j-j)p_j$
- $= (i-j)(p_j - p_i)$

$$\textcircled{S - S'} = \underbrace{(i-j)}_{i-j < 0} \times \underbrace{(p_j - p_i)}_{\geq 0}$$

$$S - S' \leq 0 \\ \Rightarrow S \leq S'$$

# ATM

<https://www.acmicpc.net/problem/11399>

- $S - S' = (i - j)(p_j - p_i)$
- 여기서  $i < j$ 이기 때문에,  $i - j < 0$  이다.
- $p_i \leq p_j$  이기 때문에  $0 \leq p_j - p_i$  이다.
- $i - j$ 는 음수이고,  $p_j - p_i$ 는 양수 또는 0이기 때문에
- $S - S' \leq 0$ 이다.
- 따라서 오름차순이 정답이다.

# ATM

<https://www.acmicpc.net/problem/11399>

- 소스: <http://codeplus.codes/edd3b32198f043ae9d8d773d4dd68281>

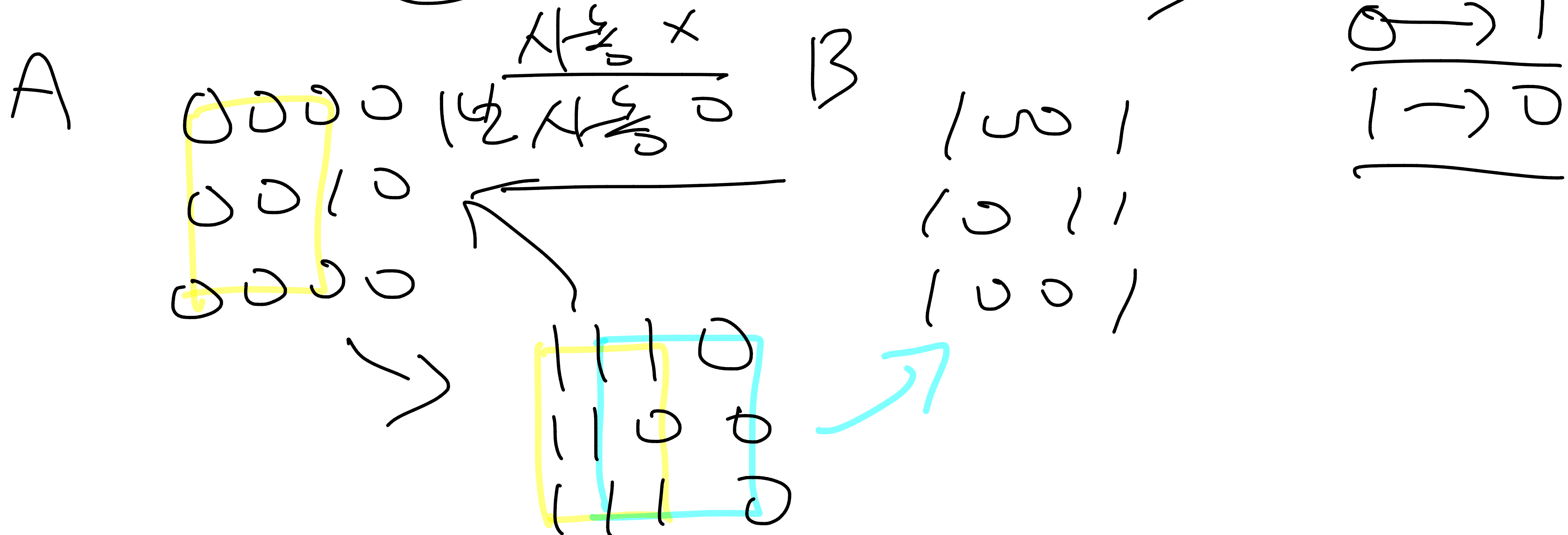
# 행렬

244

0 → 1 → 0

<https://www.acmicpc.net/problem/1080>

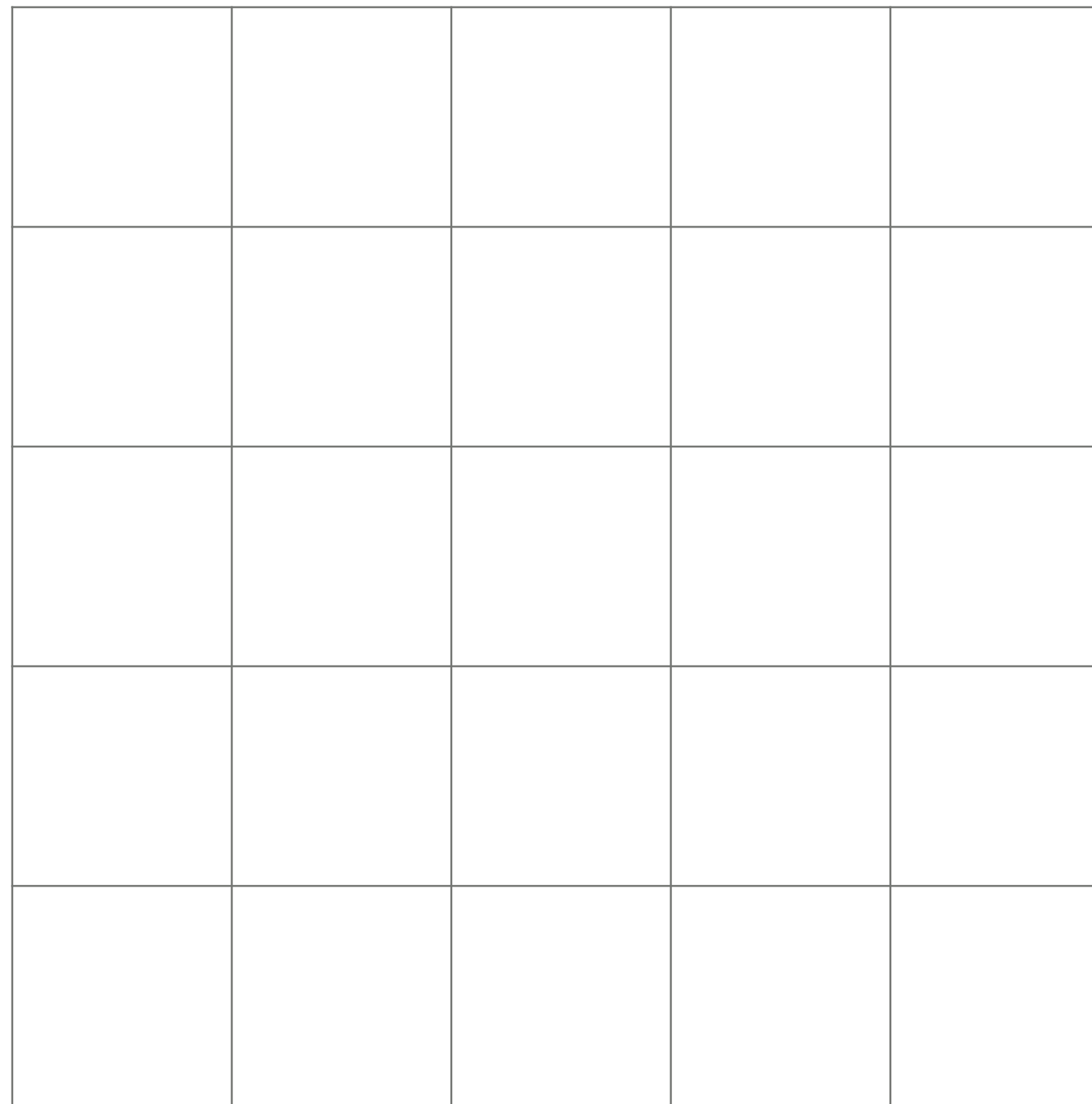
- 0과 1로만 이루어진 행렬 A와 행렬 B가 있다. 이 때, 행렬 A를 행렬 B로 바꾸는데 필요한 연산의 횟수의 최소값을 구하는 문제
- 행렬을 변환하는 연산은 어떤 3\*3크기의 부분 행렬에 있는 모든 원소를 뒤집는 것이다. (0 → 1, 1 → 0)



# 행렬

<https://www.acmicpc.net/problem/1080>

- 5×5인 경우



$N, M$

구함:  $(N-2) \times (M-2)$

$2^{N \times M}$

$2500$   
 $2$



# 행렬

<https://www.acmicpc.net/problem/1080>

- $5 \times 5$ 인 경우

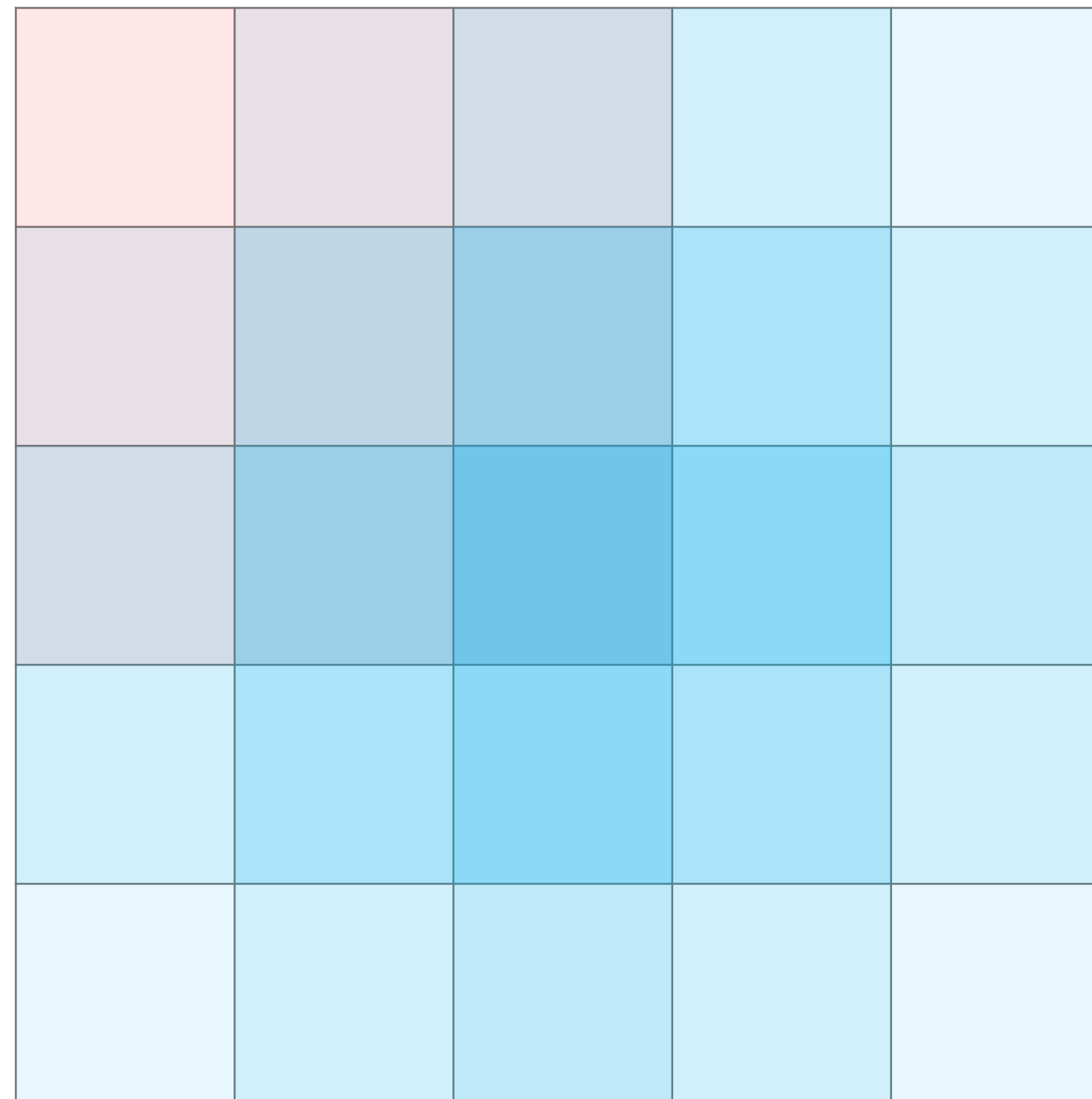
D  
12

1	2	3	2	1
2	3	4	3	2
3	4	5	4	3
2	3	4	3	2
1	2	3	2	1

# 행렬

<https://www.acmicpc.net/problem/1080>

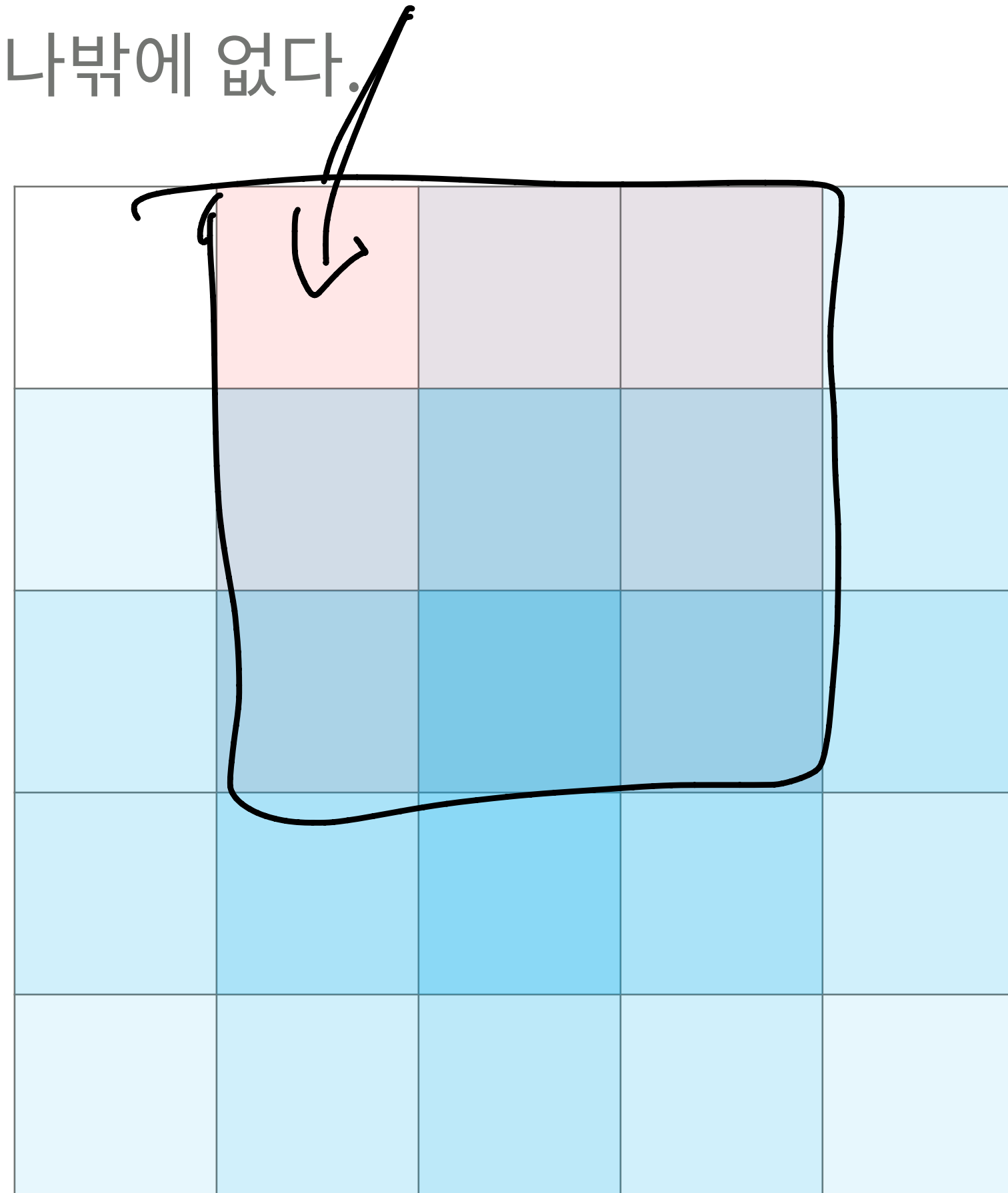
- $(0, 0)$ 을 바꿀 수 있는 방법은 하나밖에 없다.



# 행렬

<https://www.acmicpc.net/problem/1080>

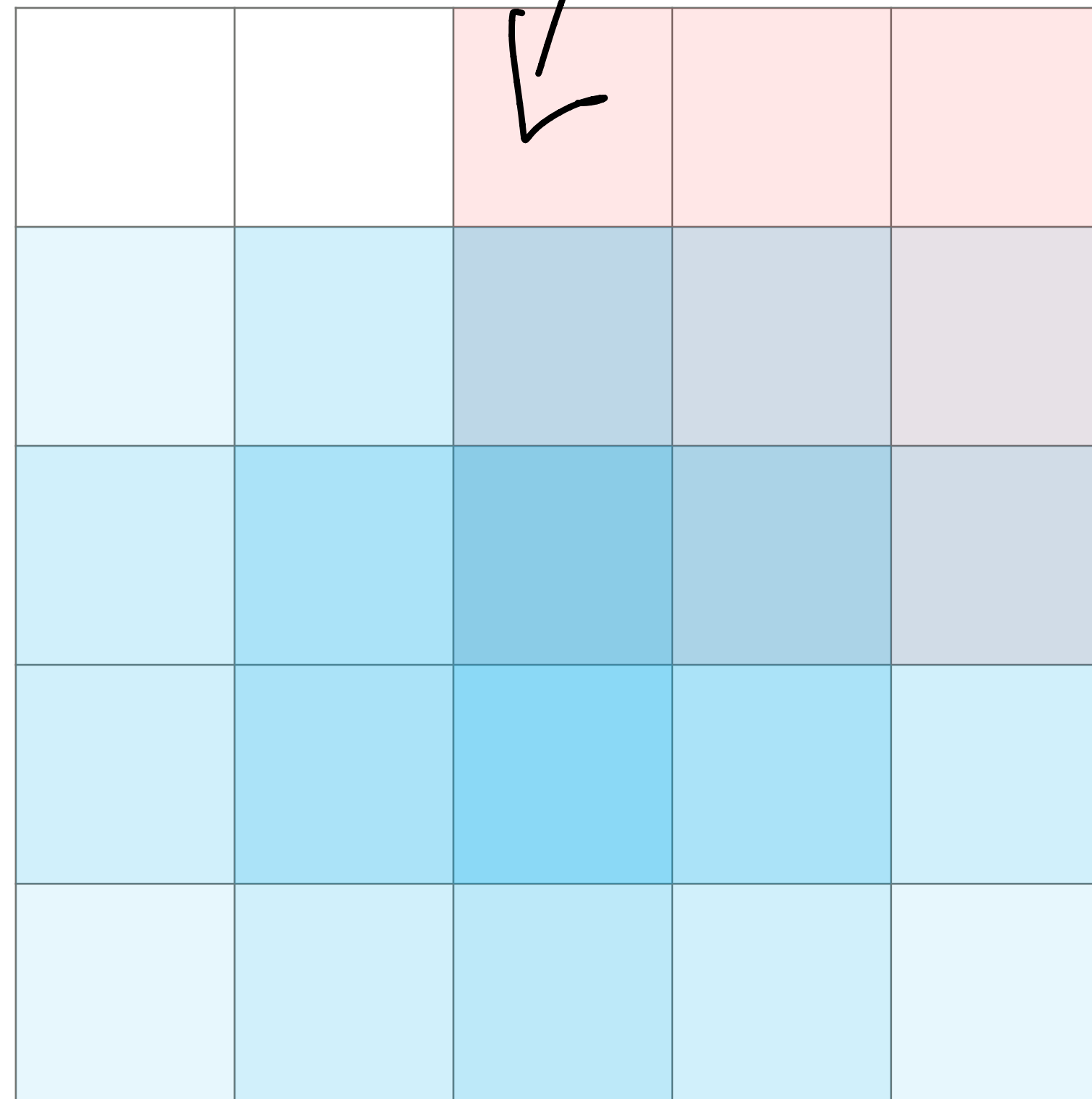
- $(0, 1)$ 을 바꿀 수 있는 방법은 하나밖에 없다.



# 행렬

<https://www.acmicpc.net/problem/1080>

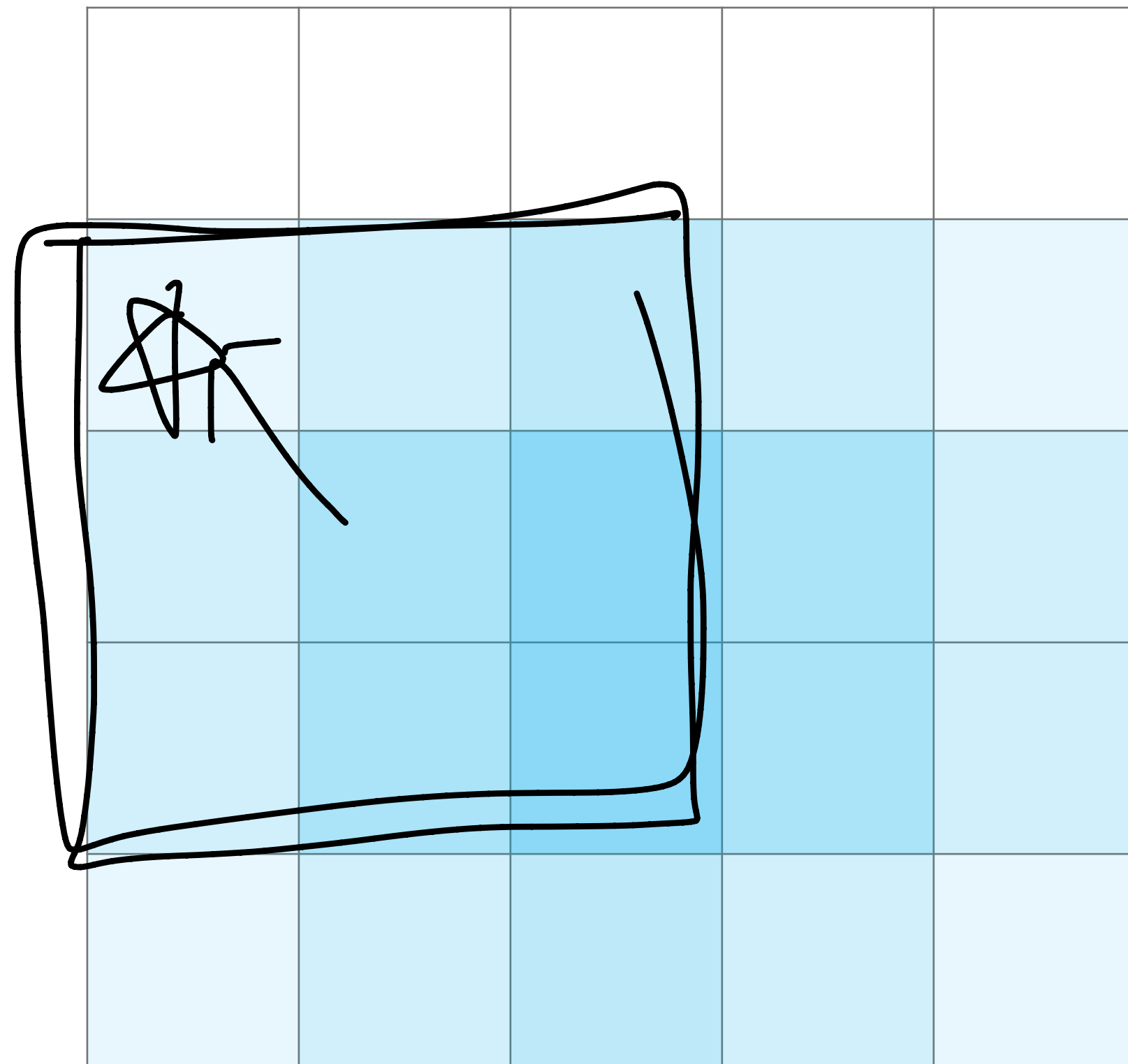
- $(0, 2)$ 를 바꿀 수 있는 방법은 하나밖에 없다.



# 행렬

<https://www.acmicpc.net/problem/1080>

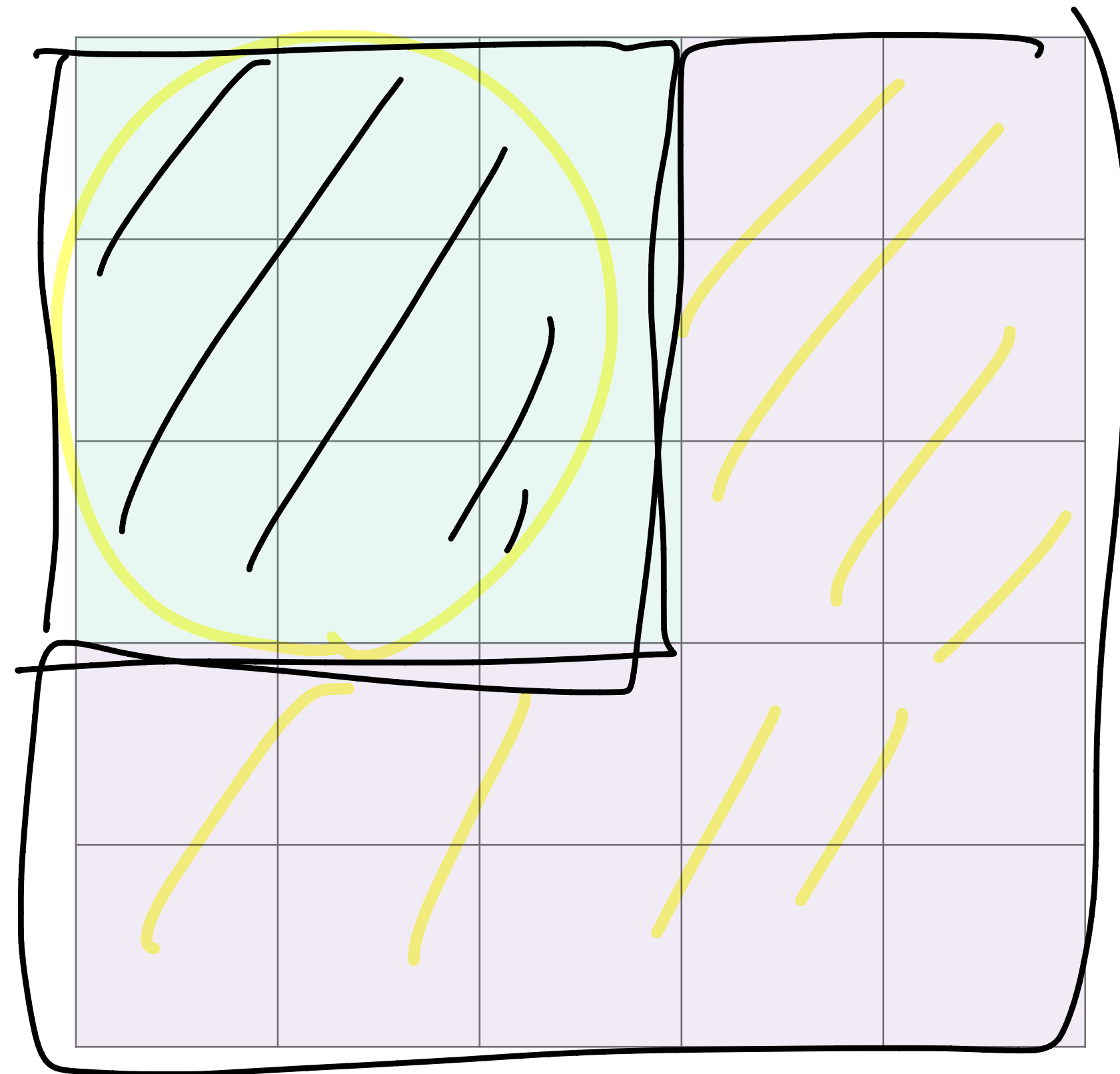
- 이런 식으로 계속 반복한다면



# 행렬

<https://www.acmicpc.net/problem/1080>

- 아래 초록색 칸에 들어있는 값은 A와 B가 같다.
- 이제 나머지 칸에 대해서
- 같은지 조사하면 된다.



# 행렬

<https://www.acmicpc.net/problem/1080>

- 소스: <http://codeplus.codes/07c27d5a3798450caffc7a2ec8427897>

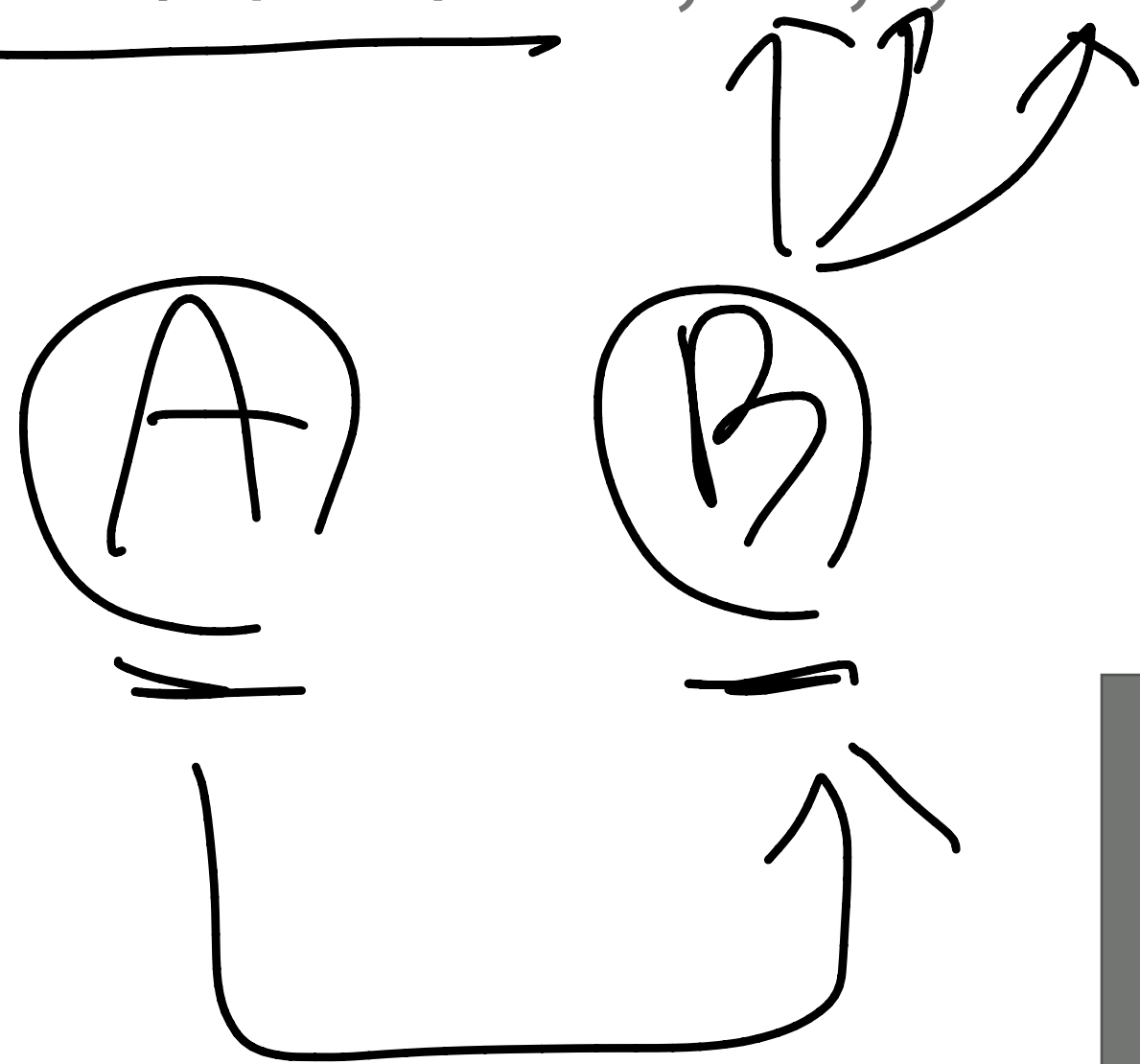
# 전구와 스위치

<https://www.acmicpc.net/problem/2138>

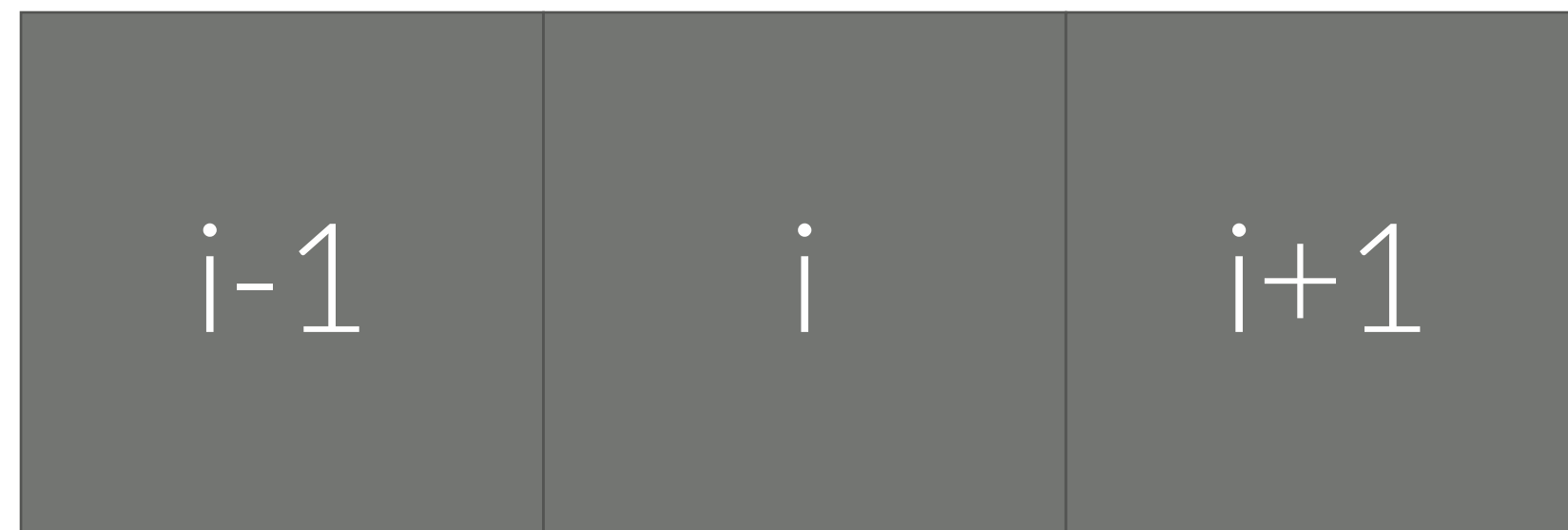
40

223 (N)

- i번 스위치를 누르면, i-1, i, i+1번 전구의 상태가 바뀐다



켜 → 꺼  
꺼 → 켜





# 전구와 스위치

<https://www.acmicpc.net/problem/2138>

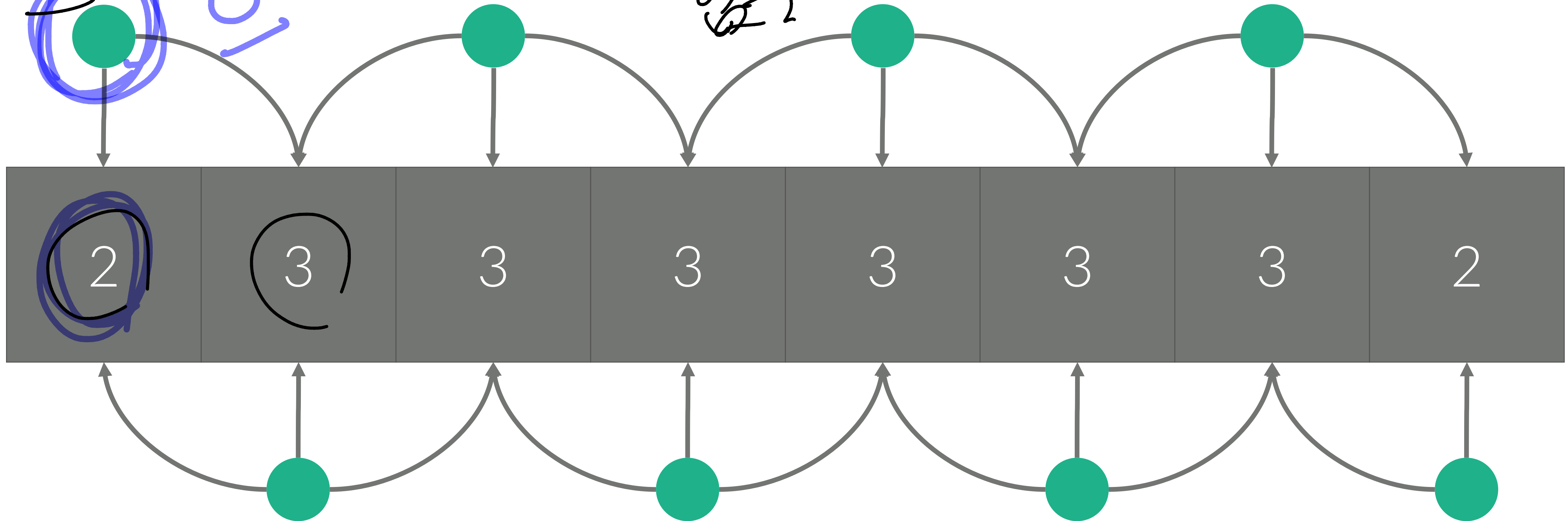
41

전구의 상태를  
바꿀 수 있는  
스위치의 개수

- $i$ 번 스위치를 누르면,  $i-1, i, i+1$ 번 전구의 상태가 바뀐다

스위치  
스위치  
1개 밖에 없는 전구가  
없어

전구

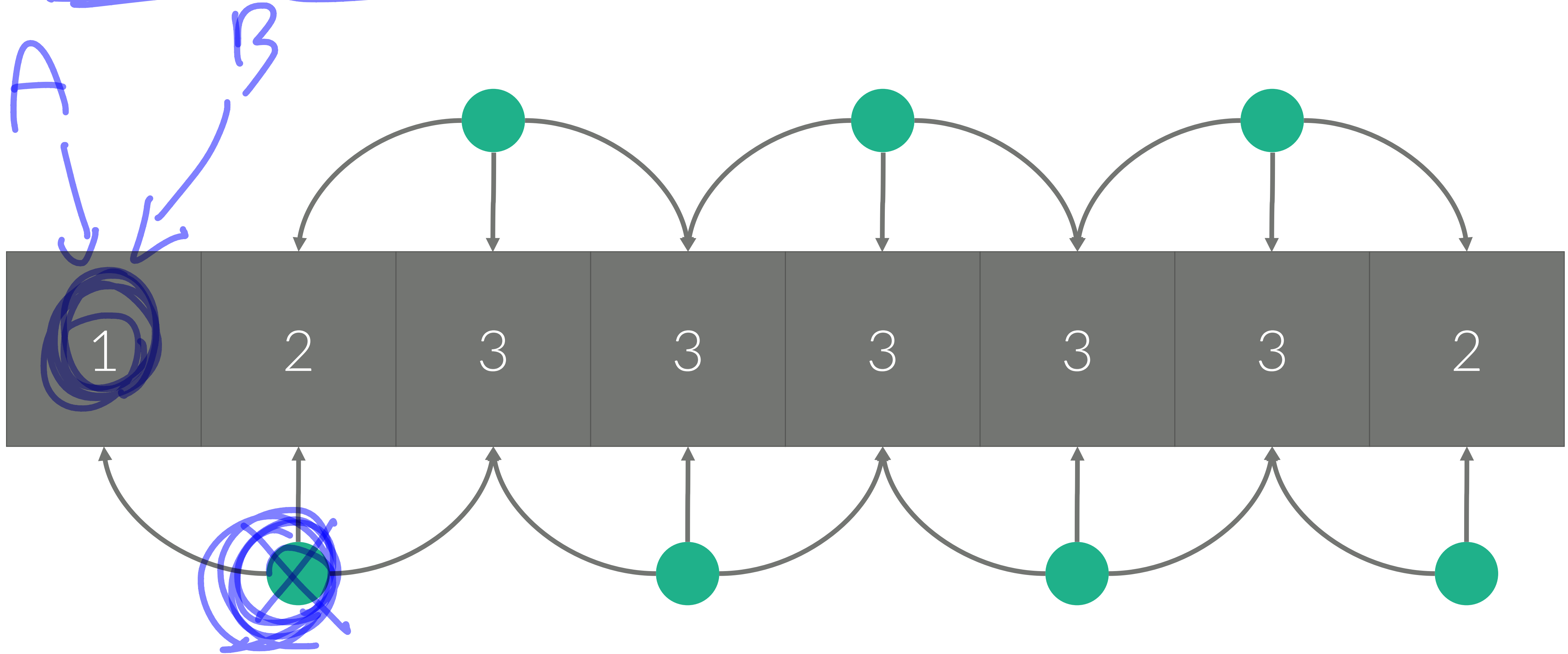


# 전구와 스위치

<https://www.acmicpc.net/problem/2138>

1개 스위치

- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다

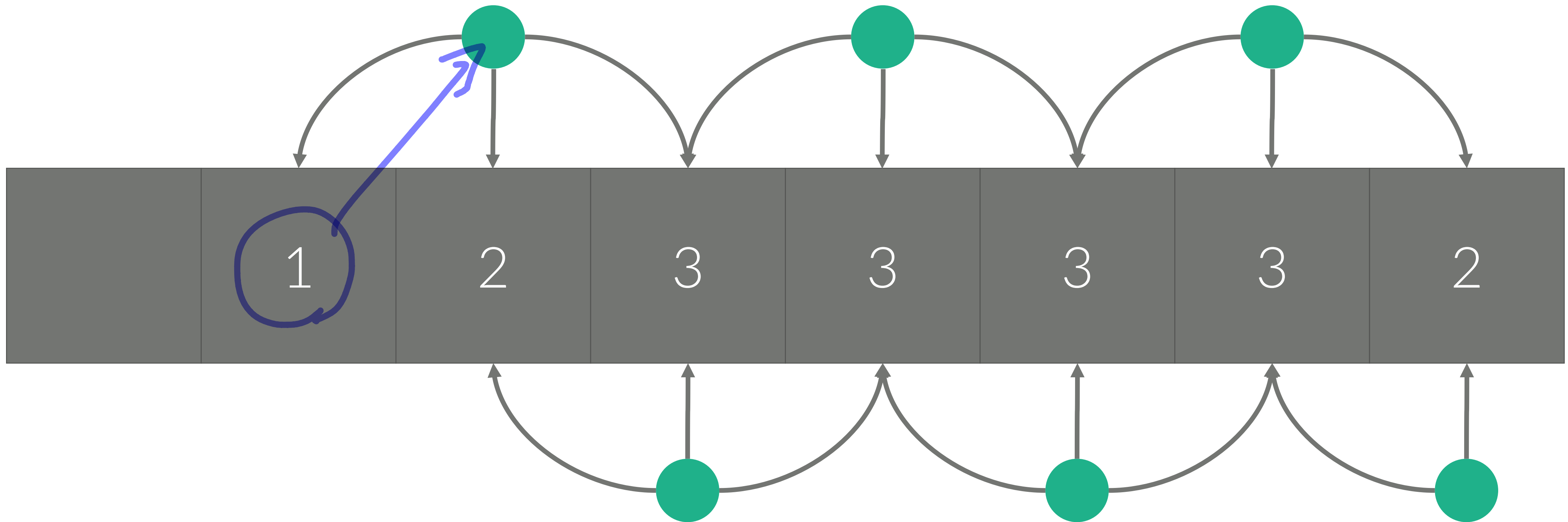


# 전구와 스위치

43

<https://www.acmicpc.net/problem/2138>

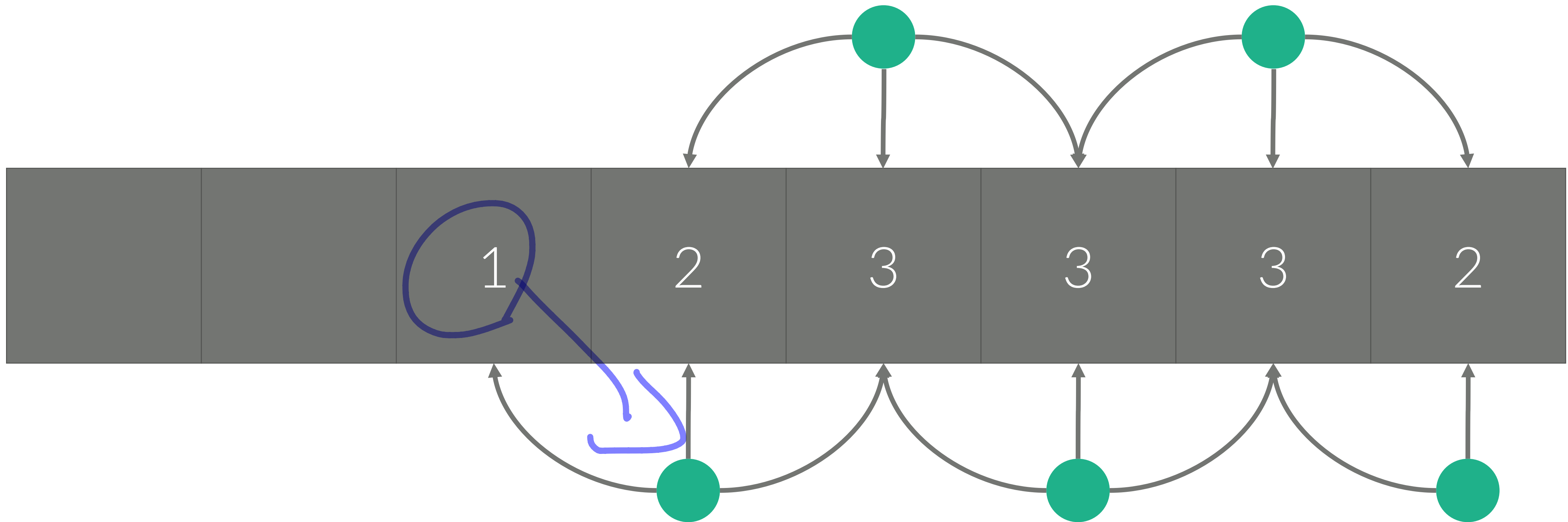
- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다



# 전구와 스위치

<https://www.acmicpc.net/problem/2138>

- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다

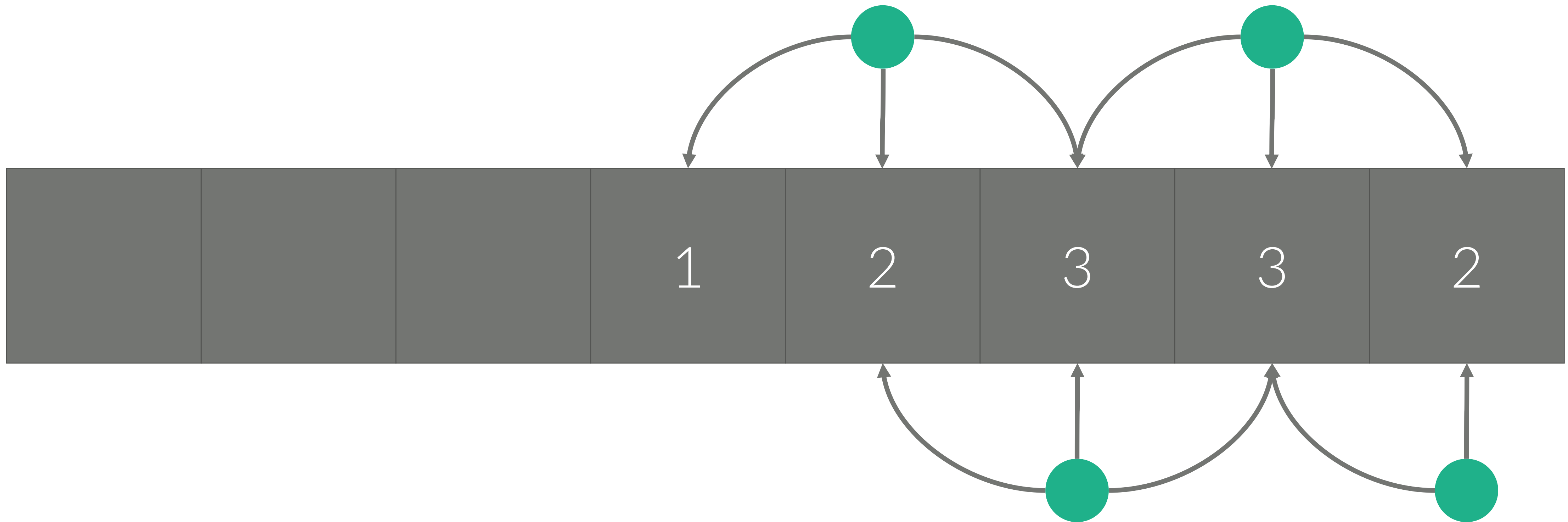


# 전구와 스위치

45

<https://www.acmicpc.net/problem/2138>

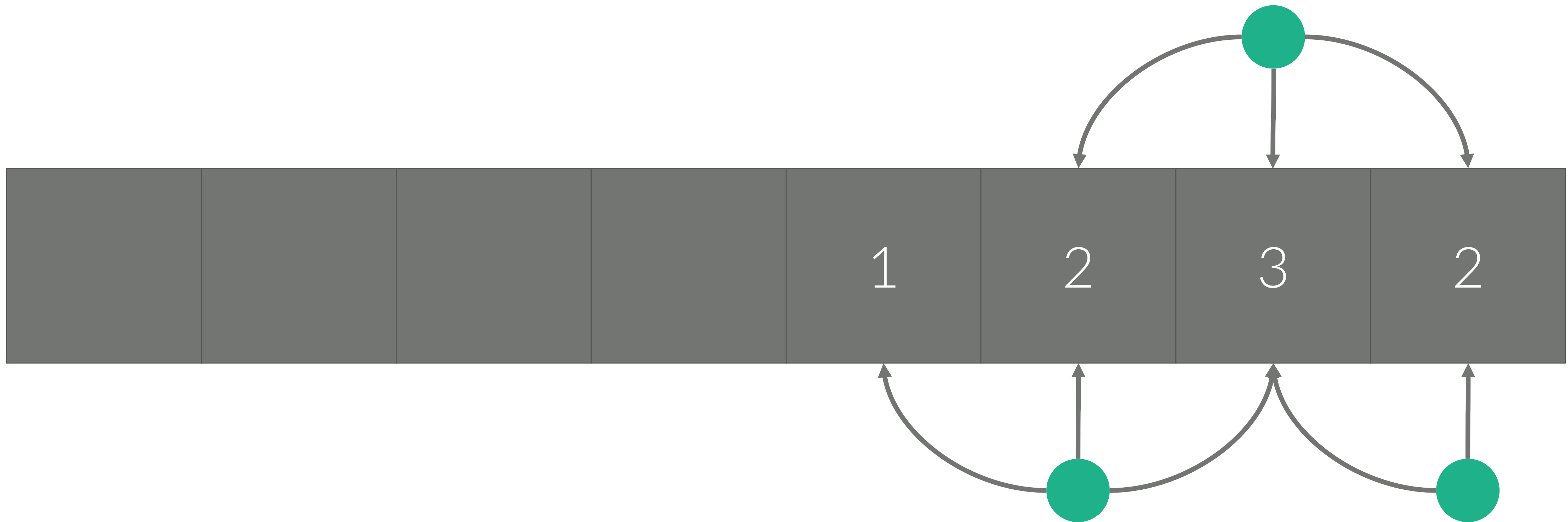
- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다



# 전구와 스위치

<https://www.acmicpc.net/problem/2138>

- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다



# 전구와 스위치

47

<https://www.acmicpc.net/problem/2138>

- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다



# 전구와 스위치

48

<https://www.acmicpc.net/problem/2138>

- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다

$$A = B$$

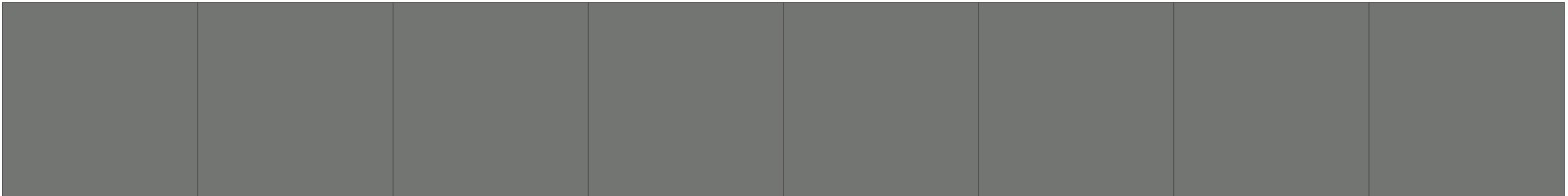




# 전구와 스위치

<https://www.acmicpc.net/problem/2138>

- 1번 스위치를 어떻게 누를지 결정하면, 각각의 칸을 바꿀 수 있는 방법이 한 개가 된다



# 전구와 스위치

50

<https://www.acmicpc.net/problem/2138>

- 소스: <http://codeplus.codes/285d1843d7d54250965051db65762372>

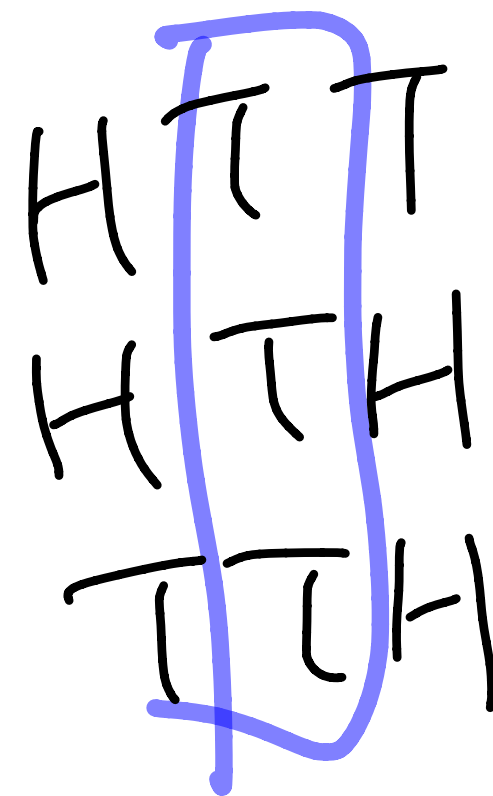
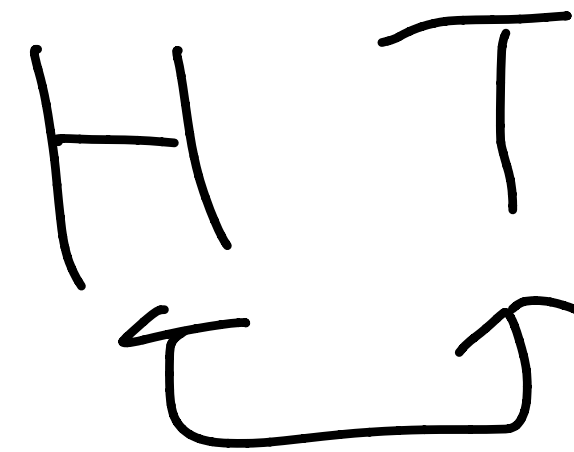
# 동전 뒤집기

$$N \leq 20$$

51

<https://www.acmicpc.net/problem/1285>

- $N^2$ 개의 동전이  $N$ 행  $N$ 열로 놓여져 있다.
- 임의의 한 행, 한 열에 놓인  $N$ 개의 동전을 뒤집는 작업을 수행할 수 있다
- 동전을 적절히 뒤집어서 T의 최소 개수를 구하는 문제

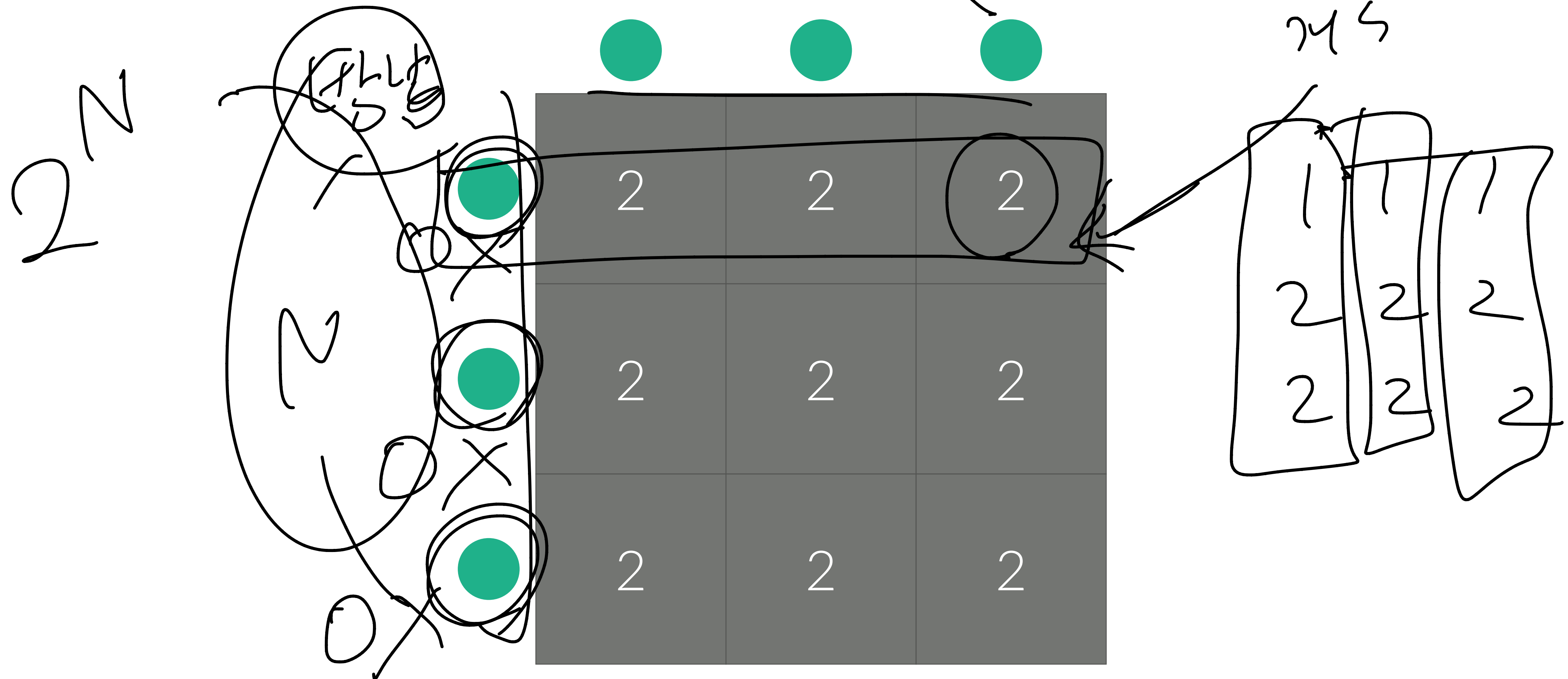


# 동전 뒤집기

<https://www.acmicpc.net/problem/1285>

52

- 각각의 칸을 바꿀 수 있는 방법은 모두 두 가지이다



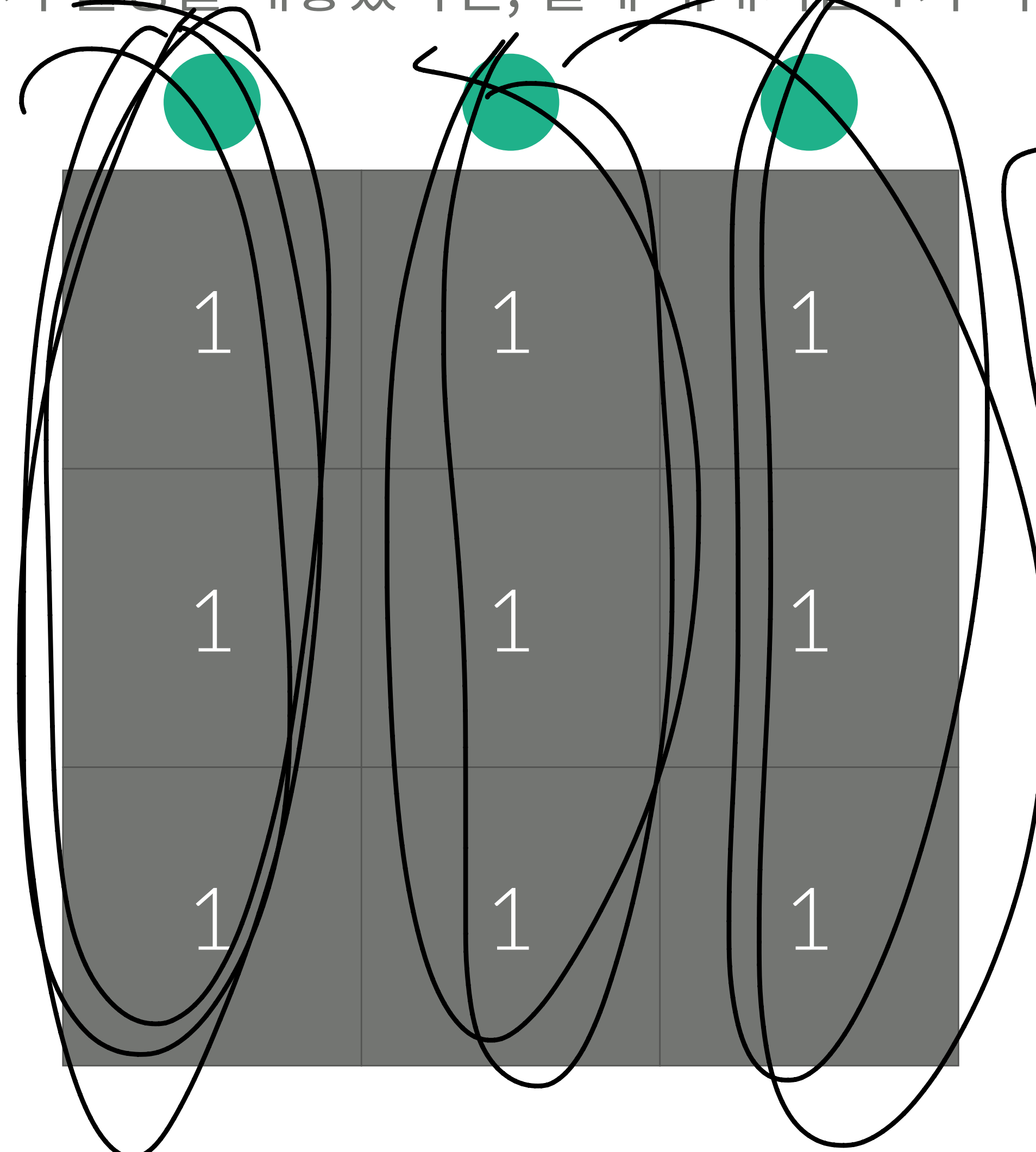
# 동전 뒤집기

<https://www.acmicpc.net/problem/1285>

53

- 한 행에 대해서 어떻게 돌릴지 결정을 해놓았다면, 열에 대해서는 T가 적은 쪽이 결정되어 버린다

1의 개수  
7의 개수



# 동전 뒤집기

<https://www.acmicpc.net/problem/1285>

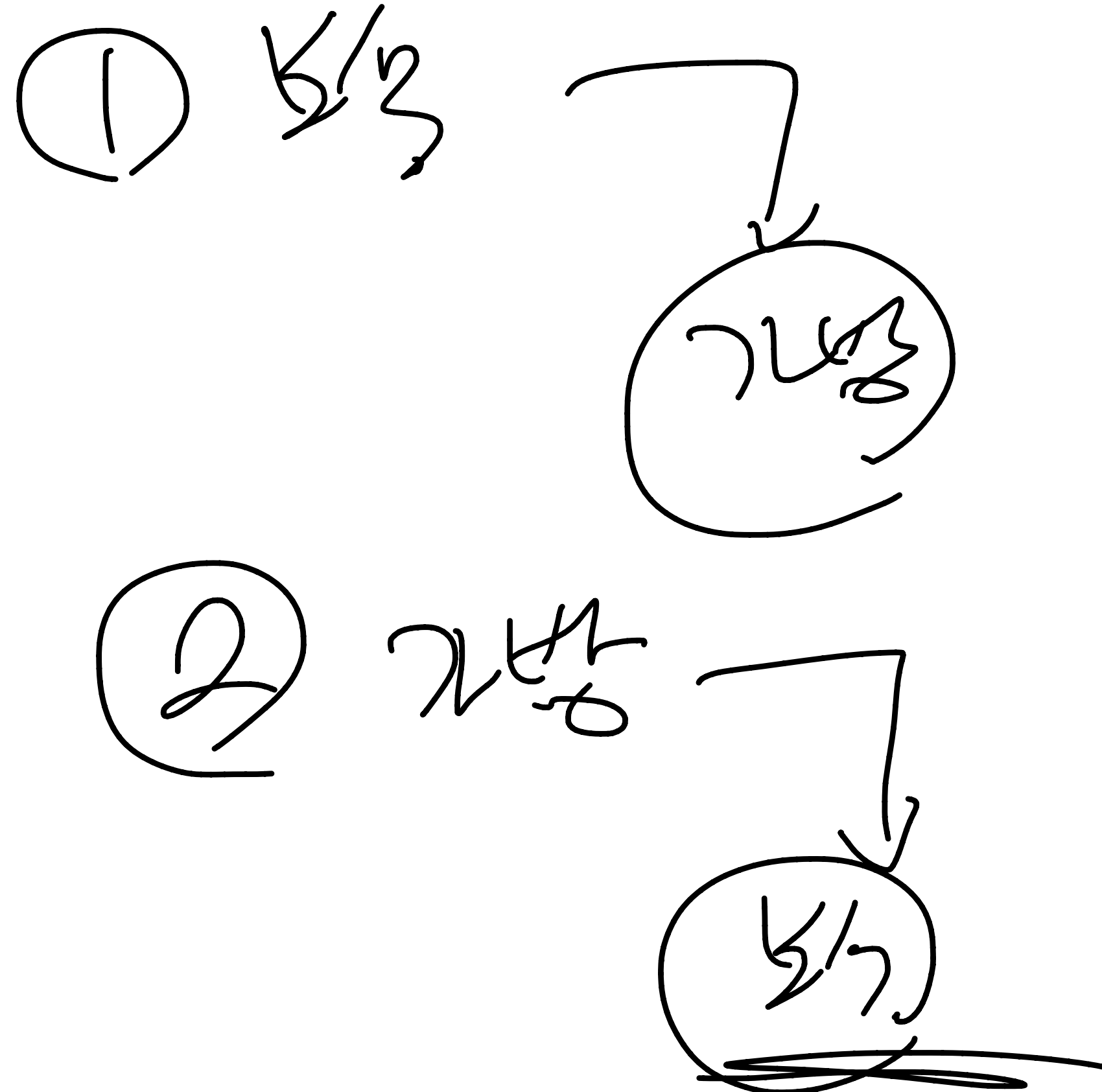
- 소스: <http://codeplus.codes/a1e857d377a041d4bdfd11ffb99876c4>

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 보석이  $N$ 개
- 각 보석은 무게  $M[i]$ 와 가격  $V[i]$ 를 가지고 있음
- 가방은  $K$ 개
- 가방에 담을 수 있는 최대 무게  $C[i]$
- 가방에는 보석 1개만 넣을 수 있음
- $1 \leq N, K \leq 300,000, 0 \leq M[i], V[i] \leq 1,000,000, 1 \leq C[i] \leq 100,000,000$

- 가방에 담을 수 있는 보석의 최대 가격 구하는 문제



# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 각각의 보석이 어떤 가방에 들어갈 수 있는지 조사해보자
- 최대 가격을 구하는 것이고, 각 가방에는 보석이 1개만 들어갈 수 있기 때문에
- 가격이 큰 보석을 최대한 많이 가져가는 것이 좋다.
- 보석 (무게, 가격) = (2, 99), (5, 65), (1, 23)
- 가방 = 2, 10
- (2, 99)는 어디에 들어가는 것이 좋을까?



# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 각각의 보석이 어떤 가방에 들어갈 수 있는지 조사해보자
- 최대 가격을 구하는 것이고, 각 가방에는 보석이 1개만 들어갈 수 있기 때문에
- 가격이 큰 보석을 최대한 많이 가져가는 것이 좋다.
- 보석 (무게, 가격) = (2, 99), (5, 65), (1, 23)
- 가방 = 2, 10
- (2, 99)는 어디에 들어가는 것이 좋을까? 최대 무게가 2인 가방에 들어가는 것이 좋다.

NK

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 각각의 보석이 어떤 가방에 들어갈 수 있는지 조사해보자
- 최대 가격을 구하는 것이고, 각 가방에는 보석이 1개만 들어갈 수 있기 때문에
- 가격이 큰 보석을 최대한 많이 가져가는 것이 좋다.
- 보석 (무게, 가격) ~~(2, 99)~~, (5, 65), (1, 23)
- 가방 ~~= 2, 10~~
- (1, 65)는 어디에 들어가는 것이 좋을까? 최대 무게가 10인 가방에 들어가는 것이 좋다.
- 만약, (2, 99)를 최대 무게가 10인 가방에 넣었다면, (5, 65)를 가져갈 수 없다.

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 가격이 높은 보석부터 차례대로 각 보석을 담을 수 있는 가방 중  $C[i]$ 가 가장 작은 가방에 넣는다.
- 이를 구현하기 위해서 다음을 효율적으로 할 수 있는 자료구조가 필요하다.

1. 어떤 수  $x$ 보다 큰 숫자 중에 가장 작은 수를 찾는다. (Lower Bound)
2. 수를 지운다.

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 가격이 높은 보석부터 차례대로 각 보석을 담을 수 있는 가방 중  $C[i]$  가 가장 작은 가방에 넣는다.
- 이를 구현하기 위해서 다음을 효율적으로 할 수 있는 자료구조가 필요하다.
  1. 어떤 수  $x$ 보다 큰 숫자 중에 가장 작은 수를 찾는다. (Lower Bound)
  2. 수를 지운다.
- 배열을 사용한다면 1은  $O(\lg K)$ 에 할 수 있지만, 2를  $O(K)$ 에 할 수 있다.

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

61

C++ multiset  
Java TreeSet

- 가격이 높은 보석부터 차례대로 각 보석을 담을 수 있는 가방 중  $C[i]$  가 가장 작은 가방에 넣는다.
- 이를 구현하기 위해서 다음을 효율적으로 할 수 있는 자료구조가 필요하다.

1. 어떤 수  $x$ 보다 큰 숫자 중에 가장 작은 수를 찾는다. (Lower Bound)
2. 수를 지운다.

- BST를 사용한다면 1과 2를  $O(\lg K)$ 에 할 수 있다.

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

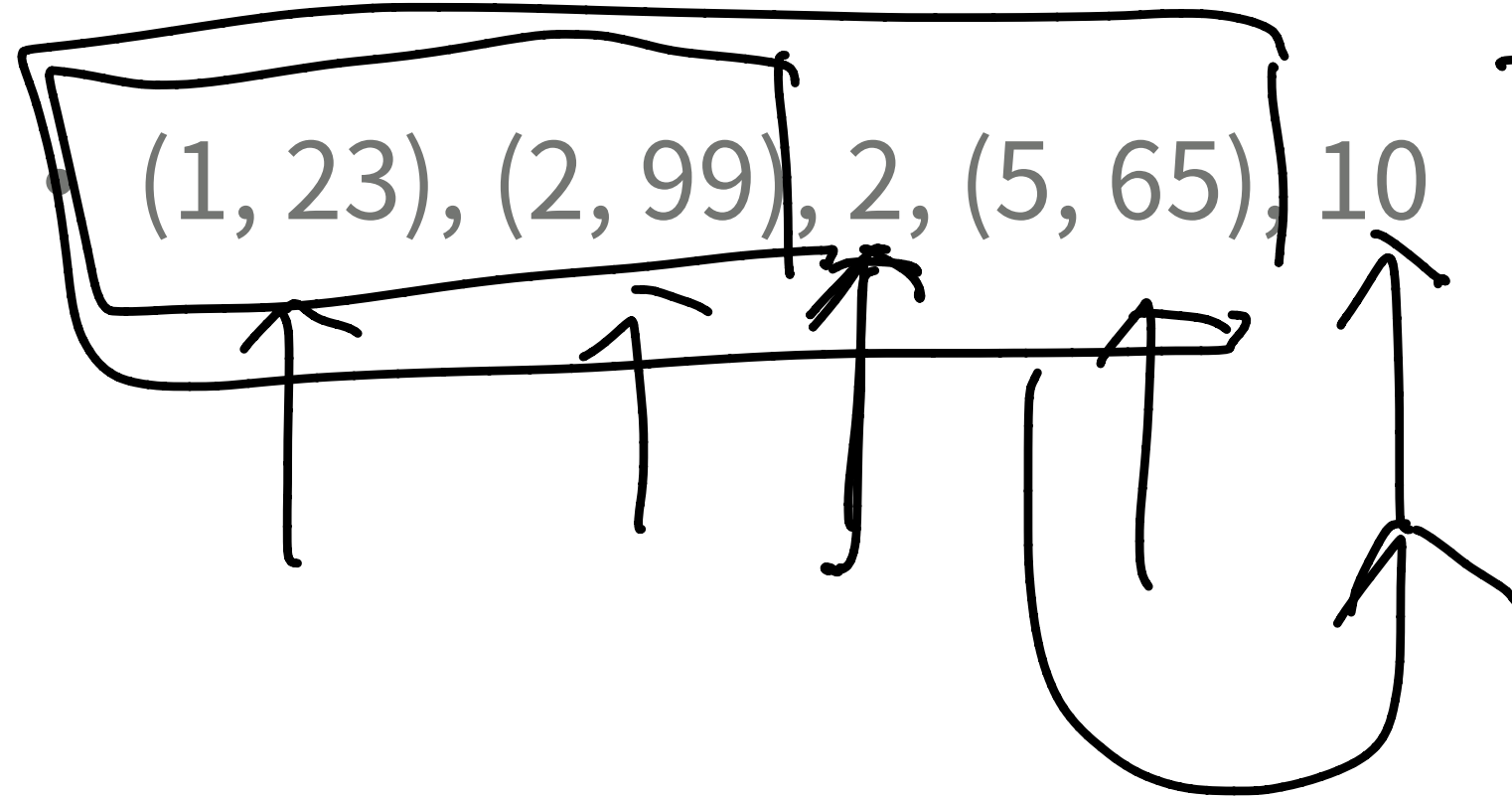
- 소스: <http://codeplus.codes/fee56f20eae043e9b7381b22cf588bd0>
- Java는 multiset이 없어서 TreeMap을 이용해서 비슷하게 구현했다.

# 보석 도둑

<https://www.acmicpc.net/problem/1202>



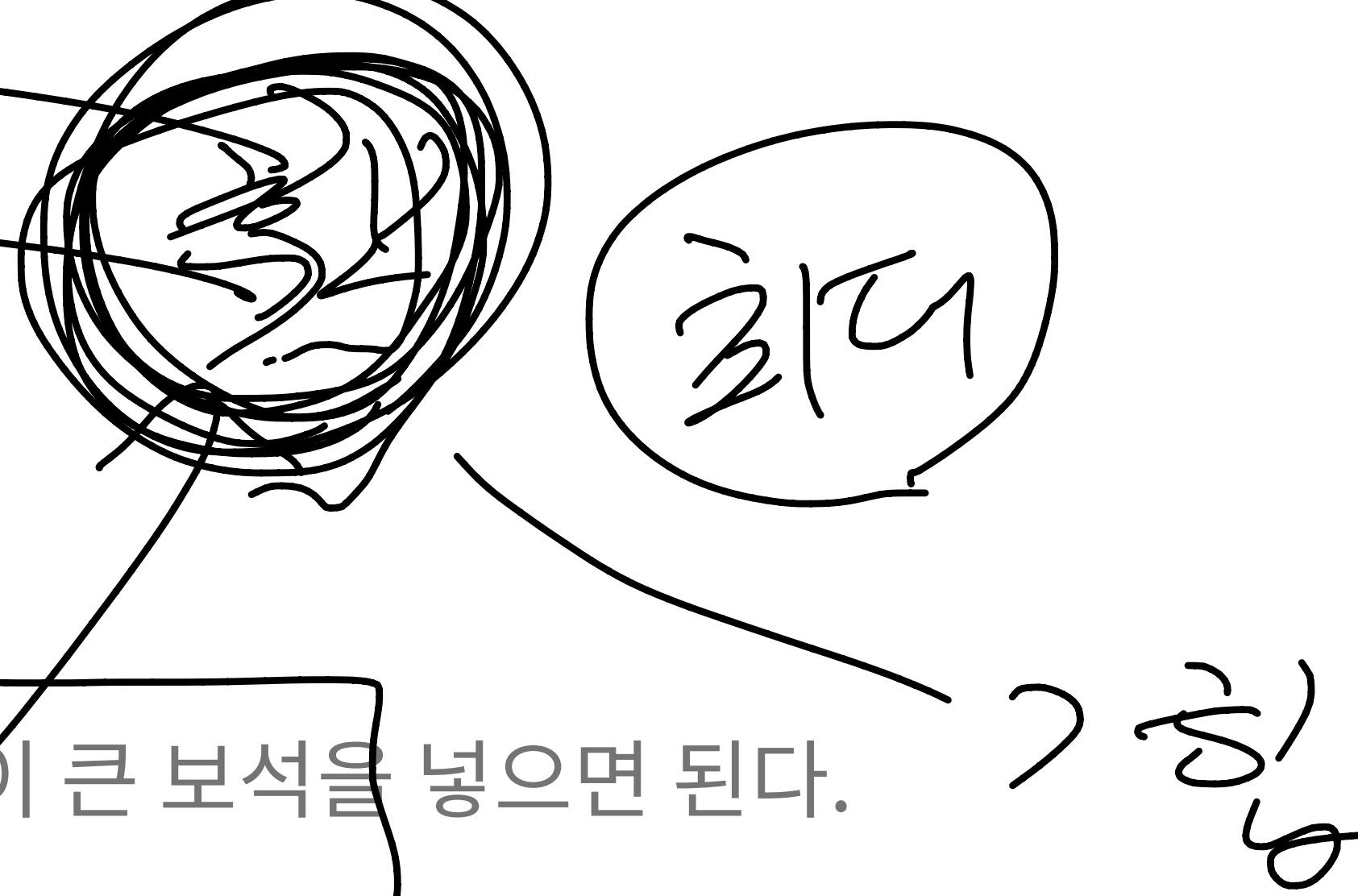
- 각각의 가방에 들어갈 수 있는 가장 가격이 높은 보석을 조사해보자
- 보석 (무게, 가격) = (2, 99), (5, 65), (1, 23)
- 가방 = 2, 10
- 보석과 가방을 하나로 합치고 무게를 기준으로 오름차순 정렬하자



# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- (1, 23), (2, 99), 2, (5, 65), 10
- 가방이 나올때마다 앞에있는 보석 중에서 가장 가격이 큰 보석을 넣으면 된다.
- (1, 23), (2, 99), 2, (5, 65), 10
- 2의 경우에는 (2, 99)를 넣는 것이 제일 좋다. (앞에 있는것 중 가장 가격이 큰 보석)
- (1, 23), ~~(2, 99)~~, 2, (5, 65), 10
- 10의 경우에는 (5, 65)를 넣는 것이 제일 좋다.





# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 무게가 증가하는 순으로 정렬했기 때문에, 앞에 있는 모든 보석은 다 가방에 들어갈 수 있다.
- 보석의 경우에는 가격을 H에 저장하고
- 가방의 경우에는 H에서 가장 큰 값을 찾고, 제거한다.
- 이를 효율적으로 할 수 있는 자료구조 H는 무엇일까?

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 무게가 증가하는 순으로 정렬했기 때문에, 앞에 있는 모든 보석은 다 가방에 들어갈 수 있다.
- 보석의 경우에는 가격을 H에 저장하고
- 가방의 경우에는 H에서 가장 큰 값을 찾고, 제거한다.
- 이를 효율적으로 할 수 있는 자료구조 H는 무엇일까? 최대 힙

# 보석 도둑

<https://www.acmicpc.net/problem/1202>

- 소스: <http://codeplus.codes/c9ab89b8b8474dbeb7a1b3b344f6a36b>

# 순회강연

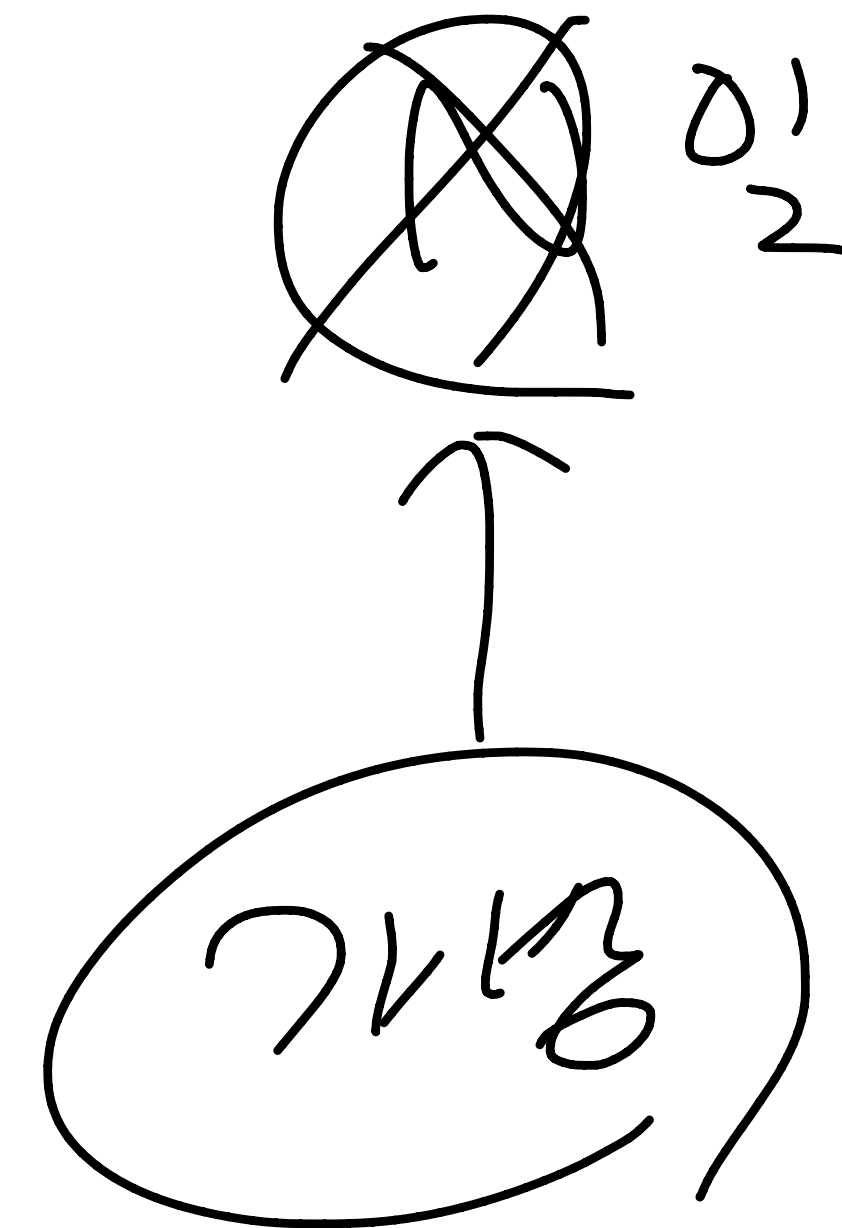
<https://www.acmicpc.net/problem/2109>

- N개의 대학에서 강연 요청을 했다.
- 강연 요청은 두 개의 (d, p)이고, d일 안에 와서 강연을 하면 p원의 강연료를 준다는 의미이다.
- 하루에 최대 한 곳에서만 강연을 할 수 있다고 가정했을 때
- 최대 수익을 구하는 문제

1 2 3 - -

보통의 문제  $\leq$  기생의 문제

기생의 문제  $\leq$  보통의 문제

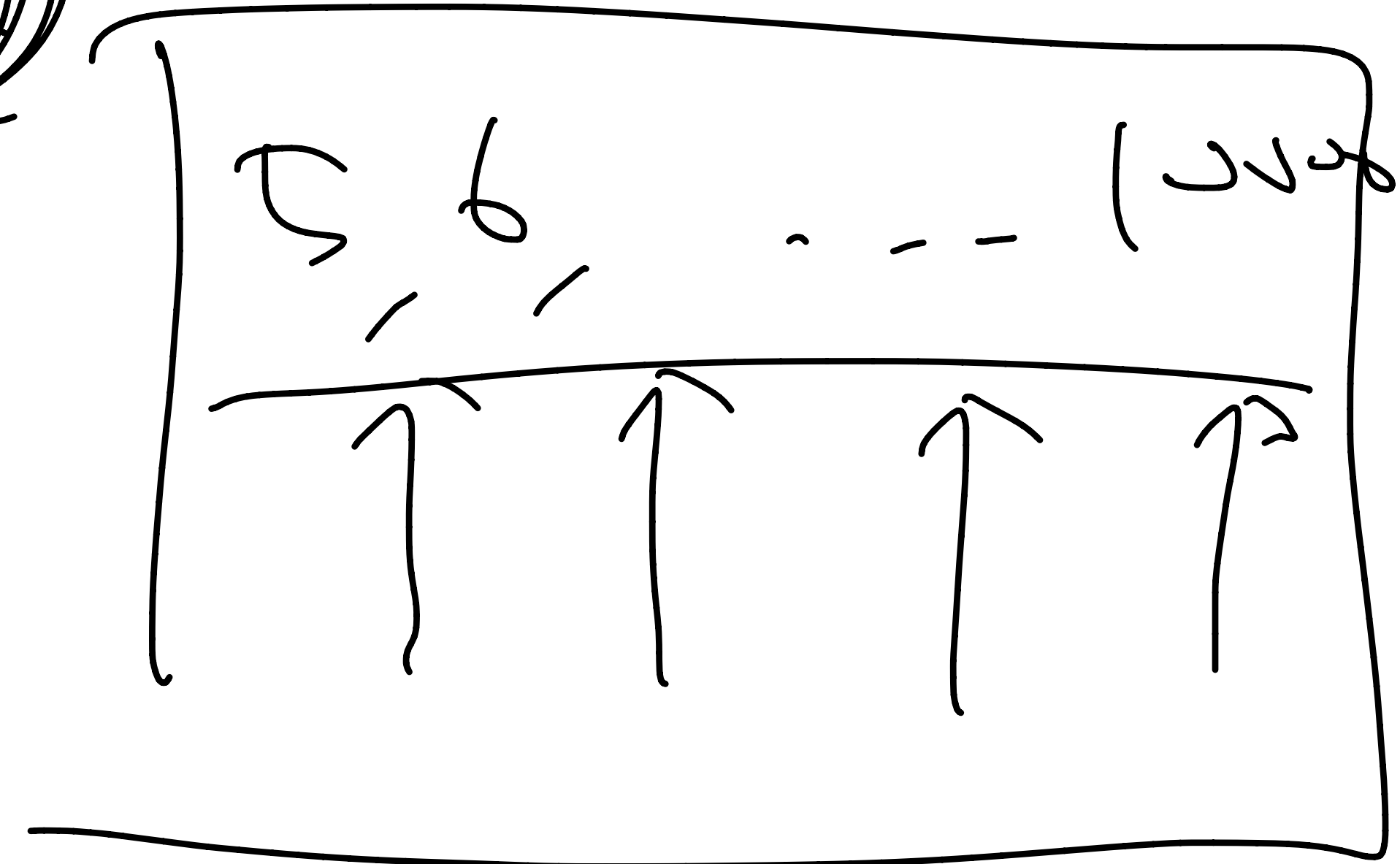


# 순회강연

<https://www.acmicpc.net/problem/2109>

69

- 이 문제는 보석 도둑 문제와 매우 비슷한 문제이다.
- 보석 도둑의 보석 = 순회강연에서 강연
- 보석 도둑의 가방 = 순회강연에서 하루
- 보석 도둑의 조건 "가방에 보석 1개를 넣을 수 있다"는
- 순회강연의 조건 "하루에 1개의 강연만 할 수 있다"와 같다.
- 보석 도둑: 보석의 무게가  $w$ 라면, 이 보석은  $c$ 의 값이  $w$ 보다 크거나 같은 가방에 들어갈 수 있다.
- 순회강연: 강연  $(d, p)$ 는  $d$ 보다 작거나 같은 날에만 강연을 할 수 있다.
- 가방을 기준으로 보석 도둑 문제를 푼 방법에서 무게를 내림차순으로 정렬하고 문제를 해결하면 순회강연 문제와 같은 의미를 갖는다.



# 순회강연

(p, d)

3일 | 4일 —

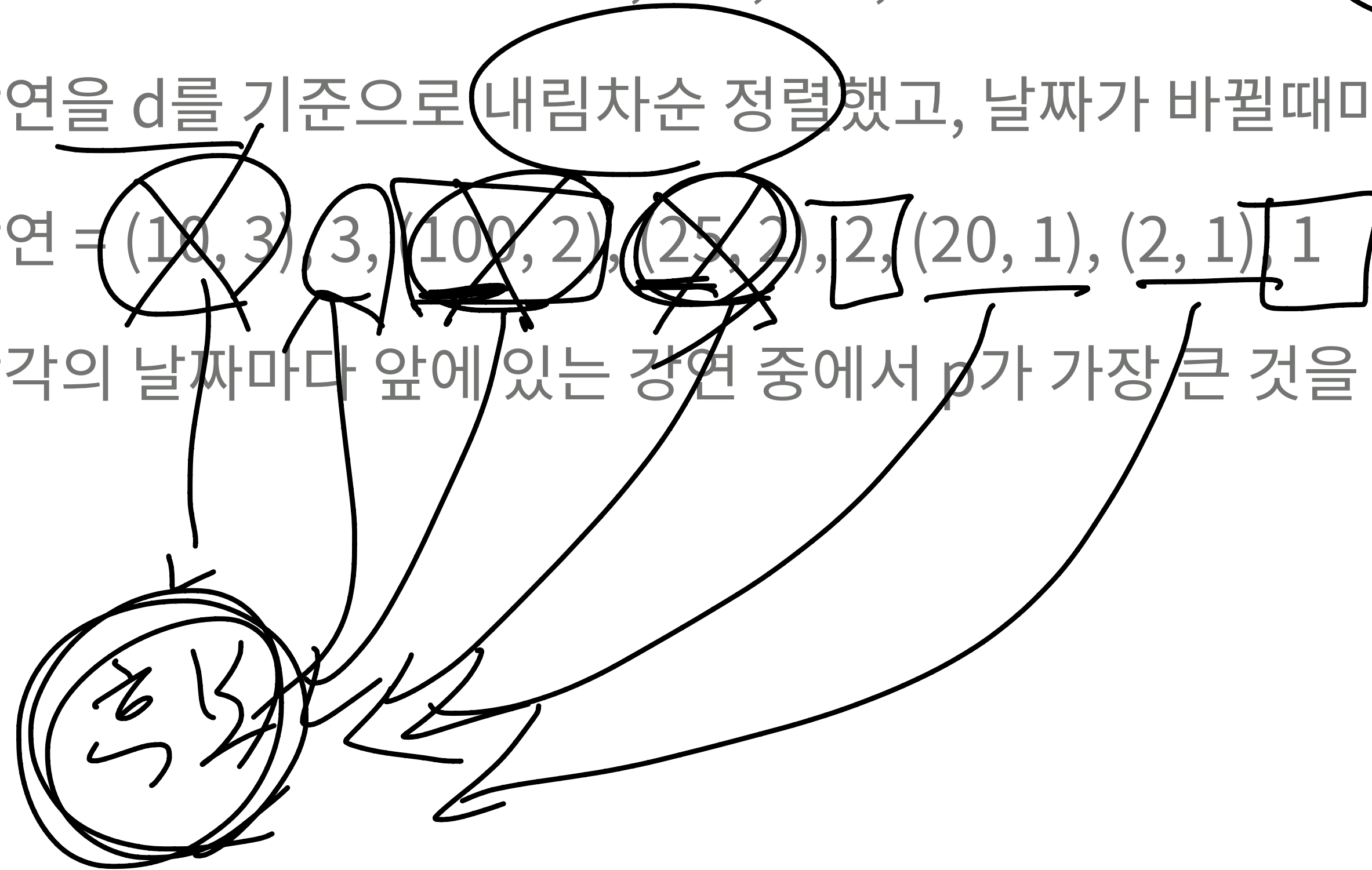
2일

1일

70

<https://www.acmicpc.net/problem/2109>

- 강연 = (20, 1), (2, 1), (10, 3), (100, 2), (25, 2)
- d의 최댓값은 3이기 때문에, 3일, 2일, 1일에 어떤 강연을 해야하는지 조사하려고 한다.
- 강연을 d를 기준으로 내림차순 정렬했고, 날짜가 바뀔때마다 날짜를 추가했다.
- 강연 = ~~(10, 3)~~, 3, ~~(100, 2)~~, ~~(25, 2)~~, 2, (20, 1), (2, 1), 1
- 각각의 날짜마다 앞에 있는 강연 중에서 p가 가장 큰 것을 고르면 된다.



# 순회강연

<https://www.acmicpc.net/problem/2109>

- 강연 = (10, 3), **3**, (100, 2), (25, 2), 2, (20, 1), (2, 1), 1
- 3일에는 (10, 3)을 하는 것이 좋다.
- 강연 = ~~(10, 3)~~, 3, (100, 2), (25, 2), **2**, (20, 1), (2, 1), 1
- 2일에는 (100, 2)를 하는 것이 좋다.
- 강연 = ~~(10, 3)~~, 3, ~~(100, 2)~~, (25, 2), 2, (20, 1), (2, 1), **1**
- 1일에는 (25, 2)를 하는 것이 좋다.

# 순회강연

72

<https://www.acmicpc.net/problem/2109>

- 강연을 최대 힙을 이용해서 유지하면 된다.



# 순회강연

73

<https://www.acmicpc.net/problem/2109>

- 소스: <http://codeplus.codes/c95cc7ebc53244edbd5d7b5e442d5a97>

# 가장 긴 증가하는 부분 수열 3

<https://www.acmicpc.net/problem/12738>

LIS

- 수열 A가 주어졌을 때, 가장 긴 증가하는 부분 수열을 구하는 프로그램을 작성하시오
- 예를 들어, 수열  $A = \{10, 20, 10, 30, 20, 50\}$  인 경우에 가장 긴 증가하는 부분 수열은  $A = \{10, 20, 10, 30, 20, 50\}$  이고, 길이는 4이다

# 가장 긴 증가하는 부분 수열 2

75

<https://www.acmicpc.net/problem/12015>

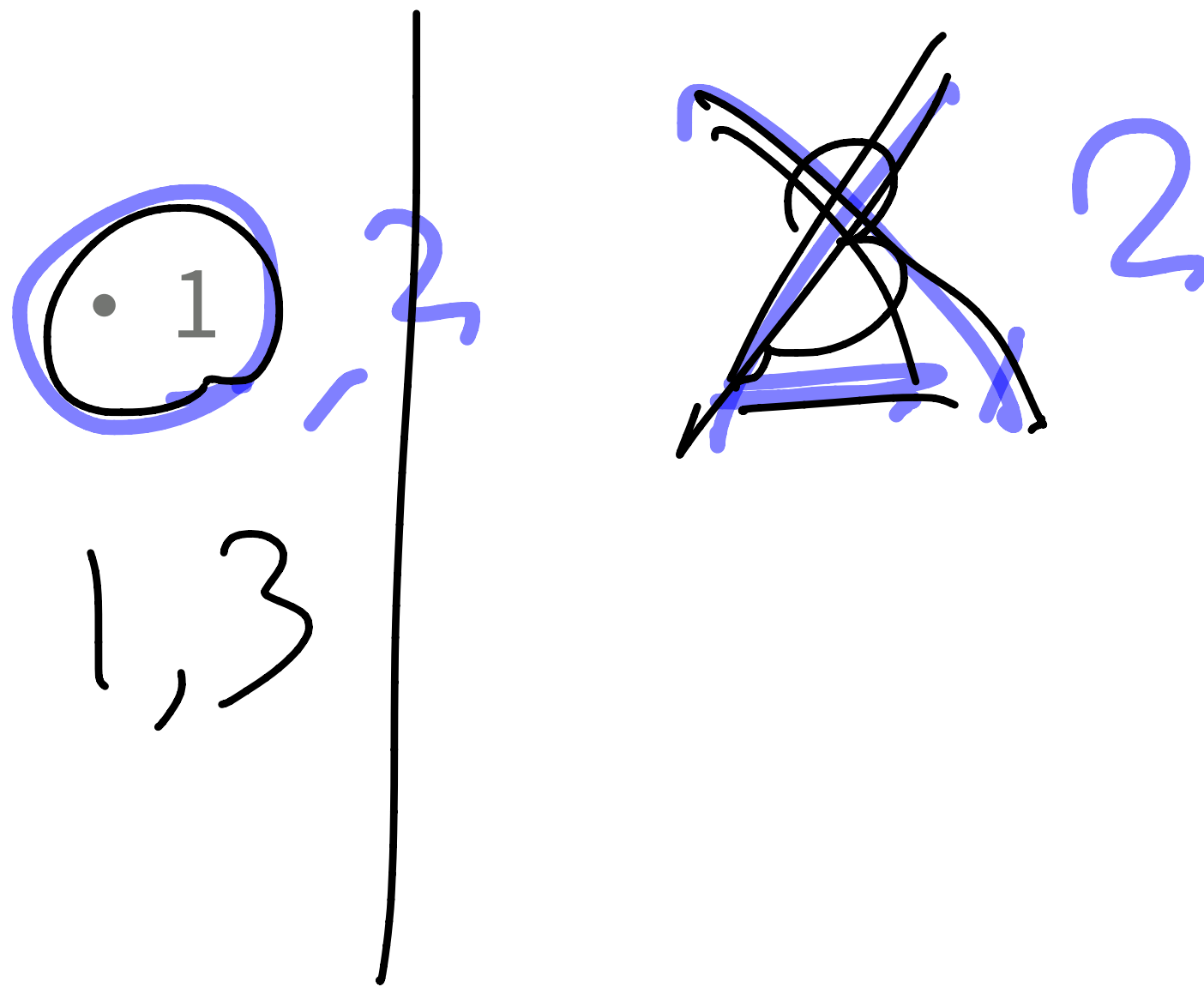
- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자

# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 리스트가 없기 때문에, 새로 추가한다

2



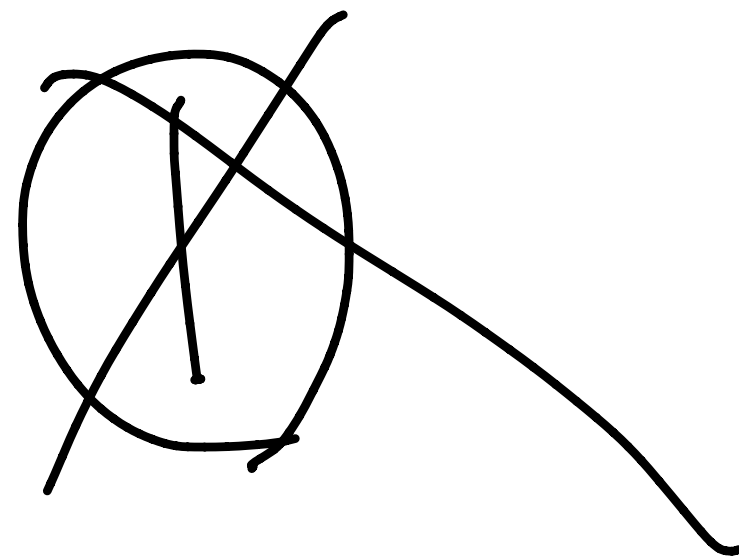
# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, **3**, 1, 2, 4, 3, 4, 2} 인 경우  
↖
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 1을 복사해서 새로 만들고, 뒤에 3을 붙이는 것이 좋다.

• 1 ~~X~~

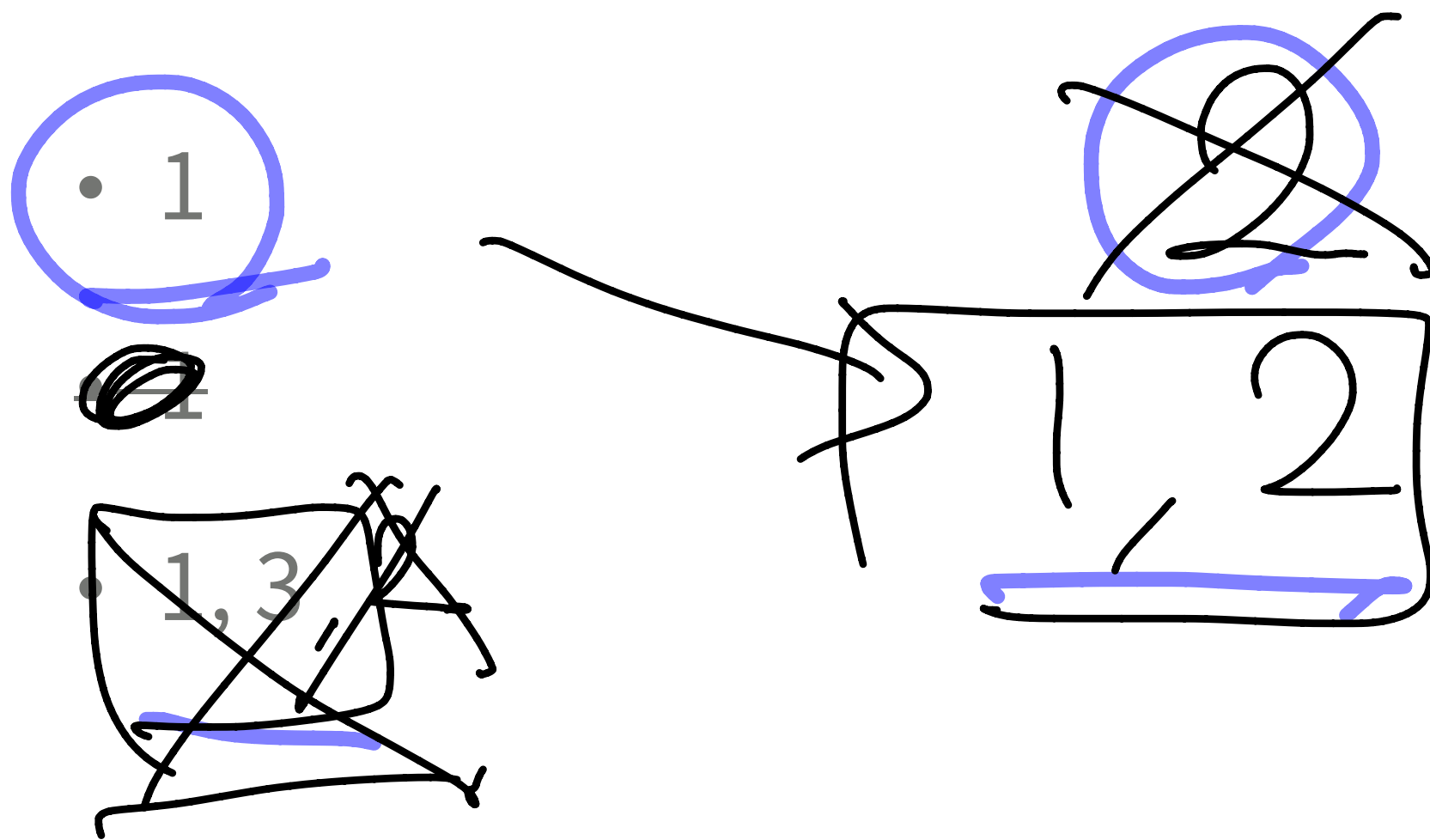
• 1, 3 ~~X~~



# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 새로 만든다. 1은 중복이라 제거한다.



# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 1을 복사한 다음, 뒤에 2를 붙이는 것이 좋다. 1, 3은 절대 정답이 될 수 없기 때문에 제거한다.

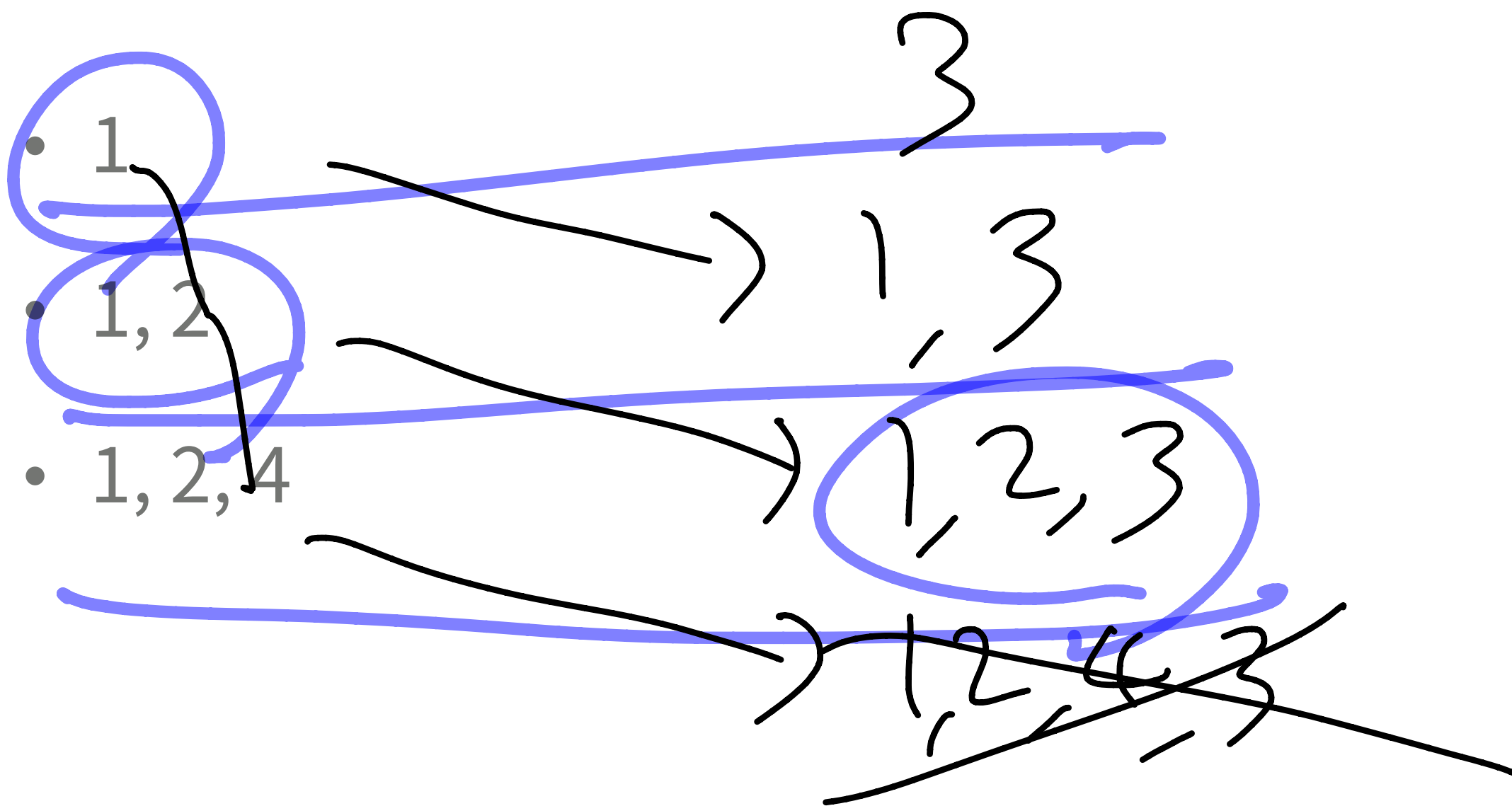
• 1  
• 1, 2  
• 1, 3

~~4~~  
~~1, 4~~  
1, 2, 4

# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 1, 2를 복사한 다음, 뒤에 4를 붙이는 것이 좋다.

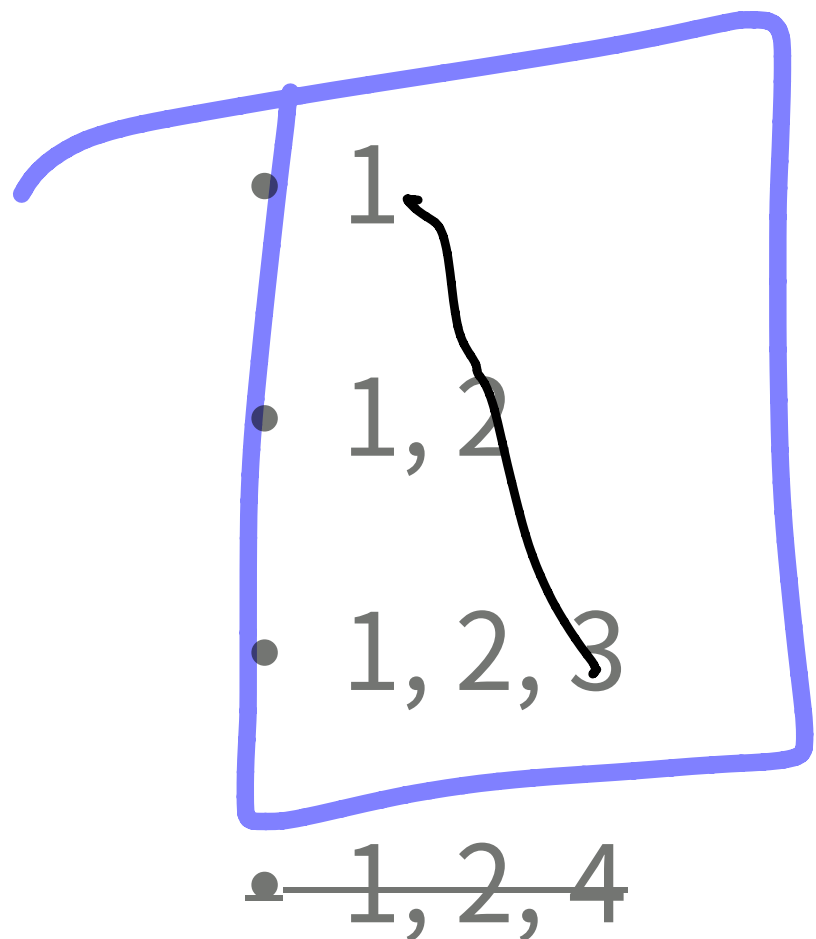




# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

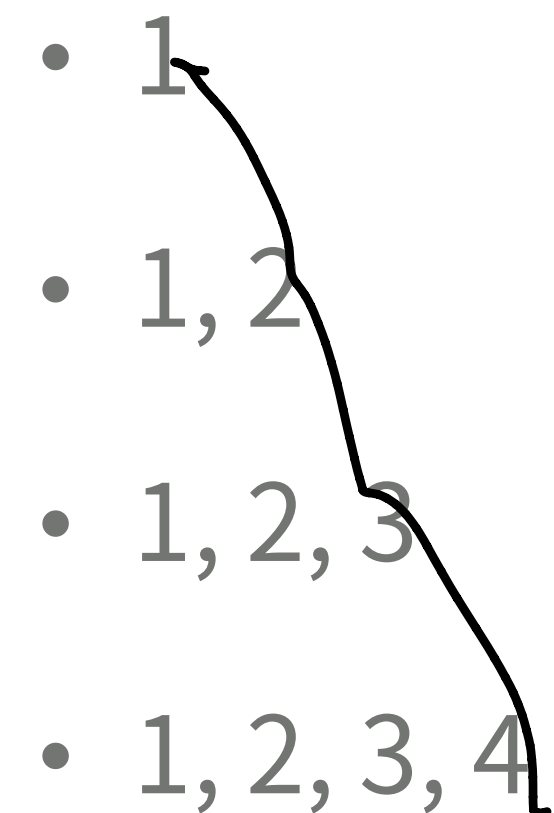
- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 1, 2를 복사한 다음, 뒤에 3를 붙이는 것이 좋다. 1, 2, 4는 정답이 될 수 없다.



# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

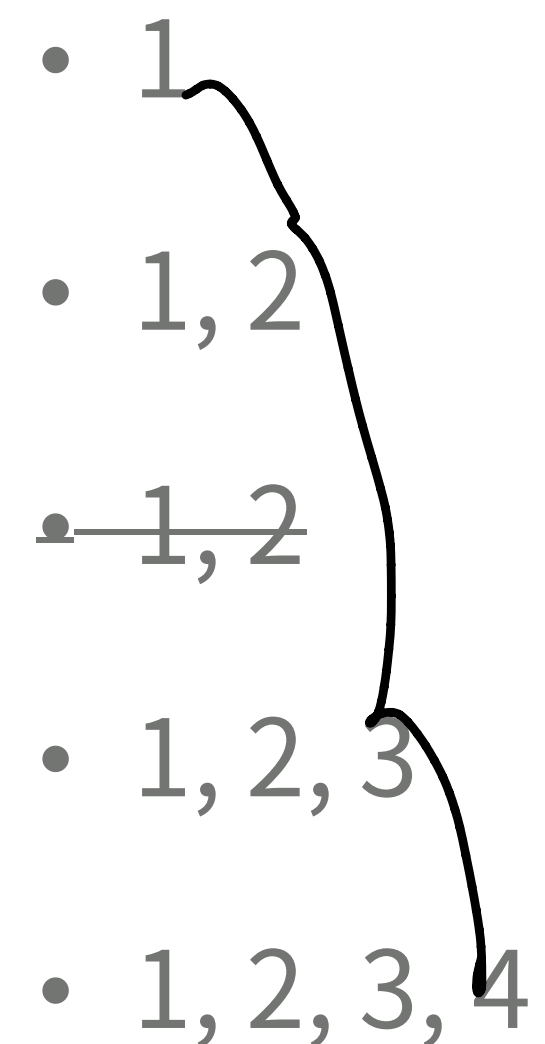
- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 1, 2, 3를 복사한 다음, 뒤에 4를 붙이는 것이 좋다.

- 1
  - 1, 2
  - 1, 2, 3
  - 1, 2, 3, 4
- 

# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자
- 1을 복사한 다음, 뒤에 2를 붙이는 것이 좋다. 1, 2는 중복이라 하나를 제거한다.

- 1
  - 1, 2
  - ~~1, 2~~
  - 1, 2, 3
  - 1, 2, 3, 4
- 

# 가장 긴 증가하는 부분 수열 2

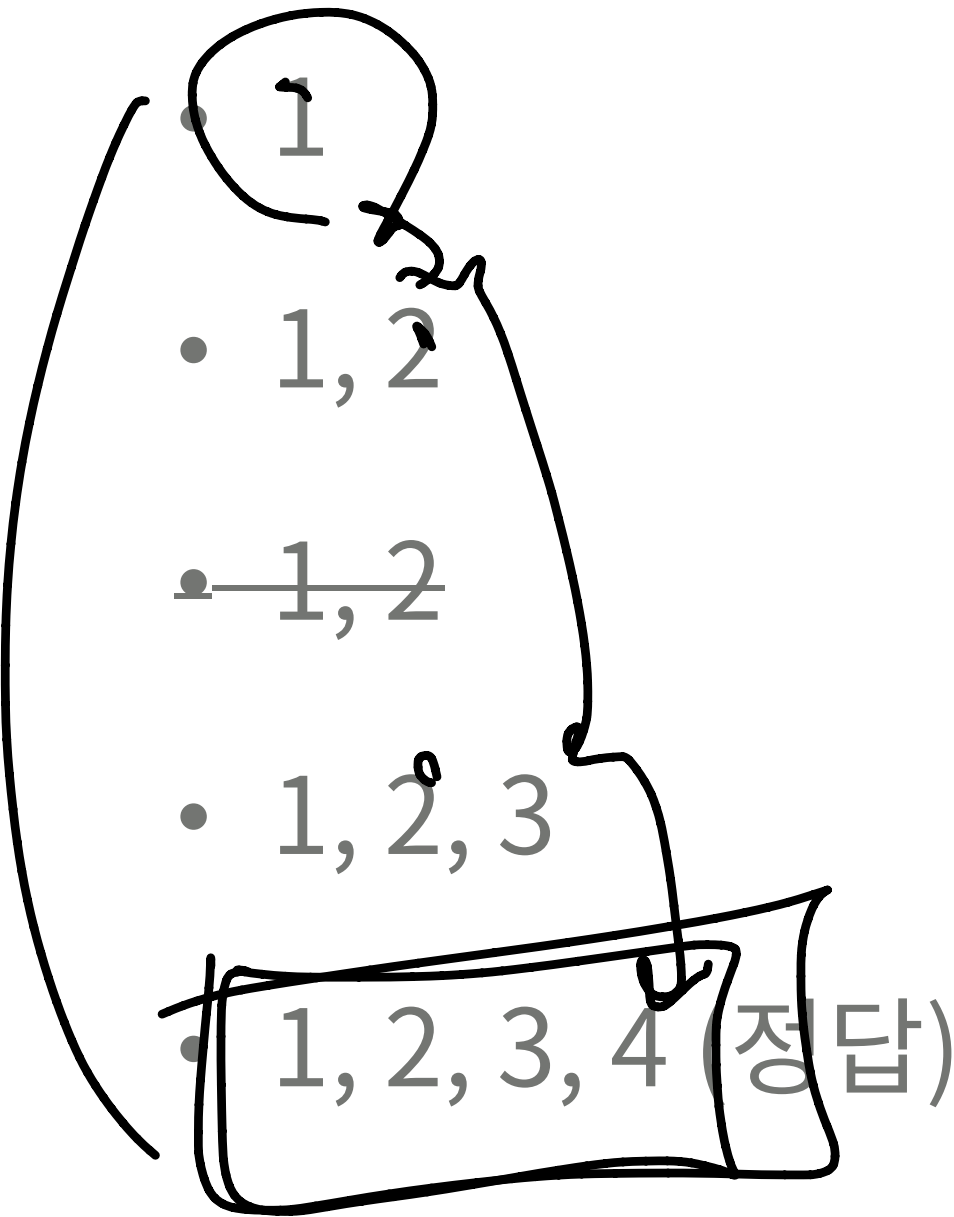
<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우
- 가능한 정답의 리스트를 만들어가면서 답을 구해보자

$N$

길이 1 -  $N$

$O(N^2)$



# 가장 긴 증가하는 부분 수열 2

85

<https://www.acmicpc.net/problem/12015>

- 앞의 방법을 1차원 배열을 하나 이용해서 구현 할 수 있다.

$N \log N$

# 가장 긴 증가하는 부분 수열 2

86

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	/				

# 가장 긴 증가하는 부분 수열 2

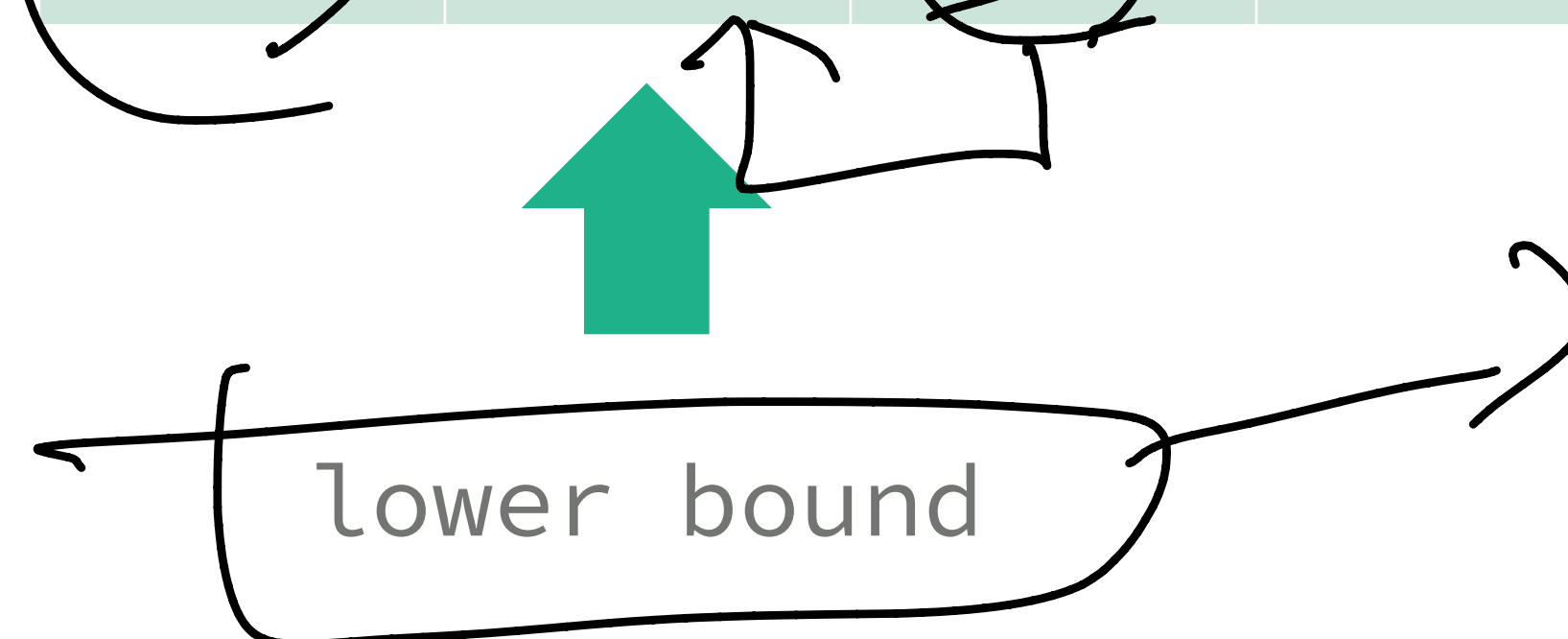
87

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	3			



크게 늘어난 것보다  
가장 빨리

# 가장 긴 증가하는 부분 수열 2

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

$O(N \log N)$

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	3			



$\log N$



# 가장 긴 증가하는 부분 수열 2

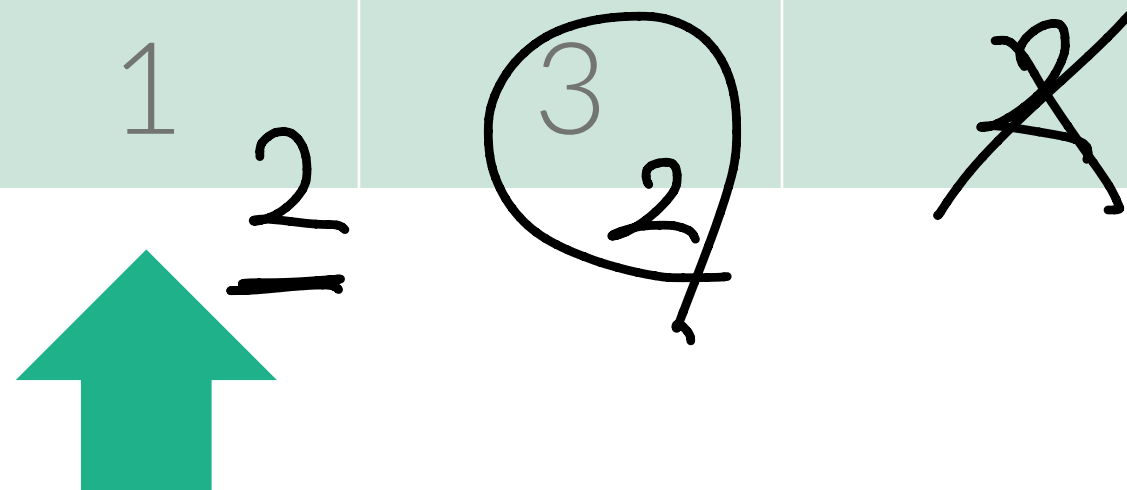
89

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	3	<del>2</del>		



lower bound

# 가장 긴 증가하는 부분 수열 2

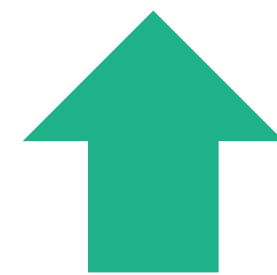
90

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	2			



lower bound

# 가장 긴 증가하는 부분 수열 2

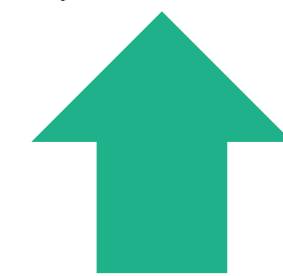
91

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	2	4		



lower bound

# 가장 긴 증가하는 부분 수열 2

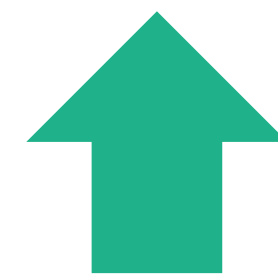
92

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	2	3		



lower bound

# 가장 긴 증가하는 부분 수열 2

93

<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	2	3	4	



lower bound

# 가장 긴 증가하는 부분 수열 2

94

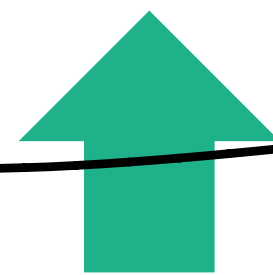
<https://www.acmicpc.net/problem/12015>

- 수열 = {1, 3, 1, 2, 4, 3, 4, 2} 인 경우

길이 4

i	1	2	3	4	5	6	7	8
A[i]	1	3	1	2	4	3	4	2

i	0	1	2	3	4
D[i]	1	2	3	4	



lower bound

- 소스: <http://codeplus.codes/0a0937f490da4773b7a02550c2078455>

