그리디 알고리즘

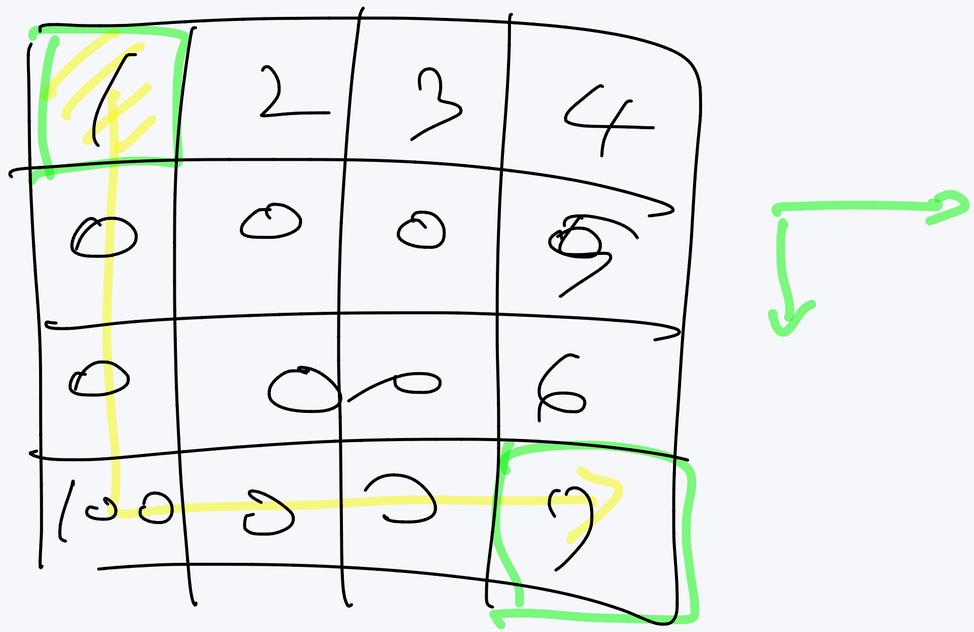
최백준 choi@startlink.io

그리디 알고리즘

그리디 알고리즘

Greedy Algorithm

- 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

10,000

2430원

9570원

- 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 최소로 해야 한다.
- 가장 큰 액면가를 가진 지폐나 동전부터 거슬러 주자.

거스름돈 문제

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

그리디 알고리즘

Greedy Algorithm

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

그리디 알고리즘

Greedy Algorithm

- 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 ㅠㅠ

동전 0

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

- 준규가 가지고 있는 동전은 총 N종류이고, 각각의 동전을 매우 많이 가지고 있다.
- 동전을 적절히 사용해서 그 가치의 합을 K로 만드려고 한다. 이 때 필요한 동전 개수의 최소값을 구하는 프로그램을 작성하시오.
- N개의 줄에 동전의 가치 A_i가 오름차순으로 주어진다. ($1 \le A_i \le 1,000,000$, $A_1 = 1$, $i \ge 2$인 경우에 A_i는 A_{i-1}의 배수)

동전 0

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

- C/C++: https://gist.github.com/Baekjoon/027cae931e328c00725db6eae3cc0c6a
- Java: https://gist.github.com/Baekjoon/6542e3d066eb6685991800027d1ef5af1

회의실 배정

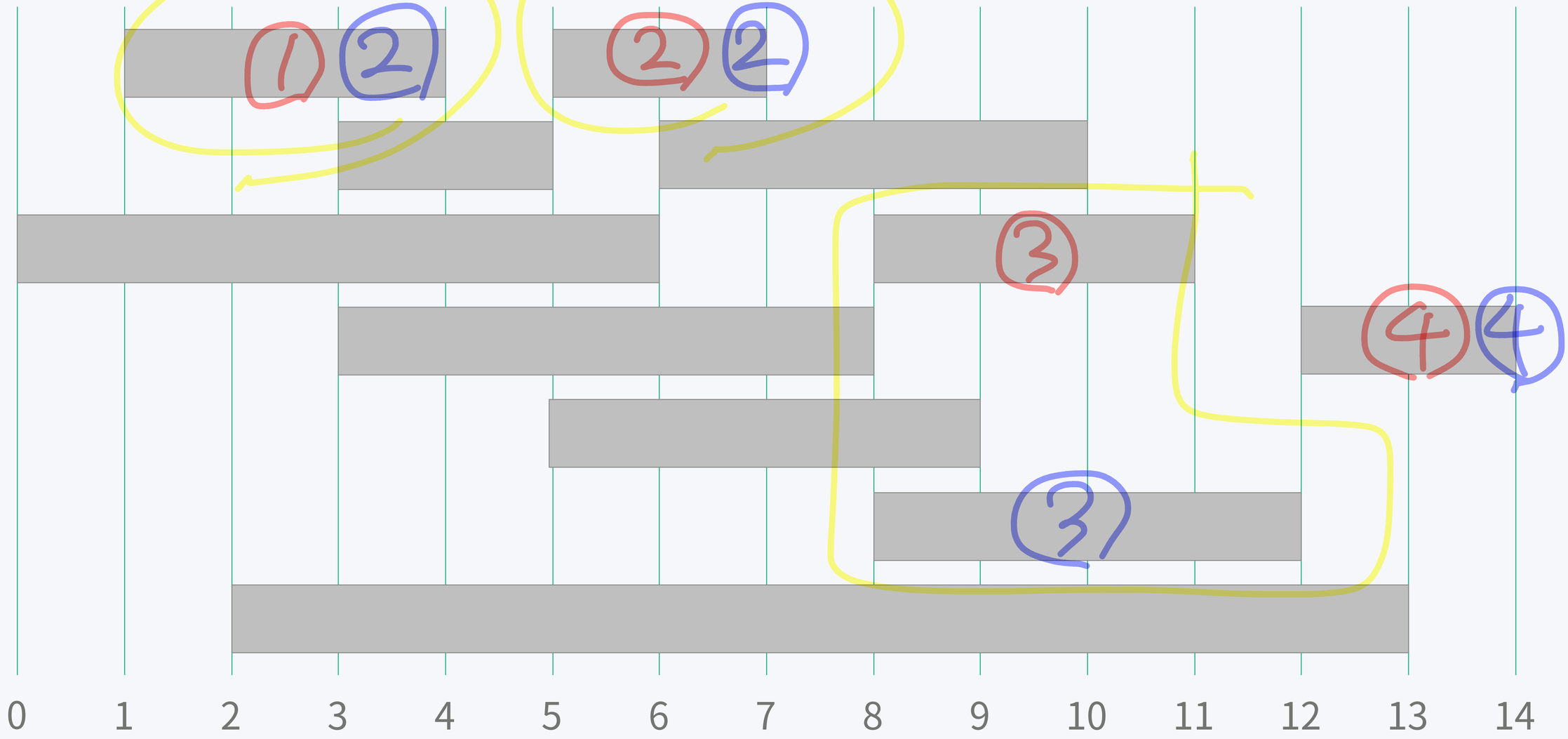
https://www.acmicpc.net/problem/1931

- 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

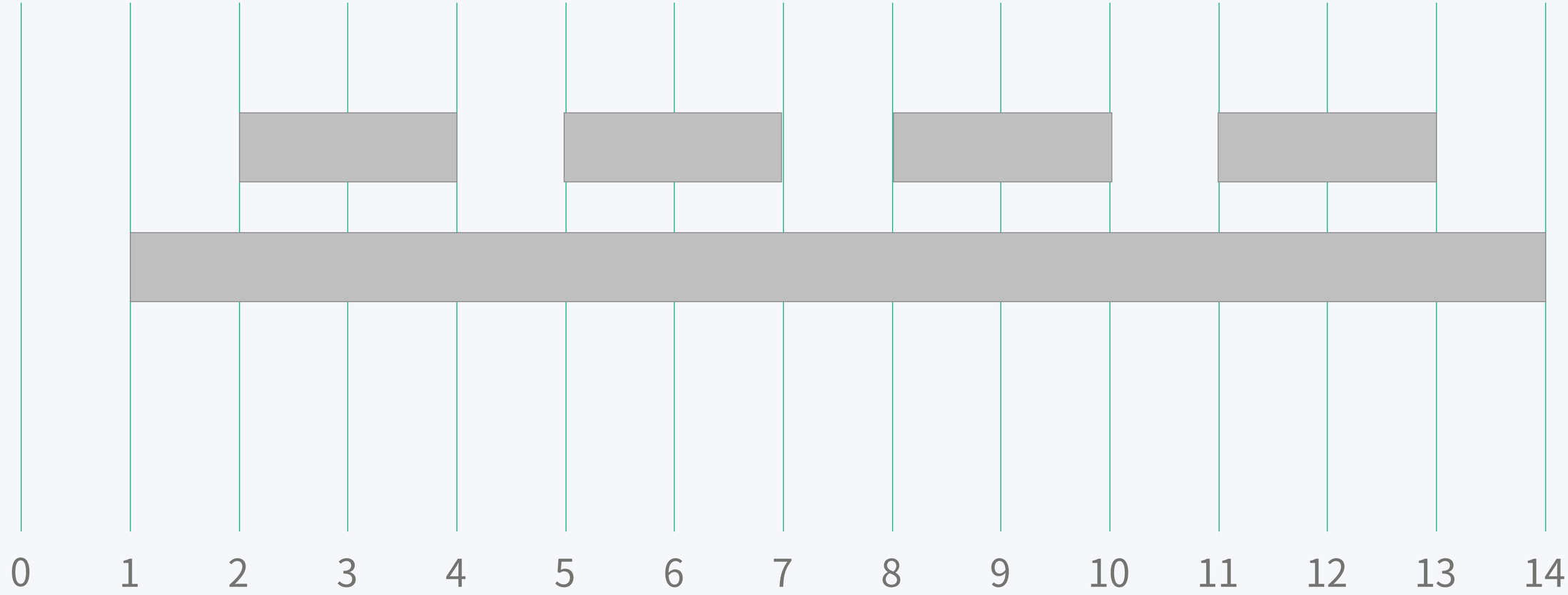
그리디
그냥 정답



회의실 배정

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

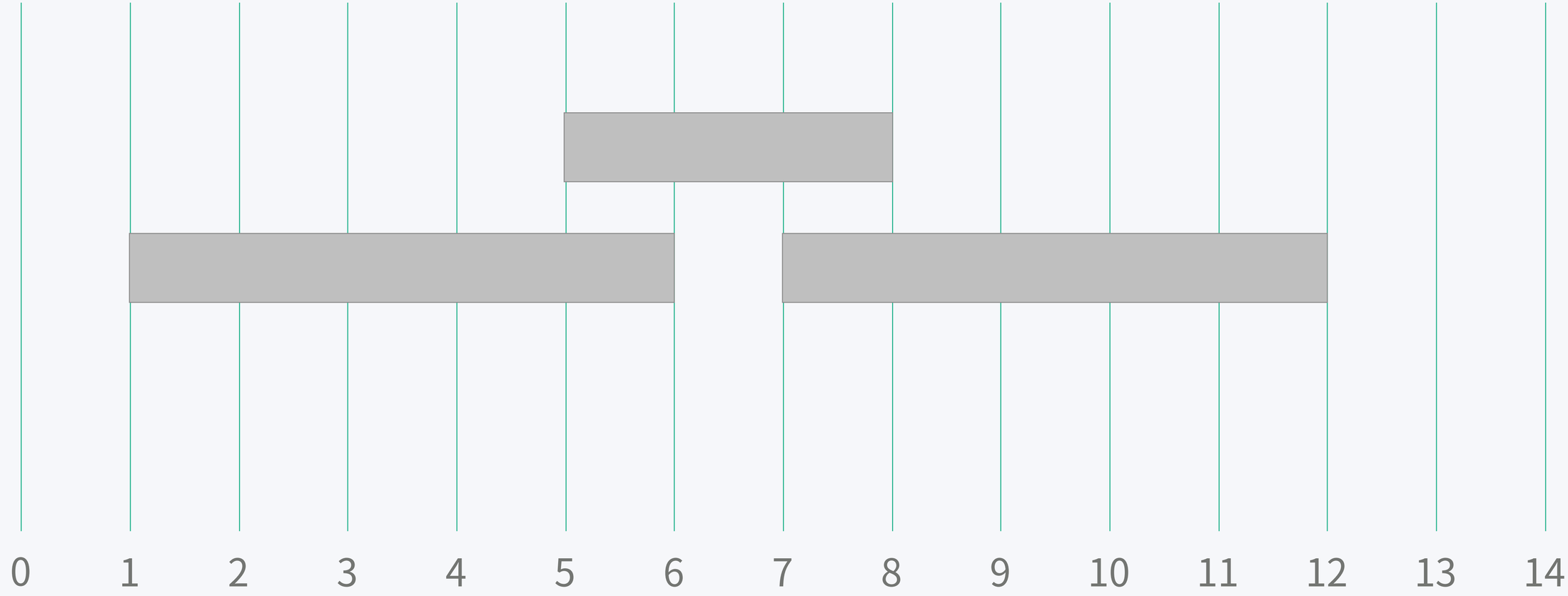
- 일찍 시작하는 회의를 배정한다



회의실 배정

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

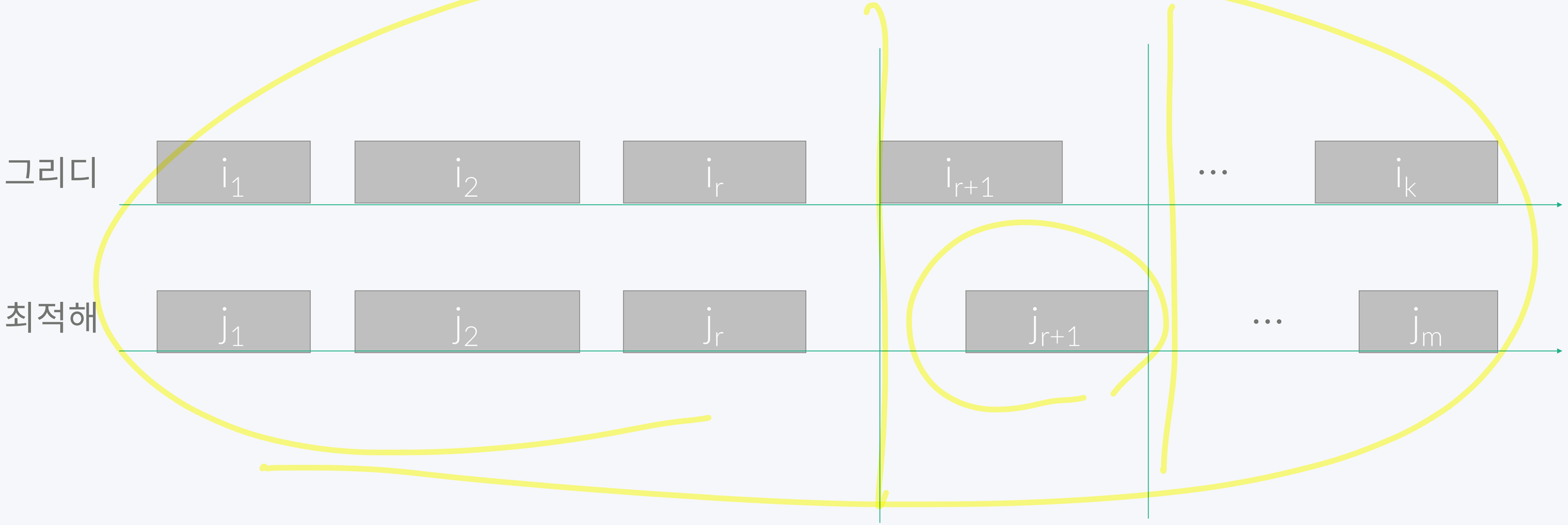
- 짧은 회의를 먼저 배정한다.



회의실 배정

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

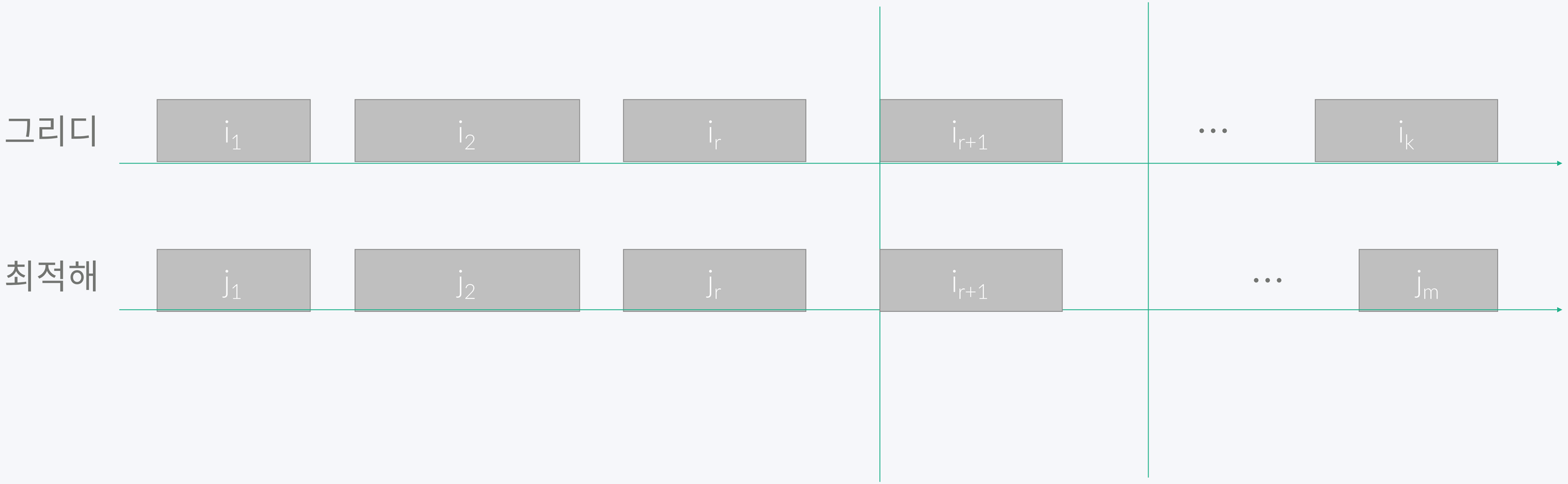
- $i_1, i_2, i_3, \cdots, i_k$를 그리디 알고리즘으로 선택한 정답
- $j_1, j_2, j_3, \cdots, j_m$을 $i_1 = j_1, i_2 = j_2, \cdots, i_r = j_r$ 중에서 r이 가장 큰 최적해라고 하자



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.



회의실 배정

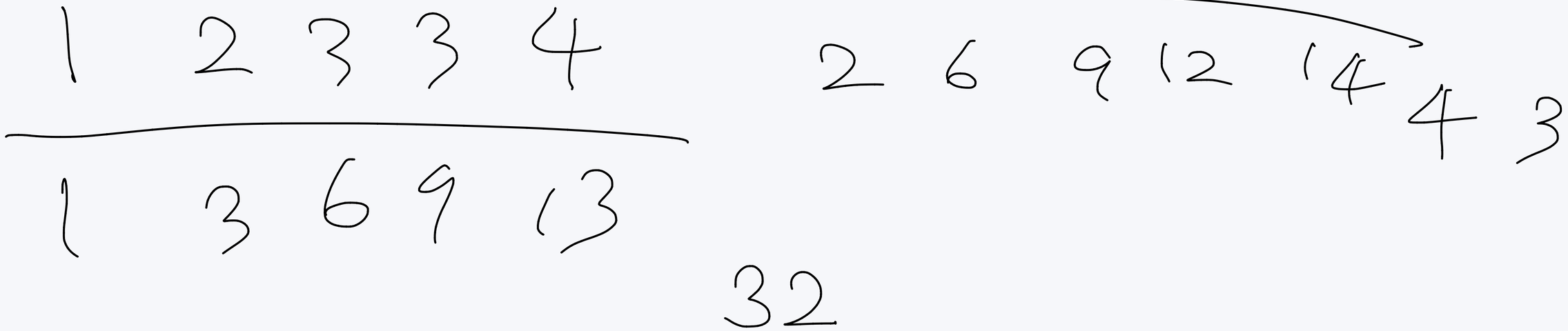
https://www.acmicpc.net/problem/1931

- C++: https://gist.github.com/Baekjoon/c4fb7e9035ec85335719

ATM

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

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



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 = 4분이 걸리게 된다.
- 3번 사람은 1번, 2번 사람이 돈을 뽑을 때까지 기다려야 하기 때문에, 총 3+1+4 = 8분이 필요하게 된다.
- 4번 사람은 3+1+4+3 = 11분, 5번 사람은 3+1+4+3+2 = 13분이 걸리게 된다.
- 이 경우에 각 사람이 돈을 인출하는데 필요한 시간의 합은 3+4+8+11+13 = 39분이 된다.

ATM

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

- 줄을 서 있는 사람의 수 N과 각 사람이 돈을 인출하는데 걸리는 시간 Pi가 주어졌을 때, 각 사람이 돈을 인출하는데 필요한 시간의 합의 최소값을 구하는 문제

ATM

$\Rightarrow P_1 \le P_2 \le \cdots \le P_N$

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

- 기다리는 시간이 짧은 사람부터 ATM을 인출하는 것이 좋다.

$$S = P_1 + (P_1 + P_2) + \cdots + (P_1 + P_2 + \cdots + P_N)$$

$$= N \cdot P_1 + (N-1) \cdot P_2 + \cdots + P_N$$

$(i < j,\ P_i \le P_j)$

ATM

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

- $p_1, p_2, p_3, \ldots, p_n$
- $p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_n$ 이 정답이라고 가정
- 총 돈을 인출하는데 걸리는 시간의 합
- $S = p_1 + (p_1+p_2) + (p_1+p_2+p_3) + \ldots. + (p_1+p_2+\ldots+p_n)$
- $S = n*p_1 + (n-1)*p_2 + \ldots + p_n$
-

ATM

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

- $p_1 \le p_2 \le p_3 \le \dots \le p_n$ 이 정답이라면, 중간에 $i < j$인 p_i와 p_j의 순서를 바꿨을 때, 더 커져야 한다.

$S \le S'$ $S - S' \le 0$

- 총 돈을 인출하는데 걸리는 시간의 합
- $S = n*p_1 + \dots + (n-(i-1))*p_i + \dots + (n-(j-1))*p_j + \dots + p_n$
- pi와 pj의 순서를 바꿨을 때 시간의 합
- $S' = n*p_1 + \dots + (n-(i-1))*p_j + \dots + (n-(j-1))*p_i + \dots + p_n$

ATM

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

- S가 정답이기 때문에, S ≤ S'를 만족해야 한다.
- S ≤ S' 라면 S-S' ≤ 0 이 되어야 한다.

ATM

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

$S - S' = (i - j)(p_j - p_i)$

$(i - j) < 0$

$(p_j - p_i) \geq 0$

$i < j$ $\quad$ $p_i \leq p_j$

- $S\text{-}S' = (n\text{-}i+1)*p_i + (n\text{-}j+1)*p_j - (n\text{-}i+1)*p_j - (n\text{-}j+1)*p_i$
- $= (n\text{-}i+1\text{-}n+j\text{-}1)p_i + (n\text{-}j+1\text{-}n+i\text{-}1)*p_j$
- $= (-i+j)p_i + (-j+i)p_j$
- $= (j-i)p_i + (i-j)p_j$
- $= -(i-j)p_i + (j-j)p_j$
- $= (i-j)(p_j-p_i)$

$S - S' \leq 0$

$S \leq S'$

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_i는 양수 또는 0이기 때문에
- $S-S' \leq 0$이다.

- 따라서 오름차순이 정답이다.

ATM

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

- C++: https://gist.github.com/Baekjoon/aa84c041b25abd96b583
- Java: https://gist.github.com/Baekjoon/a29efd6debc4c11ca8bff11673424cdc

잃어버린 괄호

https://www.acmicpc.net/problem/1541

- 식에 괄호를 적절히 쳐서 식의 값을 최소로 만드는 문제

- 1+2+3-4-(5+6+7+8)-(9+10)-11-(12+13+14+15)

잃어버린 괄호

https://www.acmicpc.net/problem/1541

- 식에 괄호를 적절히 쳐서 식의 값을 최소로 만드는 문제
- -가 나오면, 항상 뒤의 식을 모두 –로 만들 수 있다.
- `1+2+3-4-5+6+7+8-9+10-11-12+13+14+15`
- `1+2+3-4-(5+6+7+8)-(9+10)-11-(12+13+14+15)`

잃어버린 괄호

https://www.acmicpc.net/problem/1541

- C/C++: https://gist.github.com/Baekjoon/bd8686d60129d6ac99a0ec048ca58e3b

수 묶기

https://www.acmicpc.net/problem/1744

- 길이가 N인 수열에서 두 수를 적절히 묶어서 수열의 합을 최대로 하는 문제
- 수의 순서를 바꿔도 상관이 없다
- 같은 위치에 있는 수를 묶는 것은 불가능하고
- 각 수는 단 한 번만 묶거나 묶지 않아야 한다
- 묶은 후에는 두 수의 곱이 수가 됨
- 이 때 최대 찾기

수 묶기

https://www.acmicpc.net/problem/1744

- 0, 1, 2, 4, 3, 5 인 경우
- 4와 5 를 묶고 2와 3을 묶으면
- 0 + 1 + (4*5) + (2*3) = 27

수 묶기

https://www.acmicpc.net/problem/1744

- 양수는 큰 수끼리
- 음수는 작은 수 끼리
- 0은 묶지 않는 것이 최대

수 묶기

https://www.acmicpc.net/problem/1744

- 주의해야할 점
- 1은 묶지 않는 것이 좋다
- 묶이지 않는 음수가 있는 경우 0을 이용할 수 있다

수 묶기

https://www.acmicpc.net/problem/1744

- C++: https://gist.github.com/Baekjoon/5c3ff98902f88ba94638
- Java: https://gist.github.com/Baekjoon/4a24036b064f62f55d9d44d951e500e0

대회 or 인턴

https://www.acmicpc.net/problem/2875

- 여학생 N명, 남학생 M명이 있다
- 1팀: 여2, 남1
- K명은 인턴에 참가해야 한다
- 몇 팀을 만들 수 있을까?

대회 or 인턴

https://www.acmicpc.net/problem/2875

- 한 팀을 만드려면 다음과 같은 조건을 만족해야 한다
- 여학생이 2명 이상
- 남학생이 1명 이상
- $M+N \geq K+3$
    - 팀은 3명이고, 인턴은 K명이 해야하기 때문

대회 or 인턴

https://www.acmicpc.net/problem/2875

- C++: https://gist.github.com/Baekjoon/f7a98c8b5787d0df391a
- Java: https://gist.github.com/Baekjoon/1f92d575df363247d910a56aa1f61fbe

30

https://www.acmicpc.net/problem/10610

- 어떤 수 N이 주어졌을 때, 숫자를 섞어 30의 배수로 만드는 문제
- 이 때, 가장 큰 수를 만들어야 함
- N = 30, 답 = 30
- N = 102, 답 = 210
- N = 2931, 답 = 불가능

30

https://www.acmicpc.net/problem/10610

- 30은 2*3*5 이다
- 즉, N이 30으로 나누어 떨어지려면, 2, 3, 5로 나누어 떨어져야 한다
- 2로 나누어 떨어지는 수
  - 마지막 자리가 짝수
- 3으로 나누어 떨어지는 수
  - 자리의 합이 3으로 나누어 떨어져야 함
- 5로 나누어 떨어지는 수
  - 마지막 자리가 0 또는 5
- 30으로 나누어 떨어지는 수
  - 마지막 자리가 0이면서 자리의 합이 3으로 나누어 떨어져야 함

30

https://www.acmicpc.net/problem/10610

- 자리의 합은 변하지 않기 때문에, 마지막 자리만 0으로 만들어주면 되는 문제
- 가장 큰 수이기 때문에, 그냥 내림차순 정렬을 하면 되는 문제

30

https://www.acmicpc.net/problem/10610

- C++: https://gist.github.com/Baekjoon/2da16249beb204683fbc
- Java: https://gist.github.com/Baekjoon/635ebbefe004d0f22dbdf862932793d2

병든 나이트

https://www.acmicpc.net/problem/1783

- N*M 크기의 체스판 가장 왼쪽 아래칸에 나이트가 있다
- 1 ≤ N, M ≤ 2,000,000,000
- 이동할 수 있는 방법

1. 2칸 위로, 1칸 오른쪽
2. 1칸 위로, 2칸 오른쪽
3. 1칸 아래로, 2칸 오른쪽
4. 2칸 아래로, 1칸 오른쪽

- 이동 횟수가 4번 이상이면 위의 방법을 모두 1번씩은 이용해야 함
- 이 때, 방문할 수 있는 칸의 최대 개수

병든 나이트

https://www.acmicpc.net/problem/1783

1. height = 1인 경우
2. height = 2인 경우
3. height ≥ 3보다 큰 경우

병든 나이트

https://www.acmicpc.net/problem/1783

1. height = 1인 경우
    - 움직일 수 없기 때문에 정답은 1
2. height = 2인 경우
3. height ≥ 3보다 큰 경우

병든 나이트

https://www.acmicpc.net/problem/1783

1. height = 1인 경우
    - 움직일 수 없기 때문에 정답은 1
2. height = 2인 경우
    - 두 가지 방법만 사용할 수 있다. (+2, +1), (+2, -1)
    - 따라서, 정답은 min(4, (width + 1) / 2)
    - 이동 횟수 제한 때문에 4가 필요함
3. height ≥ 3보다 큰 경우

병든 나이트

https://www.acmicpc.net/problem/1783

1. height = 1인 경우
    - 움직일 수 없기 때문에 정답은 1
2. height = 2인 경우
    - 두 가지 방법만 사용할 수 있다. (+2, +1), (+2, -1)
    - 따라서, 정답은 min(4, (width + 1) / 2)
    - 이동 횟수 제한 때문에 4가 필요함
3. height ≥ 3보다 큰 경우
    1. width ≥ 7
    2. width < 7

병든 나이트

https://www.acmicpc.net/problem/1783

3. height ≥ 3보다 큰 경우
    1. width ≥ 7
        - 4가지 방법을 모두 사용하면 (7, 1)으로 이동한다
        - 여기서부터 (+1, +2)와 (+1, -2)를 번갈아가면서 사용
        - width – 2
    2. width < 7

병든 나이트

https://www.acmicpc.net/problem/1783

3. height ≥ 3보다 큰 경우
    1. width ≥ 7
    2. width < 7
        - 4가지 방법을 모두 사용할 수 없다
        - (+1, +2)와 (+1, -2)를 번갈아 가면서 사용할 수 있다
        - min(4, width)

병든 나이트

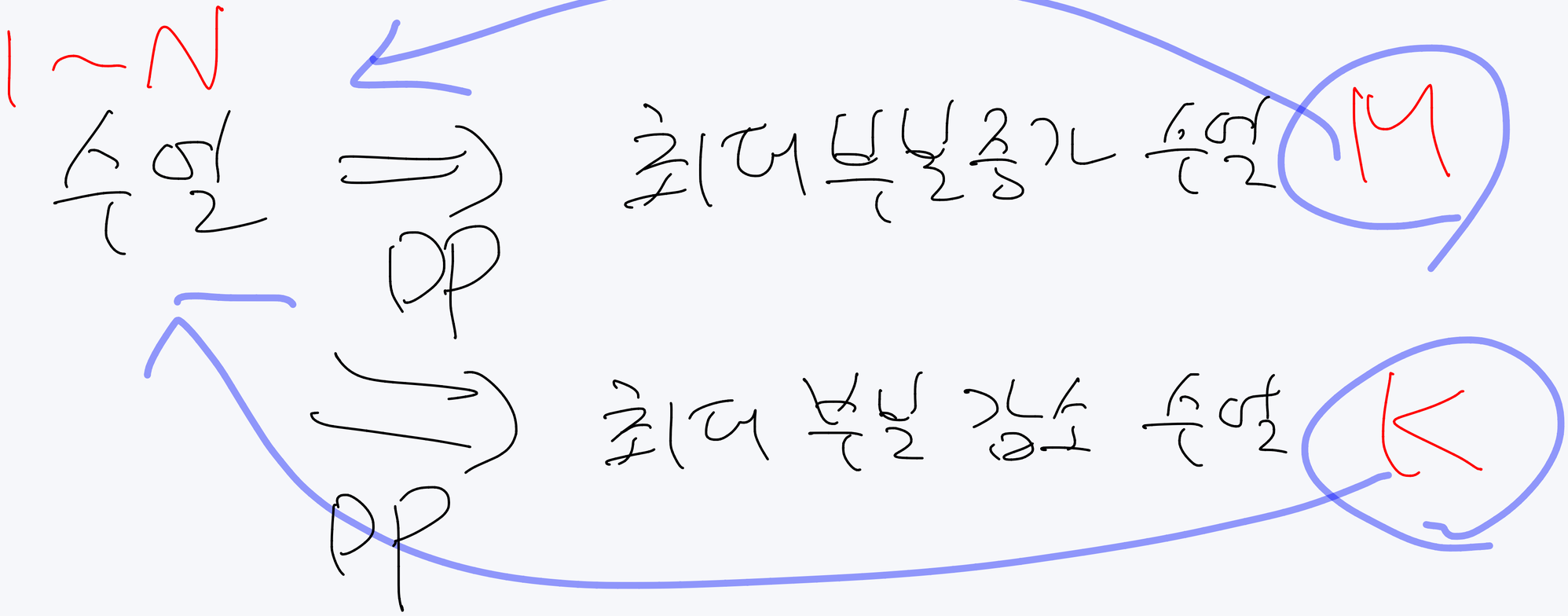
https://www.acmicpc.net/problem/1783

- C++: https://gist.github.com/Baekjoon/1a798b330887a382d947
- Java: https://gist.github.com/Baekjoon/980cd1791d8ee1784bf0af9b8d51dd11

NMK

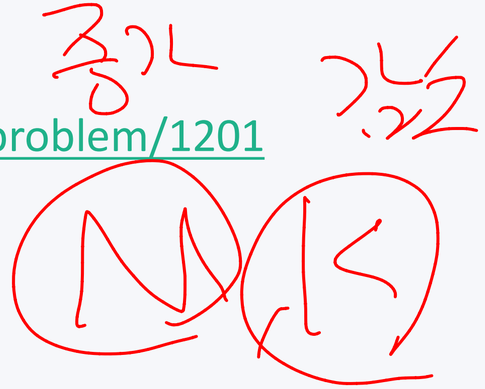
https://www.acmicpc.net/problem/1201

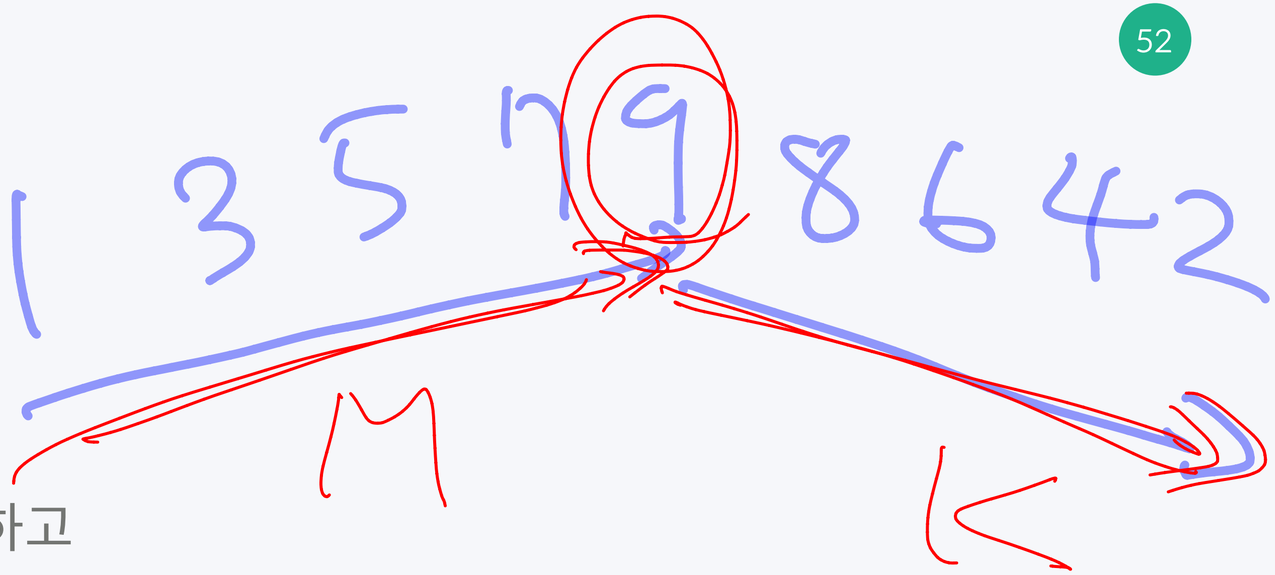
- 1부터 N까지의 수를 한 번씩 이용해서 최대 부분 증가 수열의 길이가 M이고, 최대 부분 감소 수열의 길이가 K인 수열을 구하는 문제



NMK

https://www.acmicpc.net/problem/1201





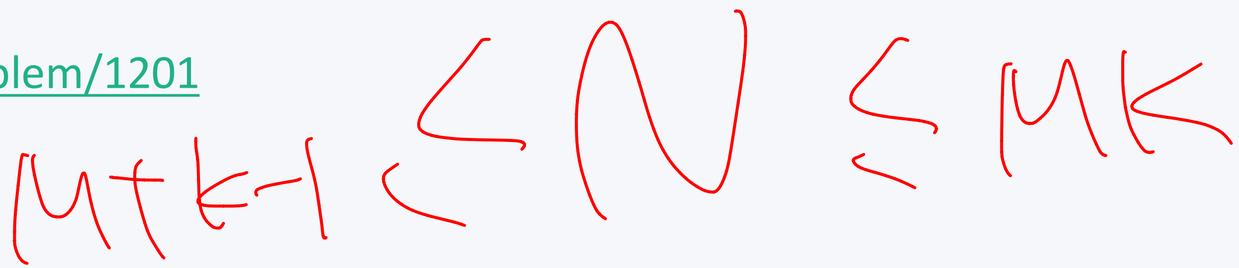
- 불가능한 경우 찾기
- 적어도 M개의 정수는 증가 수열에 포함되어야 하고
- 적어도 K개의 정수는 감소 수열에 포함되어야 한다
- 두 수열은 최대 정수 1개를 공유할 수 있기 때문에
- N ≥ M+K-1 이어야 한다

M+K-1 ≤ N ≤ ?

NMK

https://www.acmicpc.net/problem/1201

- 불가능한 경우 찾기



- 또, N은 MK를 넘을 수 없다
- N = MK+1인 경우에 길이가 M+1인 증가 수열이나 길이가 K+1인 감소 수열을 반드시 만들 수 있다.
- 비둘기집 원리로 증명할 수 있음
- Erdős-Szekeres Theorem
- http://mathworld.wolfram.com/Erdos-SzekeresTheorem.html

NMK

https://www.acmicpc.net/problem/1201

N=13, M=5, K=4

등분

- M+K-1 ≤ N ≤ MK 인 경우에만 답을 찾을 수 있다

1 2 3 4 5 6 7 8 9 10 11 12 13

4 3 2 1 6 5 9 8 7 10 13 12 11

그룹의 크기≤K

정렬기준 K

NMK

https://www.acmicpc.net/problem/1201

1. 1부터 N까지 수를 오름차순으로 적는다
2. 수를 M등분 한다. 이 때, 그룹에 들어있는 수는 K보다 작거나 같아야 하며, 적어도 한 그룹은 들어있는 수의 개수가 K이어야 한다
3. 각 그룹에 들어있는 수의 순서를 뒤집는다
4. 끝

NMK

https://www.acmicpc.net/problem/1201

- N = 13, M = 5, K =4 인 경우

1. 1 2 3 4 5 6 7 8 9 10 11 12 13
2. [1 2 3] [4] [5 6 7 8] [9 10] [11 12 13]
3. [3 2 1] [4] [8 7 6 5] [10 9] [13 12 11]

- 사전 순으로 가장 앞서는 순열을 찾는 경우에는
- [1] [2] [5 4 3] [9 8 7 6] [13 12 11 10]
- 와 같이 나누어야 한다

NMK

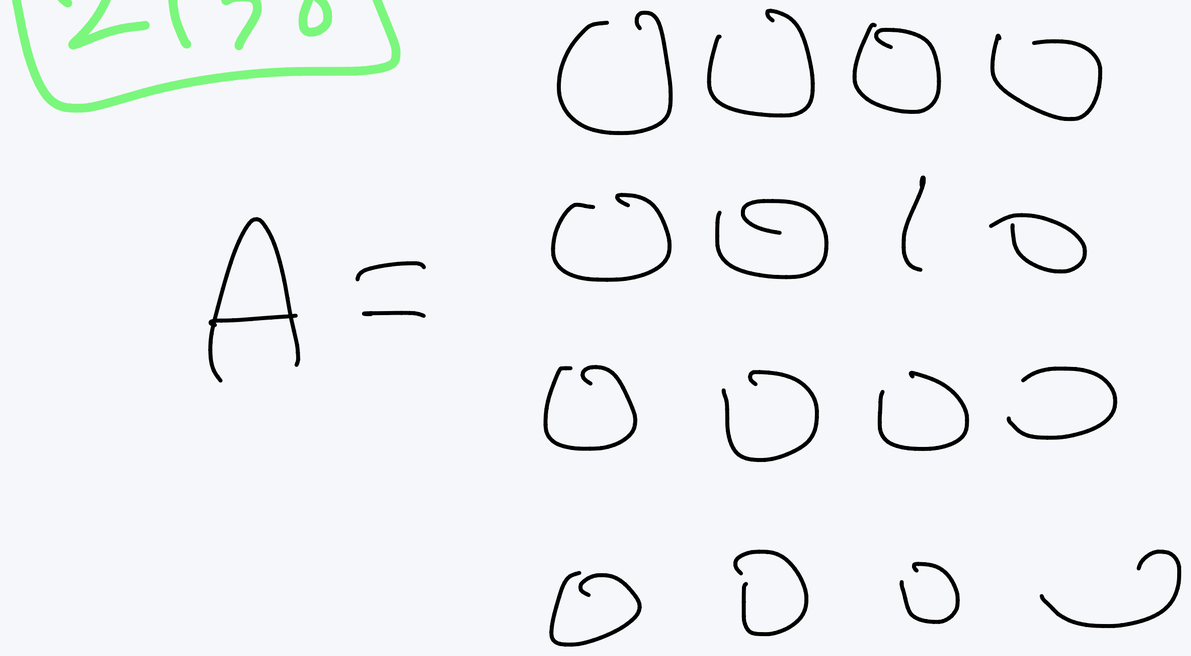
https://www.acmicpc.net/problem/1201

- C/C++: https://gist.github.com/Baekjoon/d62c5fc14d8f7f3cc68f
- Java: https://gist.github.com/Baekjoon/21fa0f6d984e0c0b4c4cc3db920332cd

행렬

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

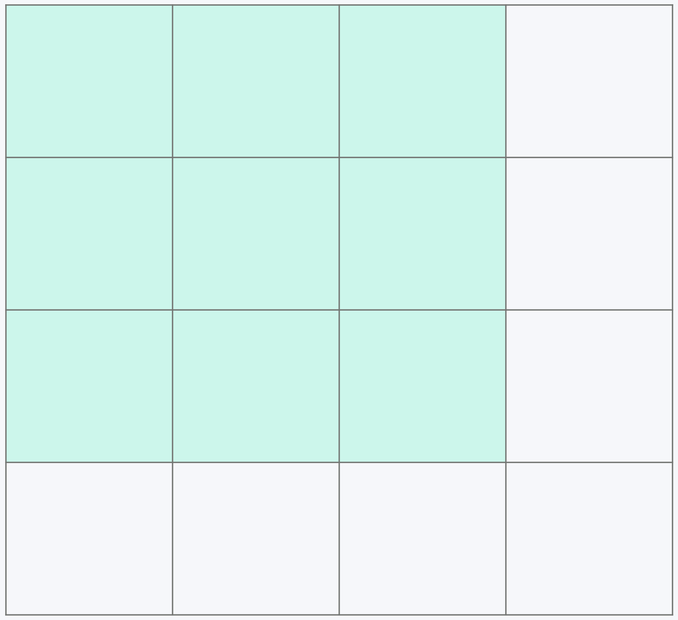
- 0과 1로만 이루어진 행렬 A와 행렬 B가 있다. 이 때, 행렬 A를 행렬 B로 바꾸는데 필요한 연산의 횟수의 최소값을 구하는 문제
- 행렬을 변환하는 연산은 어떤 3*3크기의 부분 행렬에 있는 모든 원소를 뒤집는 것이다. (0 -> 1, 1 -> 0)



행렬

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

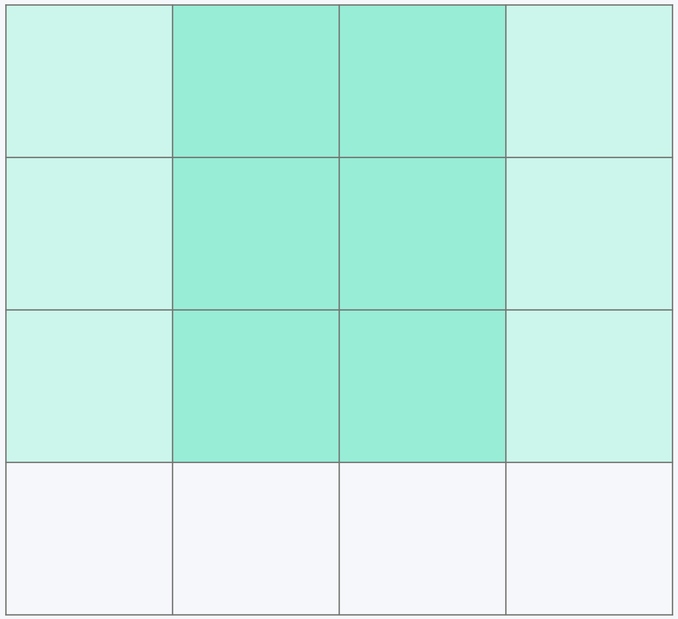
- (0, 0)을 바꿀 수 있는 칸은 (0, 0) ~ (2, 2)를 뒤집는 것 밖에 없다.



행렬

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

- (0, 1)을 바꿀 수 있는 칸은 (0, 0) ~ (2, 2)와 (0, 1) ~ (2, 3)을 뒤집는 것 밖에 없다.



행렬

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

- 모든 (i, j) (0 ≤ i < N-2, 0 ≤ j < M-2) 에 대해서, row-major 순서로 순회하면서, 다른 경우에 (i, j) ~ (i+2, j+2)를 뒤집어 주면 된다.

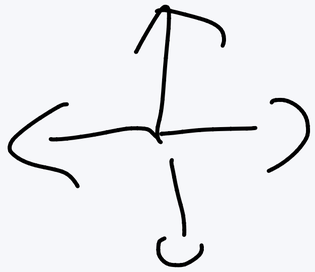
행렬

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

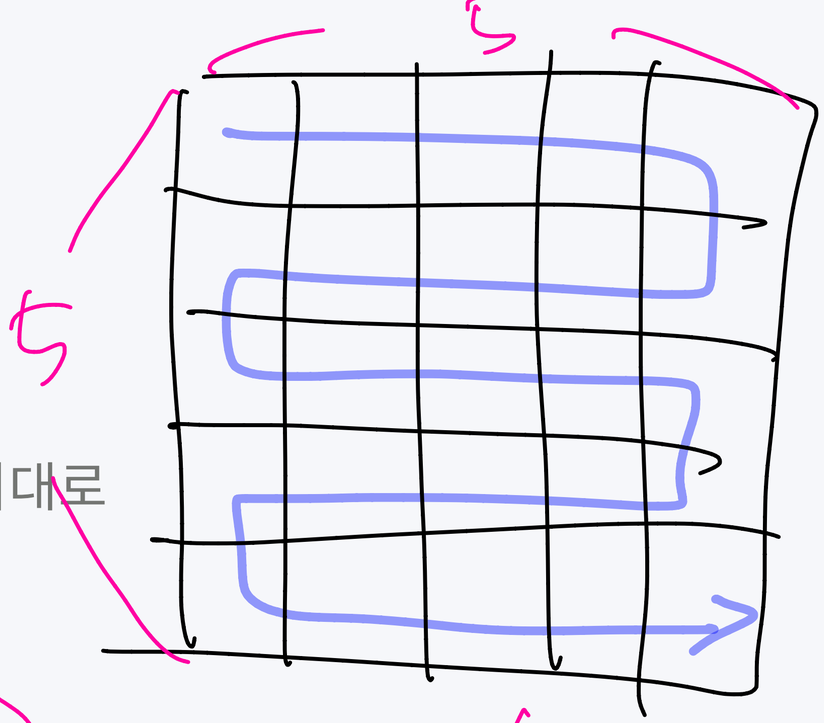
- C/C++: https://gist.github.com/Baekjoon/f88dc83db3507180c61e12b7a7339118

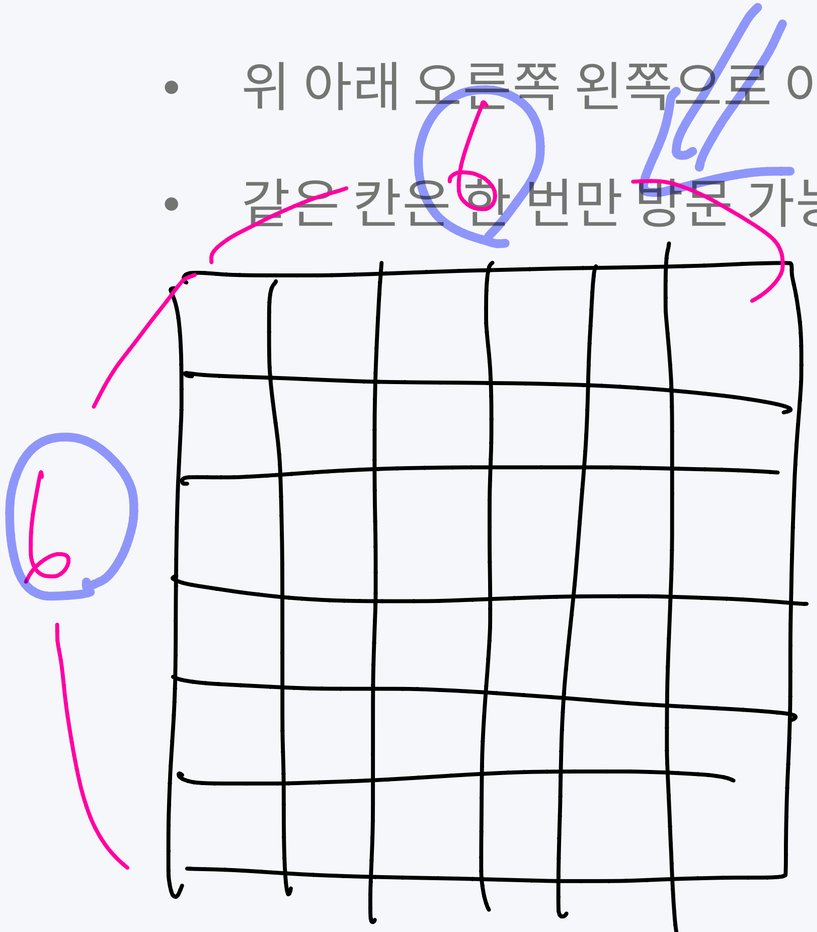
롤러코스터

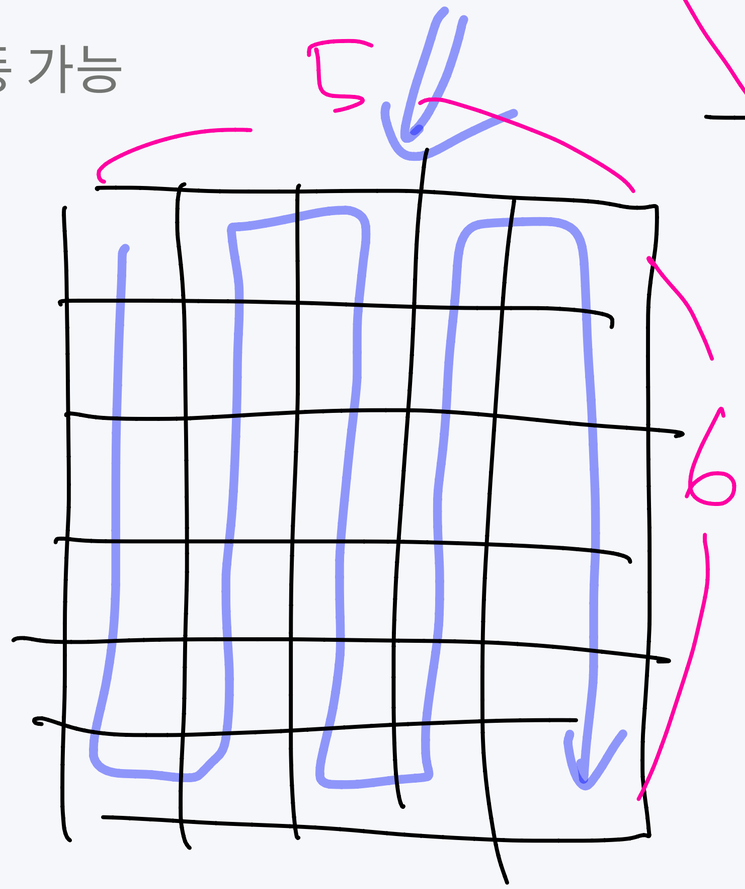
https://www.acmicpc.net/problem/2873

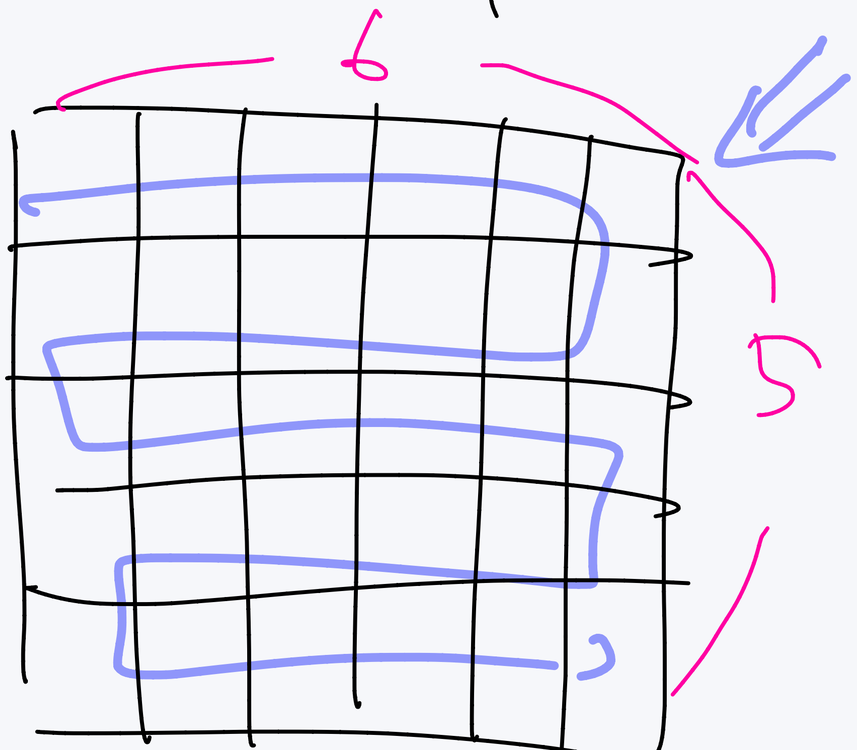


- R * C 크기의 보드의 각 칸에 숫자가 써있다
- (1, 1) -> (R, C)로 이동하는데 방문한 칸의 숫자의 합을 최대로
- 위 아래 오른쪽 왼쪽으로 이동 가능
- 같은 칸은 한 번만 방문 가능









롤러코스터

https://www.acmicpc.net/problem/2873

- R 또는 C가 홀수면 모든 칸을 방문할 수 있음

롤러코스터

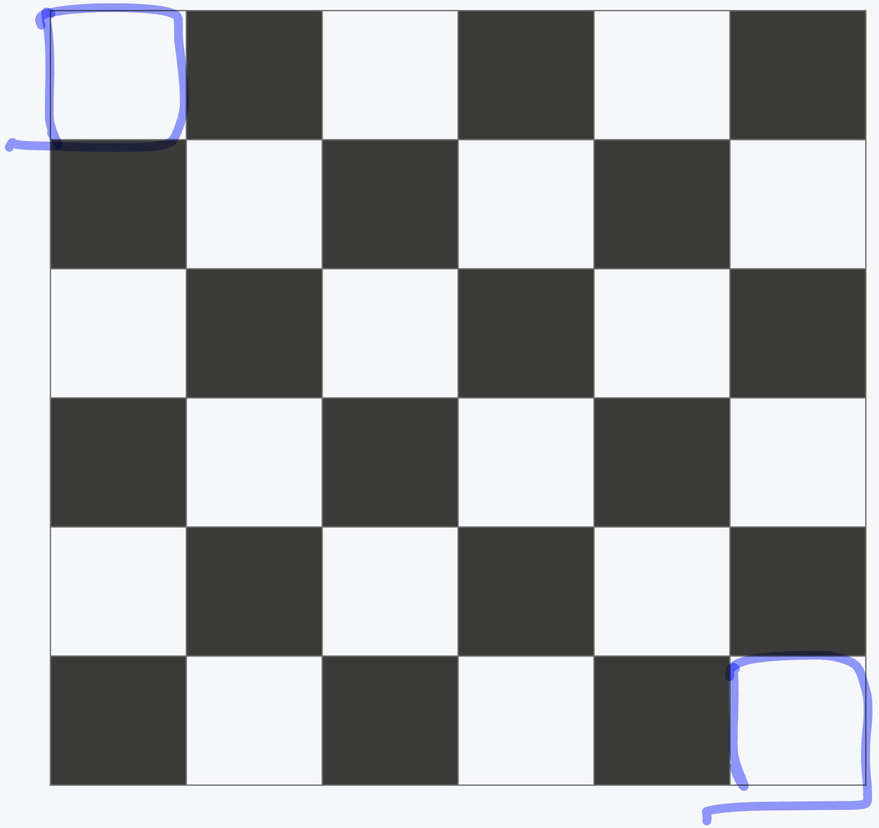
https://www.acmicpc.net/problem/2873

- R과 C가 모두 짝수면 모든 칸을 방문하는 것은 불가능

롤러코스터

https://www.acmicpc.net/problem/2873

- 모든 칸을 체스판 처럼 검정과 흰색으로 칠했다고 하자
- (1, 1)과 (R, C)의 색은 흰색이다
- (1, 1)과 (R, C)로 가는 모든 경로는
- 흰->검->흰->…->흰->검->흰
- 방문한 칸은 흰>검이다.
- 방문하지 않은 칸 흰<검
- 따라서, 모든 칸을 방문하는 것이 불가능



롤러코스터

https://www.acmicpc.net/problem/2873

- 흰 칸 한 칸을 방문하지 않는다면, 나머지 칸은 모두 방문 불가
- 검정 칸 한 칸을 방문하지 않으면, 나머지 칸을 모두 방문 가능

- 따라서, 방문하지 않을 검정 칸 하나를 선택해야 함
- 방문한 칸의 합의 최대를 구하는 문제이기 때문에, 가장 작은 값을 가지는 검정 칸을 선택!

롤러코스터

https://www.acmicpc.net/problem/2873

- 문제를 변형해서 풀기
- 두 사람이 (1, 1), (R, C)에 있고, 서로 만날때까지 이동하는 문제
- (1, 1)에 있는 사람을 A, (R, C)에 있는 사람을 B

롤러코스터

https://www.acmicpc.net/problem/2873

- 선택한 칸이 첫 두 행에 없다면
- A는 첫 행의 오른쪽으로 갔다가 아래로 한 칸 내려오고 두 번째 칸의 왼쪽으로 이동한 다음, 한 칸 아래로 내려온다
- 이렇게 되면, 위의 두 행을 무시하고 다시 문제를 풀 수 있다.

- 선택한 칸이 마지막 두 행에 없다면
- 위와 같은 식으로 B를 이동시켜
- 마지막 두 행을 무시하고 문제를 다시 풀 수 있다

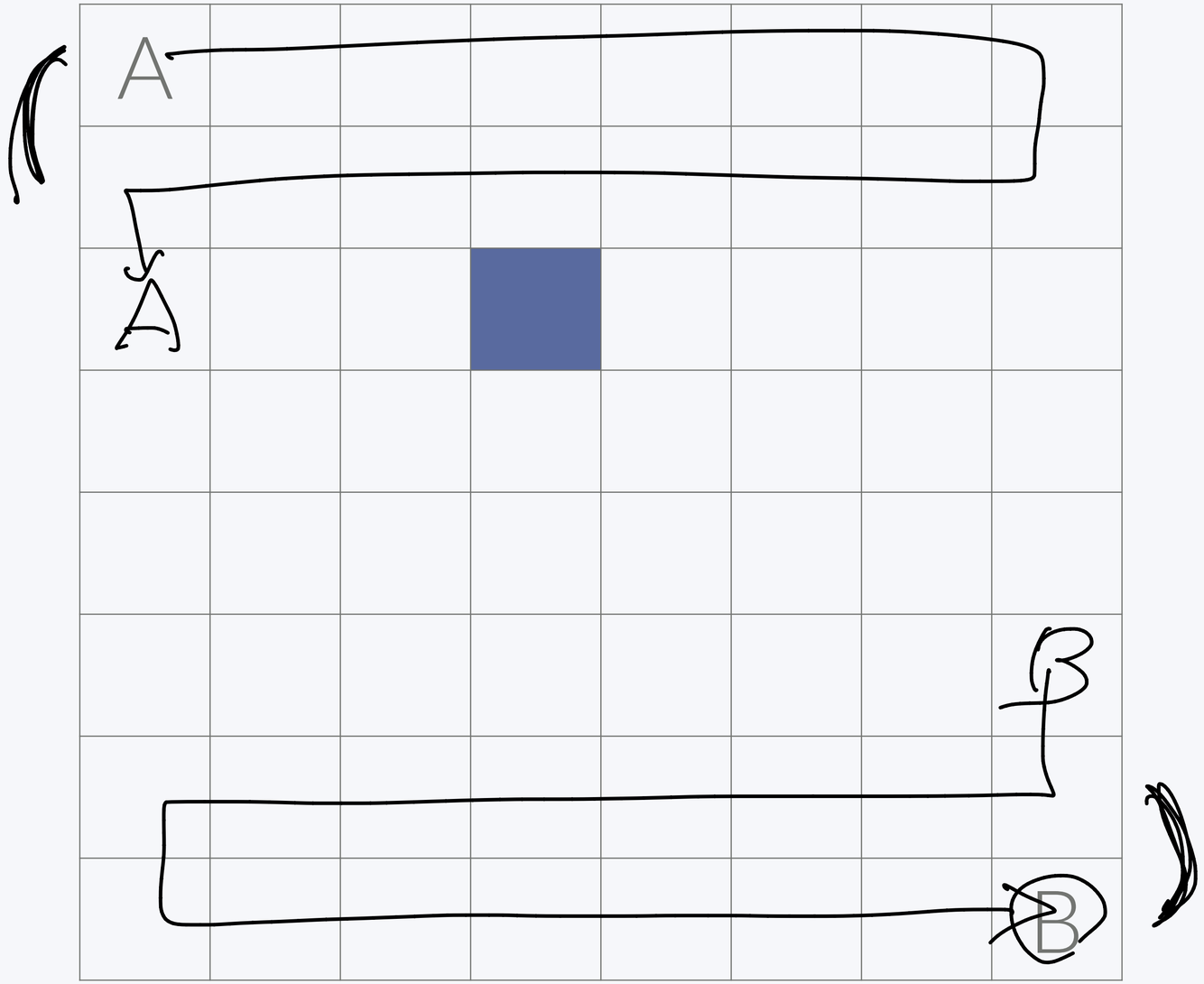
롤러코스터

https://www.acmicpc.net/problem/2873

- 이런식으로 하면, 행은 2개만 남게 되고
- 여기서부터는 열을 행과 같은 식으로 처리하면
- 결국 2*2 크기의 칸만 남게 된다

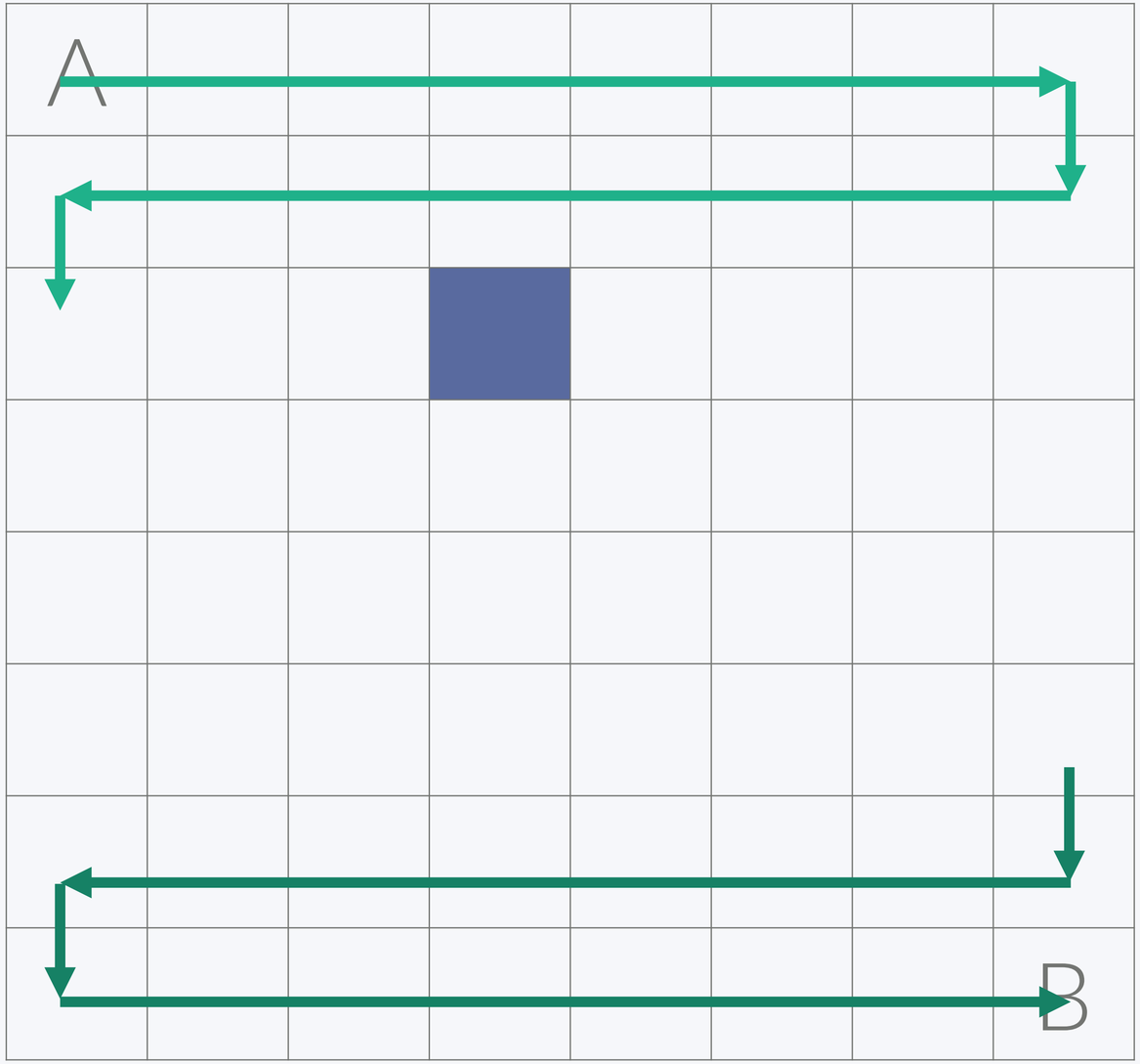
롤러코스터

https://www.acmicpc.net/problem/2873



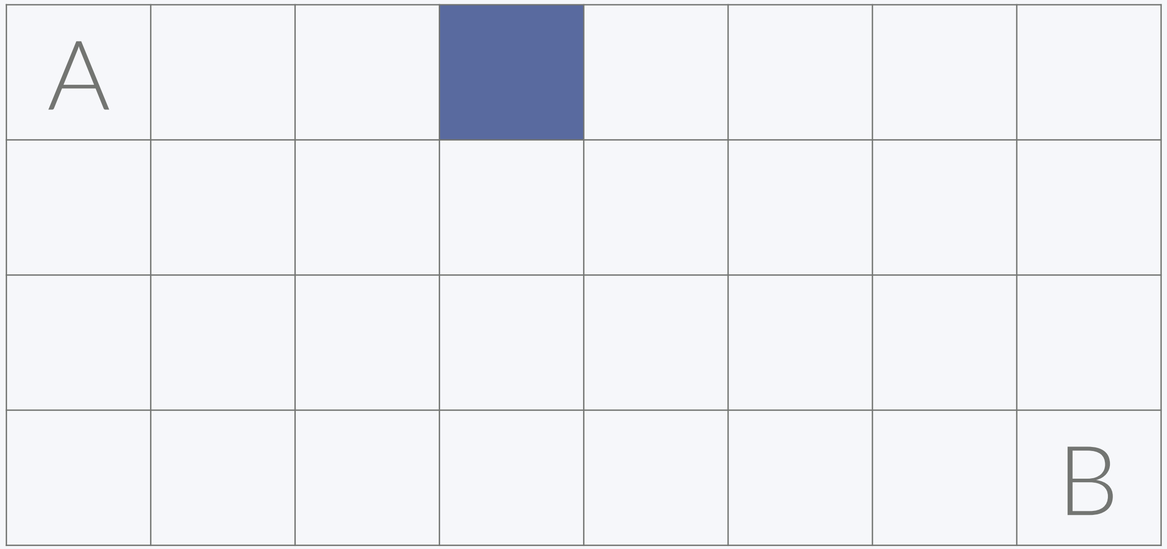
롤러코스터

https://www.acmicpc.net/problem/2873



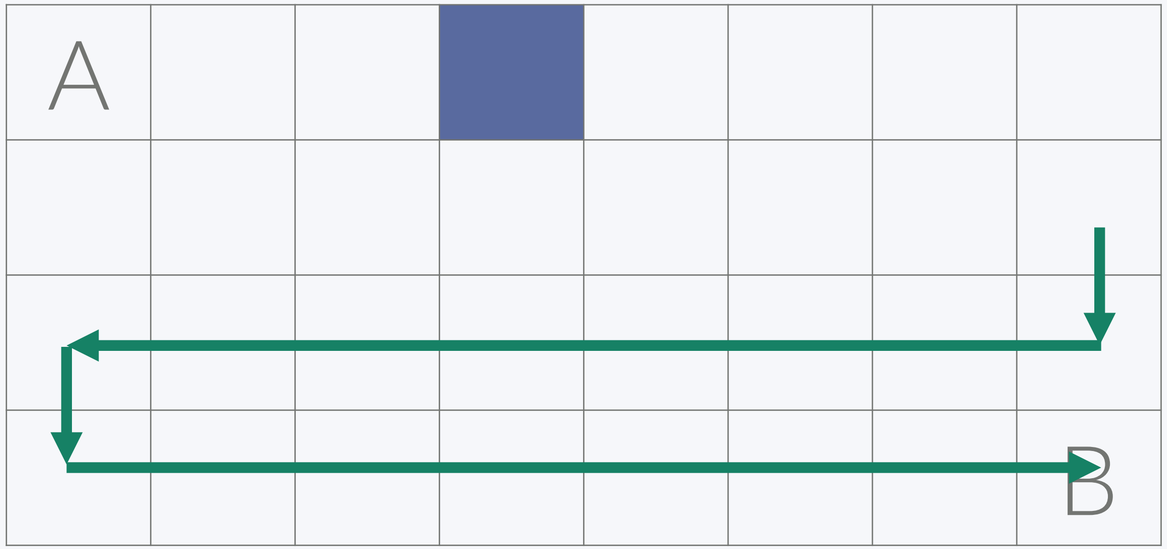
롤러코스터

https://www.acmicpc.net/problem/2873



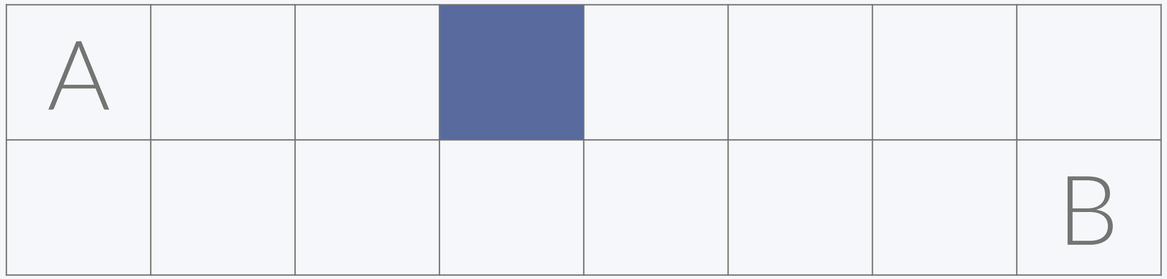
롤러코스터

https://www.acmicpc.net/problem/2873



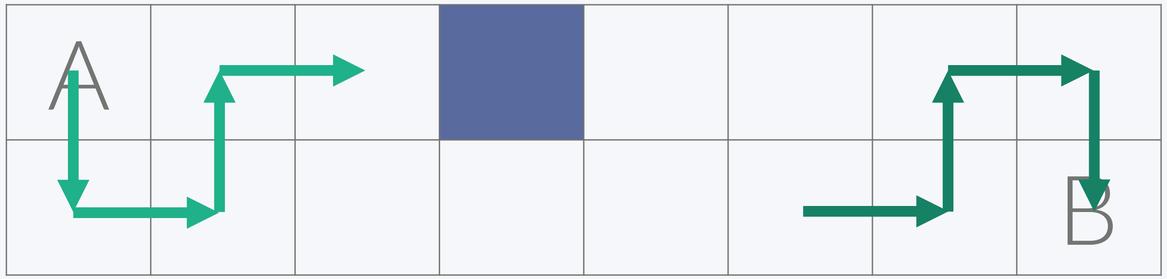
롤러코스터

https://www.acmicpc.net/problem/2873



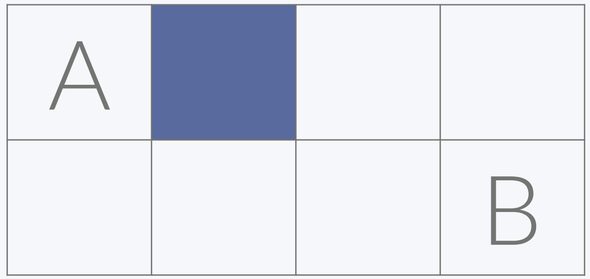
롤러코스터

https://www.acmicpc.net/problem/2873



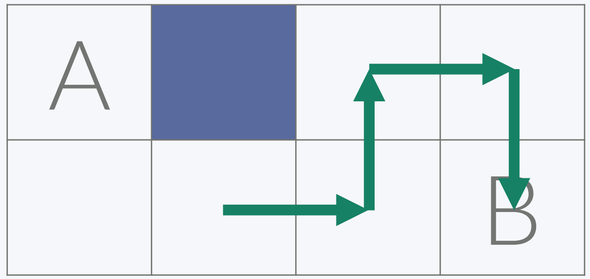
롤러코스터

https://www.acmicpc.net/problem/2873



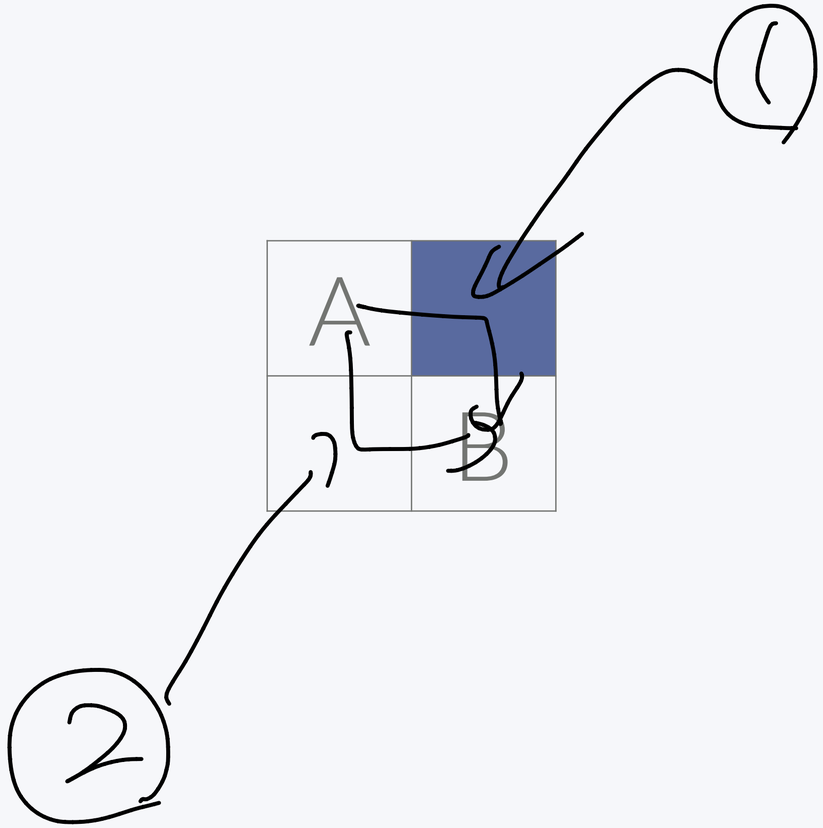
롤러코스터

https://www.acmicpc.net/problem/2873



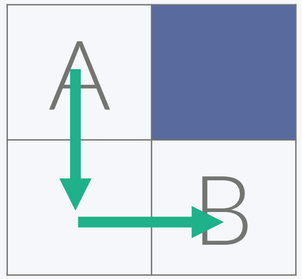
롤러코스터

https://www.acmicpc.net/problem/2873



롤러코스터

https://www.acmicpc.net/problem/2873



롤러코스터

https://www.acmicpc.net/problem/2873

- C++: https://gist.github.com/Baekjoon/0b8da2949403e3c5a676
- Java: https://gist.github.com/Baekjoon/52bac8c27c03181a1cdfcfc145428a8b

보석 도둑

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

- 보석이 N개
- 각 보석은 무게 M[i]와 가격 V[i]를 가지고 있음
- 가방은 K개
- 가방에 담을 수 있는 최대 무게 C[i]
- 가방에는 보석 1개만 넣을 수 있음

- 가방에 담을 수 있는 보석의 최대 가격 구하는 문제

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

보석 도둑

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

1. 숫자를 넣고
2. 어떤 수 x보다 큰 숫자 중에 가장 작은 수를 찾고
3. 숫자를 지운다

보석 도둑

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

- multiset을 사용한다
- C/C++: https://gist.github.com/Baekjoon/5f665ebd54dc9822dfc1

보석 도둑

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

- Max-heap을 이용해서 푸는 방법

보석 도둑

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

- 보석과 가방을 배열 하나에 넣고, 무게를 기준으로 오름차순 정렬한다
  - 보석은 M[i], 가방은 C[i]
- 보석인 경우 힙에 V[i]를 넣는다
- 가방인 경우 힙에서 가장 큰 V[i]를 뺀다

보석 도둑

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

- C/C++: https://gist.github.com/Baekjoon/b381ff404b550814631e