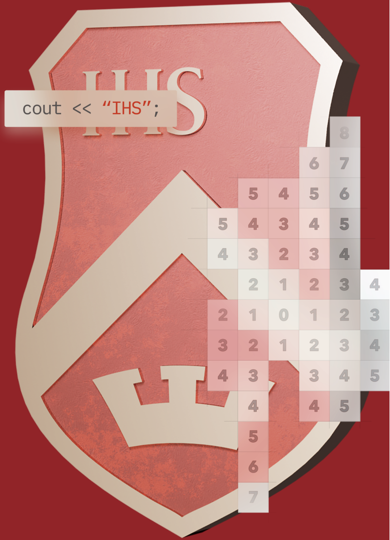
SPC 2023

2023 제19회

서강대학교 프로그래밍 대회

/* Sogang Programming
Contest 2023 */

에디토리얼



주최 Sogang ICPC Team 서강대학교 SOGANG UNIVERSITY

후원 FURIOSA HYUNDAI MOBIS SOLVED.AC STARTLINK

Master

문제		의도한 난이도	출제자
A	Knob	**Beginner**	채성우 lem0nad3
B	donstructive	**Easy**	김동건 dong_gas
C	내 집 마련하기	**Easy**	강효규 djs100201
D	서로소 싫어	**Easy**	강효규 djs100201
E	어? 금지	**Medium**	채성우 lem0nad3
F	Rush & Slash	**Medium**	채성우 lem0nad3
G	순열과 연산	**Hard**	강효규 djs100201
H	같은 풍경	**Hard**	유호영 tkfkddl59323

A. Knob

`implementation`
출제진 의도 – **Beginner**

- 제출 105번, 정답 26명 (정답률 24.762%)
- 처음 푼 사람: **박성준**, 3분
- 출제자: 채성우 `lem0nad3`

- 지문에서 주어진 정보는 다음과 같습니다.
- $l_t = r_t$ 이라면 재미도가 1 오릅니다.
- $t \geq 2$ 인 경우,
  - $l_t = l_{t-1}$ 이라면 재미도가 1 오릅니다.
  - $r_t = r_{t-1}$ 이라면 재미도가 1 오릅니다.
- if 문을 이용하여 답을 구할 수 있습니다.
- fastio를 사용하지 않으면 시간 초과를 받을 수 있습니다.

B. donstructive

`constructive`

출제진 의도 – **Easy**

- 제출 55번, 정답 29명 (정답률 52.727%)
- 처음 푼 사람: **이지훈**, 8분
- 출제자: 김동건 `dong_gas`

- 점수가 가장 높은 순열을 구하는 문제입니다.
- 각 원소가 몇 개의 연속 부분 수열에 속하는지 생각해 봅시다.
- 가운데에 있을수록 더 많은 연속 부분 수열에 속하는 것은 자명합니다.
- 따라서 가운데에 있을수록 더 큰 값을 갖도록 순열을 구성하면 됩니다.
- N 이 4 일때, 정답으로 가능한 순열로는 $[1, 3, 4, 2]$, $[1, 4, 3, 2]$, $[2, 3, 4, 1]$, $[2, 4, 3, 1]$ 이 있습니다.

MC/CA. 내 집 마련하기

`Sorting`

출제진 의도 – **Easy**

- 제출 26번, 정답 19명 (정답률 73.077%)
- 처음 푼 사람: **송근수**, 3분
- 출제자: 강효규 `djs100201`

- 사람들의 번호는 서로 다르다는 것에 집중해 봅시다.
- i 번 사람과 j 번 사람이 배정받은 집을 교환할 수 있고, $j > i$ 라고 가정해 봅시다. 이때 $A_i > A_j$ 를 만족한다면, 두 사람은 집을 교환하는게 이득입니다.
- 사람들의 번호는 서로 다르기 때문에 위의 상황에서 세금의 감면되는 양은 순증가합니다.
- 따라서 최적해에서는 $j > i$ 이면 $A_i \leq A_j$ 를 만족합니다.
- 결론적으로 집을 교환가능한 사람들의 부분 수열을 오름차순 정렬하는 것이 주어진 쿼리의 결과라고 할 수 있고, 유일하게 결정됩니다.
- N 이 300 이하이므로 쿼리마다 나이브하게 정렬해주면 문제가 해결됩니다.

D. 서로소 싫어

math, ad-hoc
출제진 의도 – **Easy**

- 제출 58번, 정답 9명 (정답률 15.517%)
- 처음 푼 사람: **박성준**, 21분
- 출제자: 강효규 djs100201

- x의 배수들은 x와 서로소가 아닙니다.
- 마찬가지로 y의 배수들은 y와 서로소가 아닙니다.
- 따라서 x에서 xy로 간 후, 다시 y로 가면 문제가 해결됩니다.

E. 어? 금지

`dp, binary_search`

출제진 의도 – **Medium**

- 제출 60번, 정답 3명 (정답률 5.000%)
- 처음 푼 사람: **박성준**, 25분
- 출제자: 채성우 `lem0nad3`

DP를 다음과 같이 정의합시다.

$$DP_i = (i\text{번째 시각에 무조건 '어?'를 외칠 때, 최종 혼란의 최댓값})$$

- 이때, 다음 점화식이 성립합니다.
- $DP_i = \max_{t_j < t_i - b_i} (DP_j) + c_i$
- 답은 $\max_{1 \le i \le N} DP_i$ 입니다.
- 하지만 이 방법의 시간복잡도는 $O(N^2)$이므로, 시간 초과가 발생합니다.

PF를 다음과 같이 정의합시다.

$$PF_i = \max_{1 \le j \le i} DP_j$$

- $k = (t_j < t_i - b_i$를 만족하는 j의 최댓값$)$ 이라고 합시다.
- 점화식은 다음과 같습니다.
- $DP_i = PF_k + c_i$
- k는 이분 탐색으로 찾을 수 있습니다.
- $O(N \log N)$에 해결할 수 있습니다.

F. Rush & Slash

`union-find`

출제진 의도 – **Medium**

- 제출 8번, 정답 2명 (정답률 25.000%)
- 처음 푼 사람: **박성준**, 41분
- 출제자: 채성우 `lem0nad3`

- union-find를 이용하여 연결된 잡초를 하나의 그룹으로 합칩시다.
- 이때, 배열 $dist$를 다음과 같이 정의합시다.

$$dist_i = (i\text{번 그룹에 속한 잡초들만 봤을 때, 원점과의 거리 중 최솟값})$$

- 마지막에 방문하는 그룹은 $dist_i$ 만큼 이동해야 합니다.
- 나머지 그룹은 $2 \times dist_i$ 만큼 이동해야 합니다.
- 즉, $dist$가 최대인 그룹을 마지막에 방문하는 것이 이득입니다.
- 답은 $2 \times (\sum dist) - \max(dist)$ 입니다.

G. 순열과 연산

case work, ad-hoc
출제진 의도 – **Hard**

- 제출 7번, 정답 1명 (정답률 14.286%)
- 처음 푼 사람: **박성준**, 73분
- 출제자: 강효규 djs100201

- 2번 연산을 잘 관찰해 문제를 해결해 봅시다.
  - 2번 연산만 수행해서는 A_1과 A_N을 동일하게 만들 수 없습니다.
  - A_i와 A_j가 동일한다면 구간 내 모든 원소가 동일하게 됩니다.
- 따라서 다음과 같은 방법을 생각할 수 있습니다.
- A_1과 A_N을 동일하게 만든 후 2번 연산을 사용합니다.
- 다음과 같은 두 가지 방법이 있습니다.
  - A_1과 A_{N-1}을 동일하게 만든 후 1번 쿼리를 날린다.
  - A_2과 A_N을 동일하게 만든 후 1번 쿼리를 날린다.

- 다음과 같은 3가지 방법 안에 A_1과 A_{N-1}이 같아지거나, A_2가 A_N과 같아집니다.
  - $[2,\ 1,\ N]$ 쿼리를 날린다.
  - $[1,\ 1]$ 쿼리를 날리고 $[2,\ 1,\ N]$ 쿼리를 날린다.
  - $[1,\ N-1]$ 쿼리를 날리고 $[2,\ 1,\ N]$ 쿼리를 날린다.
- 위 내용은 A_1, A_2, A_{N-1}, A_N의 대소 관계를 비교하면 쉽게 증명할 수 있습니다.
- 따라서 전체 문제는 4번의 연산 안에 해결됩니다.

H. 같은 풍경

`geometry`

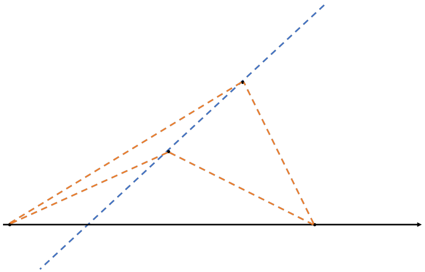
출제진 의도 – **Hard**

- 제출 5번, 정답 1명 (정답률 20.000%)
- 처음 푼 사람: **박성준**, 120분
- 출제자: 유호영 `tkfkdd159323`

- 파노라마로 사진을 찍었을 때 나무가 보이는 순서는 각정렬된 순서와 같습니다.
- 따라서 나무들이 위치한 점들을 각정렬 했을 때 동일하게 정렬되는 점들의 집합을 찾는 문제가 됩니다.
- 먼저, 간단하게 나무가 2개일 때 먼저 관찰해 봅니다.

H. 같은 풍경

- 아래 그림에서 파란 점선은 2개의 나무를 이은 직선입니다.

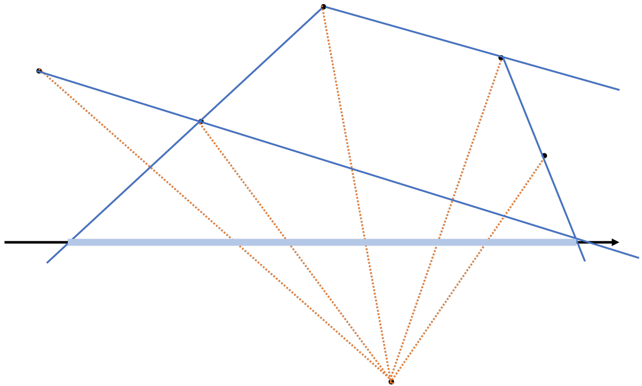


- 파란 점선을 기준으로 왼쪽과 오른쪽에 위치한 두 점에 의한 각정렬 순서는 반대가 되는 걸 알 수 있습니다.
- 반대로 같은 반평면에 위치한 점들은 모두 각정렬 순서가 보존된다는 사실도 쉽게 관찰할 수 있습니다.

- 나무에 대응되는 N 개의 점들을 호영이의 위치에서 각정렬한 순서대로 $a_1, a_2, \cdots, a_N$ 으로 표현한다고 합시다.
- 즉, $a_1 < a_2 < \cdots < a_N$ 으로 표현 가능합니다.
- 만약 우진이의 위치 중 하나의 위치인 b 에서 $a_1 < a_2, a_2 < a_3, \cdots, a_{N-1} < a_N$ 을 만족하면 $a_1 < a_2 < \cdots < a_N$ 을 만족한다는 건 쉽게 보일 수 있습니다. (단, 이는 모든 나무가 반평면 위에 존재하기 때문에 가능한 판단임에 주의합니다.)
- 이전 페이지에서 사용한 방식을 사용해서 호영이의 위치에서 각정렬한 순서대로 2 개씩 점을 이어서 만들어진 직선들로 순서가 보존되는 구간을 만들어 낼 수 있습니다.

H. 같은 풍경

- 지금까지의 풀이대로 하면 아래와 같은 그림이 그려집니다. (x축 아래쪽의 점이 호영이의 위치이며 노란 점선은 각정렬된 순서를 표시하기 위해 그린 보조선입니다.)



- 순서가 보존되는 구간의 최댓값과 최솟값만 구해서 이분탐색을 이용해 해결하면 됩니다.
- 혹은 시간복잡도가 충분하니 brute forcing으로 해당 구간 안에 포함되는 점의 개수를 직접 세도 됩니다.

Champion

문제		의도한 난이도	출제자
A	내 집 마련하기	**Easy**	강효규[djs100201]
B	댄스 타임	**Easy**	정회윤[yunny_world]
C	심심한 마루	**Medium**	정회윤[yunny_world]
D	삼월 초하루	**Medium**	박한나[crescent_h]
E	김밥천국과 도로지옥	**Medium**	유호영[tkfkdd159323]
F	재미없는 문제	**Hard**	채성우[lem0nad3]
G	직각삼각형의 동생은	**Hard**	채성우[lem0nad3]
H	렉시오	**Hard**	유호영[tkfkdd159323]

MC/CA. 내 집 마련하기

Sorting

출제진 의도 – **Easy**

- 제출 26번, 정답 19명 (정답률 73.077%)
- 처음 푼 사람: **송근수**, 3분
- 출제자: 강효규`djs100201`

- 사람들의 번호는 서로 다르다는 것에 집중해 봅시다.
- i 번 사람과 j 번 사람이 배정받은 집을 교환할 수 있고, $j > i$ 라고 가정해 봅시다. 이때 $A_i > A_j$ 를 만족한다면, 두 사람은 집을 교환하는게 이득입니다.
- 사람들의 번호는 서로 다르기 때문에 위의 상황에서 세금의 감면되는 양은 순증가합니다.
- 따라서 최적해에서는 $j > i$ 이면 $A_i \leq A_j$ 를 만족합니다.
- 결론적으로 집을 교환가능한 사람들의 부분 수열을 오름차순 정렬하는 것이 주어진 쿼리의 결과라고 할 수 있고, 유일하게 결정됩니다.
- N 이 300 이하이므로 쿼리마다 나이브하게 정렬해주면 문제가 해결됩니다.

B. 댄스 타임

`math`

출제진 의도 – **Easy**

- 제출 111번, 정답 13명 (정답률 11.712%)
- 처음 푼 사람: **송근수**, 9분
- 출제자: 정회윤 `yunny_world`

- 댄스타임에서는 최대 한 번까지 올바르지 않은 춤을 추더라도 해당 라운드를 통과할 수 있습니다.
- 따라서, 아래와 같이 두 가지 경우로 나누어서 각각의 경우의 수를 구한 후 더해 답을 구하면 됩니다.

1. 모든 라운드를 통과한 경우
2. 오직 한 번의 라운드에서만 통과하지 못하고, 나머지 라운드에서 통과한 경우

    2.1 앞을 봐서 통과 못한 경우

    2.2 뒤를 봐서 통과 못한 경우

- 모든 라운드를 통과하는 경우의 수를 구하기 앞서, 한 라운드를 통과하거나, 통과하지 못하는 경우의 수를 구해봅시다.
- 열은 우진이 춤을 출 때 보는 방향, 행은 라운드 통과 여부를 나타냅니다.

	앞	뒤
통과함	1	$M-1$
통과하지 못함	$M-1$	1

- 이제 모든 라운드에 대한 경우의 수를 구해봅시다.
- 모든 라운드 중 우진이 뒤를 보고 춤을 춘 횟수를 C 라 하겠습니다.

1. $(M-1)^C$ 개
2. 아래 두 경우의 수의 합

   2.1 $(M-C)(M-1)^{C+1}$ 개

   2.2 $C \geq 1$ 이면 $C(M-1)^{C-1}$ 개, 아니면 0 개

- 위에서 구한 경우의 수를 모두 합한 값이 답입니다.
- Integer overflow에 주의하여 모든 연산 과정에서 소수 $1\,000\,000\,007$ 로 나눈 나머지로 연산해야 하는 것에 주의합시다.

C. 심심한 마루

`math`, `prefix_sum`
출제진 의도 – **Medium**

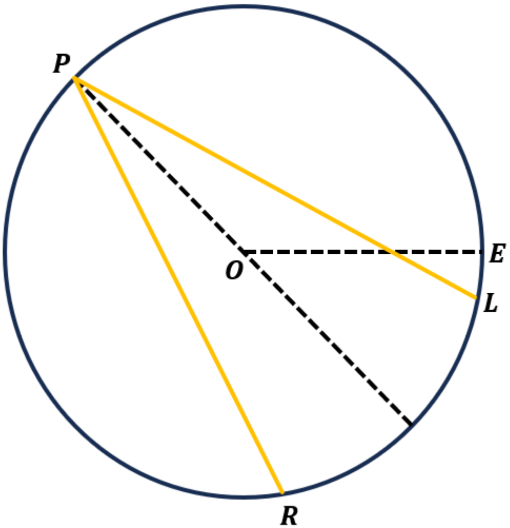
- 제출 54번, 정답 11명 (정답률 20.370%)
- 처음 푼 사람: **송근수**, 16분
- 출제자: 정회윤 `yunny_world`

C. 심심한 마루

- 이 문제는 여러 개의 범위가 주어질 때 한 번 이상 범위에 포함된 센서의 개수를 구하는 문제입니다.
- 이는 **imos법**을 이용하여 해결할 수 있습니다.

1. $P[i]$를 입구로부터 벽을 따라 반시계 방향으로 $i°$ 만큼 이동한 곳에 위치한 센서의 블레이즈 감지 여부라고 합시다.
2. 어느 한 번의 지시로 센서가 s부터 e까지의 범위 블레이즈를 감지할 때, $P[s]$에 1을, $P[e+1]$에 -1을 더해줍니다.
3. 모든 지시에 대해 위와 같은 방식으로 처리합니다.
4. $P[i]$의 모든 범위에 대해 누적합을 구했을 때, $P[i]$가 1보다 크거나 같으면 블레이즈를 감지함, 0이면 블레이즈를 감지하지 못했음을 뜻하게 됩니다.
5. 모든 i에 대하여 $P[i] \geq 1$인 i의 개수를 출력하면 됩니다.

C. 심심한 마루



- 어느 한 번의 지시로 인한 블레이즈의 범위는 원주각과 중심각의 관계를 이용해 구할 수 있습니다.
  - 점 P를 점 O에 대해 대칭시킨 점을 점 M라 합시다.
  - 중심각은 원주각의 2배이고 $\angle LPR = b^\circ$이므로 $\angle LOM = \angle ROM = b^\circ$입니다.
  - 따라서, 점 M, L, R에 해당 하는 위치는 각각 $a + 180, a + 180 - b, a + 180 + b$로 표현할 수 있습니다.
- 센서는 원을 이루며 위치해 있으므로 나머지 연산을 취하여 각각의 위치를 0보다 크거나 같고, 359보다 작거나 같은 위치로 대응시킨 후 **imos법**을 적용하면 답을 구할 수 있습니다.

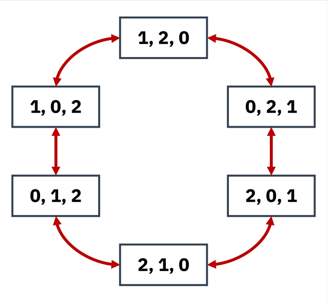
D. 삼월 초하루

`constructive`
출제진 의도 – **Medium**

- 제출 31번, 정답 6명 (정답률 19.355%)
- 처음 푼 사람: **송근수**, 30분
- 출제자: 박한나 `crescent_h`

- 이 문제는 1 번부터 N 번까지의 숙우에서 물 별로 이동해야 하는 횟수가 주어진 것으로 보아도 됩니다.
- 또한 한 숙우에서 다른 숙우로 물이 이동하는 것은 **두 물을 바꾸는 것**과 같습니다.

- 우선 가능한 케이스와 불가능한 케이스를 분류해야 합니다.
- 각 숙우에 담긴 물의 번호를 길이 3의 순열로 나타내면 총 6가지가 가능합니다.
- 물을 한 번 이동할 때 이 순열이 어떻게 바뀌는지 그래프로 그려보면 다음과 같습니다.



- 물을 이동하는 것은 순열의 두 원소를 바꾸는 것과 동일하므로 순열의 **홀짝성**이 바뀝니다.
- 시작 순열 $1, 2, 0$은 짝순열이므로 물을 짝수 번 이동하면 **짝순열**, 홀수 번 이동시 **홀순열**입니다.
- 따라서 목표 순열이 짝순열인데 물을 홀수 번 이동해야 하거나 그 반대인 경우에는 목표를 달성할 수 없습니다.
- 그리고 물마다, 목표 순열에서의 위치가 초기 순열과 다르면서 이동해야 하는 횟수가 0인 경우, 혹은 목표 순열에서의 위치가 초기 순열과 같으면서 이동해야 하는 횟수가 1인 경우도 불가능합니다.
- 그외의 경우는 모두 가능합니다.

D. 삼월 초하루

- 물 1 이 이동해야하는 횟수 n_1 은 $(100-t_1)/5$, 물 2 의 이동 횟수 n_2 는 $(100-t_2)/5$ 로 구할 수 있습니다. 총 이동 횟수 K 는 n_1+n_2 입니다.
- 예를 들어 목표 순열이 $2, 0, 1$ 이고 n_1 이 홀수이면 다음과 같은 과정으로 목표에 도달할 수 있습니다.
  - 물 1 을 빈 숙우로 옮기고, 이어서 물 2 를 빈 숙우로 옮깁니다.
  - 그 다음 물 1 을 빈 숙우로 n_1-1 번 옮기고 물 2 를 n_2-1 번 옮깁니다.
- 나머지 경우에도 동일한 방법으로 과정을 구성할 수 있습니다.
- 이외에도 너비 우선 탐색과 같은 완전 탐색을 이용하는 풀이도 있습니다.

E. 김밥천국과 도로지옥

`dijkstra`
출제진 의도 – **Medium**

- 제출 9번, 정답 2명 (정답률 22.222%)
- 처음 푼 사람: **송근수**, 50분
- 출제자: 유호영 `tkfkdd159323`

- 연결된 간선의 종류와 관계없이, 어떤 점이든 도달하고 난 후 12분 후에는 원래 점으로 돌아올 수 있습니다.
- 따라서 x분에 특정 정점에 도달했다면 $x+12, x+24, \cdots, x+12\times k$분 후에는 항상 해당 점에 올 수 있습니다.
- 도달한 시각을 12로 나눈 나머지를 이용해 각 정점을 12개로 분할해서 다익스트라를 하면 문제가 쉽게 풀립니다.

F. 재미없는 문제

`constructive`, `ad-hoc`
출제진 의도 – **Hard**

- 제출 16번, 정답 2명 (정답률 12.500%)
- 처음 푼 사람: **송근수**, 50분
- 출제자: 채성우 `lem0nad3`

두 경우로 나누어 해결하면 됩니다.

1. $M \le N - 1$ 인 경우
   - M 개의 1, $(N - M)$ 개의 0 으로 a 를 구성하면 됩니다.
2. $M \ge N$ 인 경우
   - $a_1 = 0$ 으로 둡니다.
   - $k = \left\lfloor \frac{N}{2} \right\rfloor$ 이라고 합시다.
   - $a_2 = a_3 = \cdots = a_{k+1} = 1$ 으로 채웁니다.
   - a_{k+2} 부터 $k + 1$ 을 채우다가, sum$(a) \ge M$ 이 되는 순간 멈춥니다.
   - 넘치는 값은 빼주고, 나머지는 0 으로 채우면 됩니다.

F. 재미없는 문제

이게 왜 될까요?

- $M \le N - 1$ 인 경우는 자명하니, $M \ge N$ 인 경우만 확인하겠습니다.
- 1 부터 $(k+1)$ 까지는 쉽게 만들 수 있습니다.
- a_{k+2} 를 오른쪽 끝점으로 두고, 왼쪽 끝점을 움직여서 $(k+1)$ 부터 $(2k+1)$ 까지 만들 수 있습니다.
- $(2k+2)$ 는 $(a_{k+2} + a_{k+3})$ 으로 만들 수 있습니다.
- 또다시 왼쪽 끝점을 움직여서 $(2k+2)$ 부터 $(3k+2)$ 까지 만들 수 있습니다.
- 이를 반복해서 M 까지의 모든 수를 만들 수 있습니다.
- 더 정확히, N 이 홀수면 $\dfrac{N^2+2N-3}{4}$, 짝수면 $\dfrac{N^2+2N-4}{4}$ 까지 만들 수 있고, 이는 M 제한에 충분히 들어옵니다.

G. 직각삼각형의 동생은?

segtree, offline Query
출제진 의도 – **Hard**

- 제출 19번, 정답 2명 (정답률 10.526%)
- 처음 푼 사람: **이승형**, 143분
- 출제자: 채성우 lem0nad3

각 쿼리는 독립적이므로, 이를 순서대로 처리할 필요는 없습니다.

- ✓ $(N + M)$ 개의 점을 원점을 중심으로 하여, 반시계 방향으로 정렬합시다.
- ✓ 만약 두 점의 기울기가 같은 경우에는 검은색 점이 앞에 오도록 정렬합시다.
- ✓ 이제 우리는 원점을 지나며 기울기가 양수인 어떤 직선 아래의 점들을 관리할 수 있습니다.
- ✓ 세그먼트 트리를 이용합시다.
- ✓ seg_i 는 해당 노드가 담당하는 x 좌표 구간에 속하는 검은색 점들의 개수입니다.
- ✓ 이때 x 좌표의 최댓값이 10^9 이므로, 좌표 압축을 해야 합니다.

- 편의 상 쿼리에서 주어지는 점을 **흰색 점**이라고 하겠습니다.
- 만약 현재 확인 중인 점이 검은색 점이라면 해당 x 좌표에 update하면 됩니다.
- 흰색 점이라면 구간 $[1, x]$ 에 속하는 검은색 점들의 개수를 구하면 됩니다.
- 전체 시간복잡도는 $O(N \log N)$ 입니다.

H. 렉시오

`dp, prefix_sum`
출제진 의도 – **Hard**

- 제출 4번, 정답 0명 (정답률 00.000%)
- 처음 푼 사람: **???**, ?분
- 출제자: 유호영 `tkfkdd159323`

- 이 문제는 2023 서강대학교 청정수컵의 $K512$ 에서 피자먹기 라는 문제의 강화판입니다.
- 누적합과 관련된 아이디어는 해당 문제에서 누적합을 사용하는 이유와 거의 동일하니 참고하면 좋을 것 같습니다.
- 이 문제의 첫번째 진입장벽인 정산 결과 실제로 각 사람이 얻거나 잃게 되는 점수만 고려하면 된다는 점은 문제에서 주어지므로 받아들이고 넘어가면 됩니다.
- 정산 결과 각 사람이 얻거나 잃게 되는 점수 $score[i]$ 는 남은 카드의 총 개수를 $total$ 이라고 했을 때 $total - card[i] \times n$ 으로 쉽게 계산할 수 있습니다.

- 카드를 교환할 때 n명의 사람들의 $score$의 합이 0이라고 하면 $n-1$번의 거래로 항상 정산을 마칠 수 있습니다. 간단하게, 뽑은 n명의 사람들에게 번호를 매기고 1번 사람이 각 i번 사람과 $score[i]$에 해당하는 교환을 하면 자연스럽게 정산이 완료되는 모습을 생각하면 됩니다.
- 반대로 n명의 사람들이 어떤 크기가 n보다 작은 부분집합을 이루어도 $score$의 합을 0으로 만들 수 없다면 $n-1$번보다 적은 횟수의 교환으로 정산하는 것이 불가능합니다.
- 그래프 이론을 이용하여 다음 장에서 간단하게 증명해 보겠습니다.

- 각 사람을 정점으로 취급하고, 교환을 하나의 간선으로 취급합니다. 이렇게 그래프를 만들었을 때, 사이클이 생긴다면 항상 없애줄 수 있습니다. 따라서 사이클이 있는 그래프는 최적이 아니며, 우리는 항상 트리를 만들 수 있습니다.
- 연결된 노드들의 $score$의 합이 0이면 간선을 따라 한번씩 교환해서 해당 사람들끼리는 정산이 가능합니다. 만약 합이 0이 아니면 외부의 다른 집단과 연결할 새로운 간선이 필요하게 됩니다.
- 트리의 간선의 수는 항상 $n-1$이므로 많아봤자 $n-1$번의 교환으로 정산 가능하다는 것은 보여졌습니다.
- 반대로 더 적은 간선으로는 그래프의 모든 정점을 연결시킬 수 없습니다. 따라서 적어도 $n-1$번의 교환이 필요하다는 것도 보여졌습니다.

- 다행히 이 문제는 간선의 구조가 매우 간단합니다. $score$의 누적합이 0이 될때마다 배열을 끊어주면 각각의 연결된 배열들이 합이 0이 되므로 이웃한 사람들끼리만 교환했을 때의 교환의 최솟값을 쉽게 구할 수 있습니다.
- 임의의 한 쌍의 사람을 골라서 교환을 시키는 경우도 누적합의 관점으로 관찰하면 어렵지 않게 해결할 수 있습니다.
- $i < j$인 i, j를 골랐다고 가정하고 j번 사람이 i번 사람에게 x만큼의 점수를 줬다고 생각합시다.(반대로 점수를 받았다면 x를 음수 취급하면 됩니다.) 그러면 $score[i]$를 포함해 $score[j]$ 이전 까지의 누적합은 전부 x만큼 커졌다고 생각할 수 있습니다. 그리고 그 j뒤는 원래대로 돌아옵니다.
- 변화한 상태에서 다시 누적합이 0이 될 때마다 배열을 끊어주면 됩니다.

- 아래 그림은 2번 예제에서 i가 0, j가 5로 선택되고 x는 -10인 경우입니다. (주어진 배열인 $card[]$로 부터 $score[]$를 만들고, 다시 누적합을 한 배열임에 유의합니다.)

10	10	0	10	10	10	0	20	30	0

⬇

0	0	-10	0	0	0	0	20	30	0

- 이때는 0의 개수가 총 7개 이므로, 실제로 이뤄지는 교환 횟수는 $10-7=3$이 됩니다. 실제 답은 i, j의 교환까지 추가해 4가 됩니다.
- 이때 특정 연속 구간의 값을 x만큼 늘리면 전체 0의 개수는 $(($해당 구간 내에서 $-x$의 등장 횟수$)-(0$의 등장 횟수$))$ 만큼 이득을 본다는 점을 쉽게 파악할 수 있습니다.

- 이제 i와 j를 먼저 선택하지 말고, x를 먼저 선택하는 방식으로 생각을 바꾸면, 우리는 누적합 배열에서 $-x$는 1로, 0은 -1로, 그 외의 수는 0으로 바꾼 배열에서 카데인 알고리즘을 수행하여 x에 따른 교환 횟수 감소의 최댓값을 구할 수 있다는 걸 알 수 있습니다.
- 누적합 배열에 등장하는 모든 수에 대해 위와 같은 방식으로 최적값을 구하면 되는데, 그냥 진행하면 $O(n^2)$이 되므로 적당한 최적화가 필요합니다.
- 좌표압축을 사용해 여러개의 카데인을 병렬로 수행하거나 잘 정렬해서 같은 값을 갖는 부분만 추출해서 따로따로 카데인을 해주면 $O(n \log n)$으로 최적값을 전부 구할 수 있으며, 그 값을 이용해 답을 구하면 됩니다.
- 단, map과 unordered_map을 사용하는 풀이는 다른 풀이와 실행시간 차이가 너무 커서 다른 시간복잡도의 풀이가 풀리는 이유로 사용을 제한했습니다.

- 다음은 정렬을 이용해 카데인을 하는 예시입니다.
- 아래 2개의 배열 중 위쪽 배열은 누적합 배열에 대해 (값, 인덱스) 형태로 저장해 정렬한 $sorted\_psum[]$ 배열로, 0은 사전에 제외해도 되고 안해도 됩니다.
- 그리고 아래쪽 배열은 0의 개수를 누적해 놓은 배열 $zero\_psum[]$ 입니다.

(10, 0)	(10, 1)	(10, 3)	(10, 4)	(10, 5)	(20, 7)	(30, 8)

0	0	1	1	1	1	2	2	2	3

- $sorted_psum$을 차례대로 보면서 같은 값에 대해서만 카데인을 해주고, 서로 같은 값 사이의 0의 개수는 $zero_psum$을 이용해 구하면 됩니다.