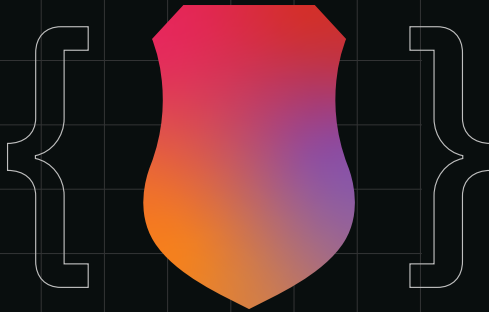
SPC 2022

/* Sogang Programming Contest 2022 */

주최 Sogang ICPC Team 서강대학교 SOGANG UNIVERSITY



2022 서강대학교 프로그래밍 대회 해설

11월 26일 토요일
오후 2 → 6시, 다산관 D104/105

후원 HYUNDAI MOBIS FURIOSA common computer STARTLINK

Master

문제		의도한 난이도	출제자
A	WARBOY	**Beginner**	김동건 dong_gas
B	유통기한	**Easy**	박수현 shiftpsh
C	DPS	**Easy**	김동건 dong_gas
D	효구와 호규 (Easy)	**Medium**	조원빈 wbcho0504
E	어려운 스케줄링	**Medium**	임지환 raararaara
F	피보나치와 마지막 수열과 쿼리	**Hard**	임지환 raararaara
G	트리 다듬기	**Hard**	김동건 dong_gas
H	완전한 수열 리버스	**Hard**	강효규 djs100201

A. WARBOY

`arithmetic`
출제진 의도 – **Easy**

- 제출 68번, 정답 51명 (정답률 75.000%)
- 처음 푼 사람: **채성우**, 0분
- 출제자: 김동건`dong_gas`

A. WARBOY

- 경쟁사 제품의 가격을 A, 경쟁사 제품의 성능을 B, WARBOY의 가격 C, WARBOY의 성능을 D라 합시다.
- A, B, C는 문제에서 주어집니다.
- 문제에서 WARBOY의 가격 대비 성능이 경쟁사 제품의 가격 대비 성능의 3배라고 하였으므로 아래와 같은 식을 세울 수 있습니다.

$$3 \times \frac{B}{A} = \frac{D}{C}$$

- 구하고자 하는 D에 관하여 식을 정리하면 아래와 같습니다.

$$D = 3 \times C \times \frac{B}{A}$$

- 계산 과정에서 C를 A로 먼저 나누고 계산하지 않도록 주의합시다.

B. 유통기한

`implementation`
출제진 의도 – **Easy**

- 제출 263번, 정답 21명 (정답률 7.985%)
- 처음 푼 사람: **채성우**, 13분
- 출제자: 박수현`shiftpsh`

B. 유통기한

A, B, C를 입력받아서 우선 다음 세 가지 경우가 올바른 날짜인지 판별합니다.

- ✓ A년 B월 C일
- ✓ C년 A월 B일
- ✓ C년 B월 A일

올바른 날짜가 없다면 `invalid`를 출력하고 다음 경우로 넘어갑니다. 윤년에 주의합시다.

B. 유통기한

위에서 올바른 날짜로 판별해낸 날짜들에 대해, 오늘 날짜와 비교합니다. 여러 가지 방법이 있습니다.

- ✓ $10\,000Y + 100M + D$를 비교하는 방법
- ✓ `std::tuple`을 사용해 (Y, M, D)를 비교하는 방법
- ✓ 조건문 세 번 사용하여 비교하는 방법

올바른 날짜들 중 하나라도 오늘 날짜 이전인 것이 있다면 `unsafe`입니다. 모든 날짜가 오늘 혹은 오늘 이후라면 `safe`입니다.

C. DPS

`mathmatics`, `combinatorics`

출제진 의도 – **Easy**

- 제출 209번, 정답 36명 (정답률 17.703%)
- 처음 푼 사람: **조성훈**, 29분
- 출제자: 김동건`dong_gas`

C. DPS

- ✓ 문제를 읽어보면 각 핸들의 첫 글자만 필요하다는 것을 쉽게 알 수 있습니다.
- ✓ 삼중 for문으로 문제를 풀면 시간초과를 받게 됩니다.
- ✓ 따라서 각 알파벳 별로 해당 알파벳으로 시작하는 핸들의 개수를 세어둡니다.
- ✓ 입력으로 주어지는 팀 명은 아래와 같은 세 가지 경우로 나눌 수 있습니다.
  - 세 알파벳이 모두 같은 경우
  - 세 알파벳 중 두 알파벳이 같고 나머지 하나의 알파벳은 다른 경우
  - 세 알파벳이 모두 다른 경우

- ✓ 세 알파벳이 모두 같은 경우
  - 해당 알파벳으로 시작하는 핸들의 개수를 a라 하면 $\binom{a}{3}$으로 한 번에 계산할 수 있습니다.
- ✓ 세 알파벳 중 두 알파벳이 같고 나머지 하나의 알파벳은 다른 경우
  - 같은 두 알파벳을 A, 나머지 하나의 알파벳을 B라 합시다.
  - A로 시작하는 핸들의 개수를 a, B로 시작하는 핸들의 개수를 b라 하면 $\binom{a}{2} \times b$로 한 번에 계산할 수 있습니다.

- 세 알파벳이 모두 다른 경우
  - 서로 다른 세 알파벳을 각각 A, B, C라 합시다.
  - A로 시작하는 핸들의 개수를 a, B로 시작하는 핸들의 개수를 b, C로 시작하는 핸들의 개수를 c라 하면 $a \times b \times c$로 한 번에 계산할 수 있습니다.

- 참고로 DPS는 2022 ICPC Seoul Regional에서 사용한 저희 팀의 이름입니다.

D. 효구와 호규 (Easy)

ad_hoc

출제진 의도 – **Medium**

- 제출 97번, 정답 22명 (정답률 22.680%)
- 처음 푼 사람: **조성훈**, 29분
- 출제자: 조원빈 wbcho0504

D. 효구와 호규(Easy)

아래의 2가지 경우에는 모든 카드를 없앨 수 없습니다.

- ✓ 0 또는 1이 적힌 카드의 개수가 홀수인 경우
- ✓ 0과 1이 적힌 카드가 체스판의 형태로 놓여 있는 경우

0	1	0
1	0	1
0	1	0

- ✓ 그 외의 경우에는 모든 카드를 없앨 수 있습니다.

D. 효구와 호규(Easy)

카드에 적힌 숫자가 같으면서 인접한 임의의 두 카드를 골라 없앱니다.

0	1	0	1
1	0	1	1
0	1	1	0
0	0	1	0

D. 효구와 호규(Easy)

없앤 카드와 인접한 카드들을 후보군에 넣어줍니다.

0	1	0	1
1	0	1	1
	1	1	0
	0	1	0

D. 효구와 호규(Easy)

여기서 후보군의 카드는 3개 이상이므로 비둘기 집의 원리에 의해 똑같은 숫자를 가진 카드가 적어도 2개 존재합니다.

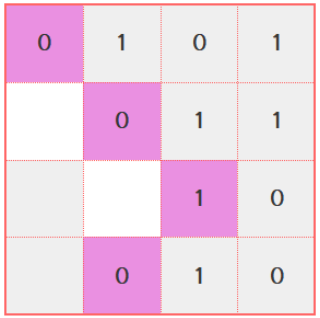
0	1	0	1
1	0	1	1
	1	1	0
	0	1	0

D. 효구와 호규(Easy)

후보군의 카드들 중 숫자가 같은 두 카드를 없앱니다.

0	1	0	1
1	0	1	1
	1	1	0
	0	1	0

위의 과정을 모든 카드를 없앨 때까지 반복합니다.



여기까지가 의도한 풀이입니다. 그러나 각 차례에 없애는 두 개의 카드 외에 다른 카드도 움직일 수 있으므로 다음의 풀이도 성립합니다.

- ✓ 첫 번째 턴에는 인접하고 숫자가 같은 두 카드를 없앱니다.
- ✓ 두 번째 턴부터는 적당히 이동하여 모든 카드를 인접하게 만들 수 있습니다.
- ✓ 따라서 두 번째 턴부터는 숫자가 같은 임의의 두 카드를 없앨 수 있습니다.

D. 효구와 호규(Easy)

앞서 설명한 방법을 통해 open contest에 출제될 효구와 호규(Hard) 문제도 해결할 수 있습니다.

E. 어려운 스케줄링

`deque`, `sorting`

출제진 의도 – **Medium**

- 제출 109번, 정답 10명 (정답률 9.174%)
- 처음 푼 사람: **채성우**, 43분
- 출제자: 임지환[raararaara]

E. 어려운 스케줄링

Master Division은 추가, 정렬, 뒤집기 연산만 주어지고 최종 결과에 대하여 앞에서 k번째에 해당하는 업무의 고유번호를 구하는 문제입니다.
Champion Division에서는 반복적으로 가장 앞에 있는 업무의 고유번호를 출력하는 연산이 추가되었습니다.

E. 어려운 스케줄링

반복적인 정렬 연산과 뒤집기 연산이 등장합니다. 매번 정렬을 하고 뒤집기를 수행하면 약 $\mathcal{O}(QN \log N)$ 의 시간복잡도로 시간초과를 받게 됩니다.

다음과 같은 사항을 고려해야 합니다.

- ✓ 매번 정렬을 해야 하는지
- ✓ 매번 뒤집기 연산을 해야 하는지
- ✓ 정렬과 뒤집기가 발생하는 상황에서 업무 추가 연산은 어떻게 해야 하는지

정렬이 없다면 문제는 쉽습니다. 덱(deque) 등을 활용하여 뒤집기 연산이 짝수 번(0번 포함) 발생했을 경우는 덱의 앞에, 홀수 번 발생했을 경우는 덱의 뒤에 넣고, 뒤집기 연산의 최종 발생 횟수에 따라 앞에서부터 k번째인지, 뒤에서부터 k번째인지를 구하면 됩니다.

매번 정렬을 할 필요가 있을까요?

뒤에 일어난 정렬 연산에 의해 앞서 일어난 정렬 연산 및 뒤집기 연산은 아무 상관이 없어집니다.
즉, 가장 마지막에 일어난 정렬 연산 이전까지 업무 추가 연산만 적용한 후 정렬 연산을
수행합니다.

E. 어려운 스케줄링

정렬에 대한 문제가 해결되었습니다. 이후에 발생하는 업무 추가 연산과 뒤집기 연산은 정렬된 리스트 상에 추가로 원소를 붙이면 됩니다.

업무 추가 연산이 주어졌을 때, 그 전까지의 뒤집기 연산의 발생 횟수가

- ✓ 짝수 번(0번 포함)이라면 리스트의 앞쪽에 업무 추가
- ✓ 홀수 번이라면 리스트의 뒤쪽에 업무 추가

하는 방식으로 해결할 수 있습니다.

마찬가지로 덱(deque)을 활용하면 리스트의 앞과 뒤에 원소 추가를 $O(1)$에 수행할 수 있습니다.
정렬된 리스트를 덱에 넣고 이후 발생하는 뒤집기 연산, 업무 추가 연산에 대해 적절히 덱의 앞, 또는 뒤에 넣어주면 됩니다.

우선순위 큐를 이용하여 문제를 해결할 수도 있습니다.

마지막 정렬 연산 이전까지의 원소들을 우선순위 큐로 관리하고, 뒤집기 연산의 적용 횟수가 홀수인지 짝수인지에 따라 앞에 들어갈 업무들을 관리하는 배열과 뒤에 들어갈 업무들을 관리하는 배열을 각각 만들면 됩니다.

F. 피보나치와 마지막 수열과 쿼리

`offline queries, disjoint set`

출제진 의도 – **Hard**

- ✓ 제출 54번, 정답 3명 (정답률 5.556%)
- ✓ 처음 푼 사람: **조성훈**, 117분
- ✓ 출제자: 임지환 `raararaara`

문제에서 요구하는 결과는 모든 쿼리가 순서대로 적용되었을 때의 결과입니다. 즉, 뒤에 적용된 쿼리의 결과가 덧씌워집니다.

F. 피보나치와 마지막 수열과 쿼리

- 결과가 덧씌워짐을 방지하기 위해 주어진 쿼리를 역순으로 처리해봅시다.
- 동시에, 직전까지 처리된 쿼리들에 의해 값이 쓰여진 곳은 최종 결과가 되며, 이는 쿼리 처리 과정에서 건들면 안되는 칸이 됩니다.

쿼리 처리 과정에서 건들면 안되는 칸에 대한 처리를 어떻게 할까요?

BOJ 10775의 Disjoint Set Union 풀이와 유사하게 처리할 수 있습니다.
값이 쓰여져있는 연속구간에 있는 모든 칸들의 대표값을 해당 연속구간의 오른쪽 끝 인덱스로 설정하여 관리합니다.

이를 통해 각각의 인덱스에 대한 접근을 최대 1번으로 제한할 수 있고, 약 $O(N)$ 의 시간복잡도로 처리할 수 있습니다.

F. 피보나치와 마지막 수열과 쿼리

수열의 범위가 작다면 건들면 안되는 칸에 대한 관리를 std::set등으로 할 수 있겠지만, 이 풀이를 막기 위해 노력했습니다.

G. 트리 다듬기

`tree`, `greedy`
출제진 의도 – **Hard**

- 제출 9번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: **X**
- 출제자: 김동건`dong_gas`

G. 트리 다듬기

- 트리에서 정점 t를 루트로 하는 서브트리를 떼어낸 후, t를 기존 트리와 다시 연결하는 것을 문제에서 작업이라고 합니다.
- 우리는 작업을 K번 이하로 하여, 트리의 지름을 가장 길게 만들고 싶습니다.
- 즉, 작업을 통해 트리를 일자형 트리에 가깝게 만들어야 합니다.
- 작업 기회가 남아있는데, 이미 일자형 트리라면 작업을 더 하지 않아도 됩니다.

G. 트리 다듬기

- 여기까지 생각해보면 작업을 통해 서브트리를 지름의 끝에 붙여 지름을 늘리고 싶다는 생각이 들게 됩니다.
- 과연 이 생각은 정당할까요?
- 작업을 한 번만 한다면 당연히 한 번의 작업으로 지름을 늘리는 것이 이득입니다.

G. 트리 다듬기

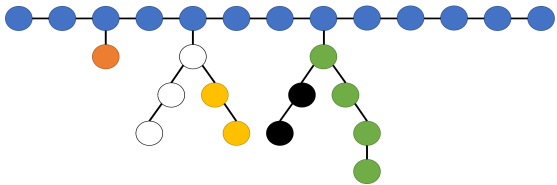
- 그러나 작업을 여러번 할 수 있다면 아래와 같은 생각을 해봐야 합니다.
- 작업 몇 번으로 어떠한 서브트리의 깊이를 깊게 만든 후, 나중에 그 서브트리를 지름의 끝에 붙이는 것이 이득이진 않을까요?
- 그러나 생각해보면 한 번에 모아서 지름을 늘리는 것과, 한 번씩 작업하며 지름을 늘리는 것 간의 지름 차이는 없습니다.
- 따라서 작업을 할 때마다 붙일 수 있는 깊이가 가장 깊은 서브트리를 지름에 붙이는 전략을 사용합시다.

G. 트리 다듬기

- 그럼 작업을 할 때마다 지름을 제외한 서브트리 중 깊이가 가장 깊은 것을 찾아야 할까요?
- 이는 구현이 복잡할뿐만아니라 $O(NK)$ 의 시간복잡도를 갖기 때문에 시간초과를 받게 됩니다.
- 우리는 $O(N)$ 만에 앞으로 작업할 서브트리들의 깊이를 구할 수 있습니다.

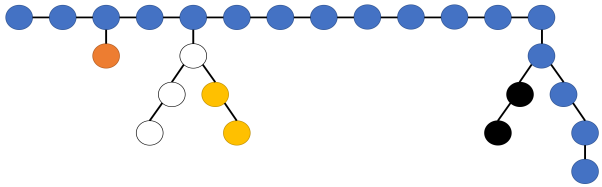
G. 트리 다듬기

- ✓ 아래 트리에서 지름은 파란색이고, 그 외의 노드들은 같은 색끼리 하나의 후보가 됩니다.
- ✓ 각 후보는 서브트리 중 깊이가 가장 깊은 하나의 경로와 같습니다.



- ✓ 이제 우리의 전략대로 깊이가 가장 깊은 초록색 후보를 지름의 끝에 붙여봅시다.

G. 트리 다듬기



- 작업 후 초록색 후보는 지름에 포함되었고, 다른 후보들(주황색, 흰색, 노란색, 검은색)은 모양이 변하지 않은 것을 확인할 수 있습니다.
- 즉, 우리는 이전 슬라이드의 그래프와 같이 지름을 구하고, 각 후보들을 미리 구해둔다면, 값이 큰 K 개를 처음 지름에 더하여 답을 구할 수 있습니다.

G. 트리 다듬기

- 이제 각 후보들을 구하는 방법을 알아봅시다.
- 먼저, DFS 한 번으로 지름의 끝점 중 하나를 찾을 수 있습니다. 이 방법은 잘 알려져 있습니다.
- 인터넷에 트리의 지름을 검색 하거나 boj.kr/1167 문제의 풀이를 공부하면 됩니다.
- 이제, 우리가 찾은 지름의 끝점을 시작으로 DFS를 수행합니다. 각 DFS의 return 값을 해당 노드부터 리프노드까지의 깊이 중 가장 큰 값이라 합시다.
- 그럼 자식 노드의 DFS 값들 중 가장 큰 값을 제외하고는 후보에 추가할 수 있습니다.

G. 트리 다듬기

- 이렇게 구한 후보값들 중 가장 큰 값은 제외해야 합니다. 이는 지름을 나타내기 때문입니다.
- 가장 큰 값을 제외한 후보값들 중 값이 큰 K 개를 지름에 더한 값이 답이 됩니다.
- 후보값이 K 개보다 작을 수도 있음에 주의해야 합니다. 이는 작업을 K 번 하기 전에 트리가 일자형 트리가 됨을 의미합니다.

G. 트리 다듬기

- ✓ 후보값을 std::priority_queue로 관리하거나, 정렬을 진행하면 문제를 $O(NlogK)$에 문제를 해결할 수 있습니다.
- ✓ 후보값의 범위가 10만을 넘지 않는 것을 이용하여 계수 정렬과 비슷한 방식으로 관리하면 문제를 $O(N)$에 해결할 수도 있습니다.

H. 완전한 수열 리버스

`constructive`

출제진 의도 – **Hard**

- 제출 9번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: **X**
- 출제자: 강효규 `djs100201`

H. 완전한 수열 리버스

- ✓ $[2, 0, 0, 2]$ 라는 수열을 단위수열 이라고 정의해봅시다.
- ✓ 단위 수열은 길이가 정확히 4 이고, 안정도가 4 인 수열입니다.
- ✓ M 의 최댓값이 N 이므로, 우리는 단위 수열을 이용할 것입니다.

H. 완전한 수열 리버스

- 단위 수열을 연속해서 이어 붙여 더 확장된 수열을 만들어 봅시다.
- 예를 들어 $[2, 0, 0, 2, 2, 0, 0, 2]$ 라는 수열을 살펴봅시다.
- 이때, 두 단위 수열을 포함한 완전한 연속 부분 수열은 존재하지 않습니다.
- 두 단위 수열을 포함하는 연속 부분 수열의 합은 4 이상인 짝수이기 때문입니다.

- ✓ 따라서 단위 수열을 적절히 이어 붙여 안정도가 M에 가깝게 만들수 있습니다.
- ✓ 기본적으로 $M/4$개의 단위 수열을 이어 붙인 수열을 생각해 봅시다.
- ✓ 이제 M을 4로 나눈 나머지에 따라 case가 조금은 달라집니다.

H. 완전한 수열 리버스

- M을 4로 나눈 나머지가 0일 경우에는 N에 맞게 2를 맨 앞에 붙여줍니다..
- M을 4로 나눈 나머지가 1일 경우에는 맨 마지막 단위 수열을 $[2, 0, 2, 0, 0]$으로 바꾼 뒤 N에 맞게 2를 맨 앞에 붙여줍니다.
- M을 4로 나눈 나머지가 2일 경우에는 맨 마지막 단위 수열을 $[2, 0, 0, 2, 0]$으로 바꾼 뒤 N에 맞게 2를 맨 앞에 붙여줍니다
- M을 4로 나눈 나머지가 3일 경우에는 맨 마지막 단위 수열을 $[2, 0, 0, 2, 0, 2]$으로 바꾼 뒤 N에 맞게 2를 맨 앞에 붙여줍니다.

H. 완전한 수열 리버스

- M 이 3 이하인 경우에는 예외가 생길 수 있으니 적절히 예외처리 해 줍시다.
- 이 풀이 외에도 검수과정에서 여러 풀이가 나왔습니다.
- 적절히 떠올린 후 문제를 해결해 주시면 됩니다.

Champion

문제		의도한 난이도	출제자
A	완전한 수열	**Easy**	강효규 djs100201
B	DPS	**Easy**	김동건 dong_gas
C	현대모비스 소프트웨어 아카데미	**Easy**	김동건 dong_gas
D	수학적인 최소 공통 조상	**Medium**	강효규 djs100201
E	고양이 목에 리본 달기	**Medium**	이윤제 yjyj1027
F	더 어려운 스케줄링	**Hard**	임지환 raararaara
G	Maximize MEX	**Hard**	강효규 djs100201
H	슈퍼 블랙잭	**Hard**	박수현 shiftpsh

A. 완전한 수열

`bruteforce`, `primality test`

출제진 의도 – **Easy**

- ✓ 제출 160번, 정답 36명 (정답률 22.500%)
- ✓ 처음 푼 사람: **박재형**, 4분
- ✓ 출제자: 강효규 `djs100201`

A. 완전한 수열

- ✓ N 과 a_i 의 제한에 주목해봅시다.
- ✓ 모든 연속 부분 수열은 총 $N \cdot (N-1)/2$ 개 있습니다.
- ✓ N 이 500 으로 충분히 작습니다.
- ✓ 따라서 모든 연속 부분 수열에 대해서 완전한 수열인지를 판단하면 문제를 해결할 수 있습니다.

A. 완전한 수열

- $\mathcal{O}\left(\sqrt{N}\right)$ 의 시간복잡도로 소수 판정을 해주면 문제를 해결할 수 있습니다.
- $\mathcal{O}\left(N\right)$ 의 시간복잡도로 소수 판정을 하면 시간 초과를 받게 됩니다.
- 쉬운 문제로 설계되었기 때문에 구간합을 구하는 과정을 $\mathcal{O}\left(N^3\right)$ 의 시간복잡도로 구현하여도 정답을 받을 수 있습니다.

B. DPS

`mathmatics`, `combinatorics`

출제진 의도 – **Easy**

- ✓ 제출 186번, 정답 37명 (정답률 19.892%)
- ✓ 처음 푼 사람: **손기령**, 8분
- ✓ 출제자: 김동건`dong_gas`

B. DPS

- ✓ 문제를 읽어보면 각 핸들의 첫 글자만 필요하다는 것을 쉽게 알 수 있습니다.
- ✓ 삼중 for문으로 문제를 풀면 시간초과를 받게 됩니다.
- ✓ 따라서 각 알파벳 별로 해당 알파벳으로 시작하는 핸들의 개수를 세어둡니다.
- ✓ 입력으로 주어지는 팀 명은 아래와 같은 세 가지 경우로 나눌 수 있습니다.
  - 세 알파벳이 모두 같은 경우
  - 세 알파벳 중 두 알파벳이 같고 나머지 하나의 알파벳은 다른 경우
  - 세 알파벳이 모두 다른 경우

B. DPS

- ✓ 세 알파벳이 모두 같은 경우
  - 해당 알파벳으로 시작하는 핸들의 개수를 a라 하면 $\binom{a}{3}$으로 한 번에 계산할 수 있습니다.
- ✓ 세 알파벳 중 두 알파벳이 같고 나머지 하나의 알파벳은 다른 경우
  - 같은 두 알파벳을 A, 나머지 하나의 알파벳을 B라 합시다.
  - A로 시작하는 핸들의 개수를 a, B로 시작하는 핸들의 개수를 b라 하면 $\binom{a}{2} \times b$로 한 번에 계산할 수 있습니다.

B. DPS

- 세 알파벳이 모두 다른 경우
  - 서로 다른 세 알파벳을 각각 A, B, C라 합시다.
  - A로 시작하는 핸들의 개수를 a, B로 시작하는 핸들의 개수를 b, C로 시작하는 핸들의 개수를 c라 하면 $a \times b \times c$로 한 번에 계산할 수 있습니다.

- 참고로 DPS는 2022 ICPC Seoul Regional에서 사용한 저희 팀의 이름입니다.

C. 현대모비스 소프트웨어 아카데미

`sorting`, `two_pointer`
출제진 의도 – **Easy**

- ✓ 제출 63번, 정답 32명 (정답률 50.794%)
- ✓ 처음 푼 사람: **손기령**, 12분
- ✓ 출제자: 김동건`dong_gas`

- ✓ 조건을 만족하는 팀을 최대한 많이 구성하는 것이 목표입니다.
- ✓ 아래와 같은 전략을 떠올리는 것은 크게 어렵지 않습니다.
  - 남아 있는 학회원 중 능력치가 가장 큰 A를 뽑는다.
  - A를 뽑은 후, 남아 있는 학회원 중 A와의 능력치 합이 M 이상인 학회원 중 능력치가 가장 낮은 학회원 B를 뽑는다.
  - A와 B를 한 팀으로 구성한다.
  - 위 작업을 남은 학회원이 1명 이하가 될 때까지 반복한다.
- ✓ 이 문제에서 위 전략은 성립할까요?
- ✓ 결론부터 말하면, 위 전략은 성립합니다.
- ✓ 위 전략으로 K개의 팀이 구성되었다고 합시다.

- ✓ 먼저, 선택된 $2K$ 명 중 한 명이 빠지고, 선택되지 않은 사람 중 한 명이 들어오면 어떻게 되는지 봅시다.
  - K 개의 팀 중 한 팀을 골라 팀원 중 능력치가 더 높은 사람을 X 라 해봅시다.
  - X 를 팀이 없는 학생인 Y 와 바꿔봅시다.
  - 전략에 의해 X 의 능력치는 Y 의 능력치보다 크거나 같을 수밖에 없습니다. Y 의 능력치가 더 컸다면, X 가 아닌 Y 가 선택되었을 것이기 때문입니다.
  - 즉, X 와 Y 를 바꾸는 것은 팀 능력치를 감소시키거나 유지시키기만 할 뿐이므로, 그렇게 하지 않는 것이 항상 이득입니다.
  - 비슷한 방식으로 한 팀을 골라 능력치가 더 낮은 사람을 팀이 없는 사람과 바꾸는 방법도 손해라는 것을 증명할 수 있습니다.
- ✓ 따라서, 우리가 고른 $2K$ 명으로 K 개의 팀을 구성하는 것이 항상 최적입니다.

- ✓ 이번엔, 동일한 $2K$ 명으로 팀을 구성하지만, 기존 답과 팀 구성이 달라졌다고 해봅시다.
  - K 개의 팀 중 팀 S 와 팀 T 를 골라 팀원 한 명을 교환해 봅시다.
  - 기존에는 팀 S 와 팀 T 모두 능력치가 M 이상이었습니다.
  - 그러나 팀원 한 명을 교환한다면, 한 팀의 능력치는 감소하거나 유지됩니다. 감소되는 경우 팀의 능력치가 M 보다 작아지는 경우가 발생할 수 있습니다.
  - 따라서 팀원을 교환한다면 총 팀의 수는 유지되거나 감소되기만 합니다.
- ✓ 따라서 우리가 구성한 팀이 항상 최적입니다.

- ✓ 위 전략은 정렬을 한 뒤, 투 포인터 기법을 이용하여 구현할 수 있습니다.
  - 정렬에 $\mathcal{O}(N \log N)$, 투 포인터 기법에 $\mathcal{O}(N)$의 시간이 필요하므로 총 $\mathcal{O}(N \log N)$에 문제를 해결할 수 있습니다.
- ✓ std::multiset과 이분탐색을 이용하여 구현하는 방법도 있습니다.

D. 수학적인 최소 공통 조상

`number theory`

출제진 의도 – **Medium**

- ✓ 제출 118번, 정답 18명 (정답률 15.254%)
- ✓ 처음 푼 사람: **박재형**, 31분
- ✓ 출제자: 강효규 `djs100201`

D. 수학적인 최소 공통 조상

- ✓ 문제를 우선 정확히 이해해 봅시다.
- ✓ 1 번 정점이 루트인 특수한 성질의 트리에서 두 정점의 최소 공통 조상을 구해야 합니다.
- ✓ 일반적인 트리에서 최소 공통 조상을 $\mathcal{O}(\log N)$ 의 시간복잡도로 빠르게 구할 수 있습니다.
- ✓ 하지만 이 문제에서 10^{12} 개의 정점과 간선을 모두 저장하기에는 메모리가 부족합니다.
- ✓ 따라서 주어진 특수한 성질을 잘 이용해봅시다.

D. 수학적인 최소 공통 조상

- ✓ 모든 소인수는 2보다 크거나 같습니다.
- ✓ 따라서 x번 정점의 자식들의 번호는 모두 $2x$보다 크거나 같습니다.
- ✓ 이를 잘 생각해보면 1번 정점에서 x번 정점까지의 거리는 최대 $\mathcal{O}(\log x)$ 정도임을 알 수 있습니다.
- ✓ 따라서 a와 b에 대해서, 1에 도달할 때까지 소인수로 나눠주면서 그 정수들을 기록합니다.
- ✓ $\log 10^{12} \leq 50$이므로 각각 50개 정도의 정수만이 기록됩니다.

D. 수학적인 최소 공통 조상

- ✓ 따라서 가장 먼저 겹치는 정수를 찾게 되면 문제를 해결할 수 있습니다.
- ✓ 그런데 어떻게 빠르게 임의의 정수 a 를 소인수로 나눠갈 수 있을까요?
- ✓ 이는 $\sqrt{a}$ 이상의 소인수는 많아야 하나 존재함을 이용하면 됩니다.
- ✓ 따라서 $\sqrt{a}$ 까지 반복문을 돌면서, 나누어 떨어진다면 나눠줍니다.
- ✓ 마지막에 1 이 아니라면 소인수로 판정해줍니다.

D. 수학적인 최소 공통 조상

- ✓ 이러한 방법을 이용하여 두 정수 a와 b를 모두 소인수분해 할 수 있습니다.
- ✓ 따라서 $\mathcal{O}\left(\sqrt{N}\right)$의 시간복잡도로 문제를 해결할 수 있습니다.

E. 고양이 목에 리본 달기

`Dynamic Programming`
출제진 의도 – **Medium**

- ✓ 제출 53번, 정답 14명 (정답률 26.415%)
- ✓ 처음 푼 사람: **박재형**, 36분
- ✓ 출제자: 이윤제`yjyj1027`

E. 고양이 목에 리본 달기

고양이들의 만족도를 극대화하려면 어떻게 해야 할까요?

- ✓ 각 고양이에 대해 만족도가 가장 큰 리본을 달아주면 됩니다.
- ✓ 이웃한 고양이들이 가장 선호하는 리본이 같다면 누구에게 달아주어야 할까요?
- ✓ 상황에 따라 다릅니다.
- ✓ 모든 상황을 고려해 최적의 답을 내야 할 것입니다.

E. 고양이 목에 리본 달기

$Satis_{i,j} = i$ 번째 고양이에게 j 번째 리본을 달아 주었을 때의 최대 만족도 합

- ✓ $Satis_{i,j}$ = $\max(Satis_(i-1, j_{prev})) + s_{i,j}$ where $0 \leq j_{prev} < n$, $j_{prev} \neq j$
- ✓ $\mathcal{O}\left(NK^2\right)$
- ✓ 더 효율적으로 알 수 있는 방법이 없을까요?

E. 고양이 목에 리본 달기

$Satis_{i,j} = i$ 번째 고양이에게 j 번째 리본을 달아 주었을 때의 최대 만족도 합

- ✓ $Satis_{i,j}$ = $\max(Satis_(i-1, j_{prev})) + s_{i,j}$ where $0 \leq j_{prev} < n$, $j_{prev} \neq j$
- ✓ $Satis_{i-1,j}$ 중 가장 큰 2개를 알고 있다면?
- ✓ 겹치는 리본일 때: 두 번째로 큰 값 선택
- ✓ 겹치지 않을 때: 가장 큰 값 선택
- ✓ $\mathcal{O}(NK)$

F. 더 어려운 스케줄링

`deque`, `data structure`

출제진 의도 – **Hard**

- ✓ 제출 55번, 정답 4명 (정답률 7.273%)
- ✓ 처음 푼 사람: **박재형**, 58분
- ✓ 출제자: 임지환 `raararaara`

F. 더 어려운 스케줄링

Master Division은 추가, 정렬, 뒤집기 연산만 주어지고 최종 결과에 대하여 앞에서 k번째에 해당하는 업무의 고유번호를 구하는 문제입니다.
Champion Division에서는 반복적으로 가장 앞에 있는 업무의 고유번호를 출력하는 연산이 추가되었습니다.

정렬 연산에 대해 좀더 동적으로 반응해야 합니다.
하지만 '어려운 스케줄링' 문제와 마찬가지로 매번 정렬을 하고 뒤집기를 수행하면 약 $\mathcal{O}\left(QN \log N\right)$ 의 시간복잡도로 시간초과를 받게 됩니다.

F. 더 어려운 스케줄링

마찬가지로 정렬 연산이 없는 경우에 대해서 문제를 해결해봅시다.

덱(deque) 2개를 활용하여 뒤집기 연산이 짝수 번 발생했는지, 홀수 번 발생했는지에 따라 각각 따로 관리해줍니다.
이후 발생하는 업무 처리 연산(3번 연산)에 대해서는 덱 각각의 앞부분에서 가져오면 됩니다.

정렬까지 추가하여 고려해봅시다.

특정 시점에 스케줄러의 상태는 크게 다음과 같습니다.

- ✓ 모든 업무가 정렬된 상태
- ✓ 정렬 연산이 적용된 이후 업무 추가 연산, 뒤집기 연산이 발생한 상태

남아있는 모든 업무가 정렬된 상태인 경우

업무 처리 연산이 주어졌을 때, 그 전까지의 뒤집기 연산의 발생 횟수가

- ✓ 짝수 번(0번 포함)이라면 앞쪽을 관리하는 리스트에서
- ✓ 홀수 번이라면 뒤쪽을 관리하는 리스트에서

업무를 가져와 처리하면 됩니다.

F. 더 어려운 스케줄링

정렬 연산이 적용된 이후 업무 추가 연산, 뒤집기 연산이 발생한 상태라면

- ✓ 앞쪽과 뒤쪽을 관리하는 리스트에 원소가 남아있다면 해당 업무를
- ✓ 아닌 경우 뒤집기 연산의 발생 횟수에 따라 최댓값, 최솟값을

처리하면 됩니다.

F. 더 어려운 스케줄링

정렬된 리스트를 동적으로 관리하면서, 그 내부에서 최댓값 혹은 최솟값을 가져올 수 있는 자료구조는 std::set, 우선순위 큐 등을 이용합니다.

정렬 연산이 주어졌을 때, 앞쪽과 뒤쪽을 관리하는 리스트에 있는 모든 업무들을 위에서 언급한 자료구조로 옮겨줍니다.
k개의 업무를 std::set, 우선순위 큐 등으로 옮기는데에는 $O(klogN)$ 만큼의 시간복잡도를 가지고, 업무를 처리할 경우는 덱에서 꺼낼 때 $O(1)$, std::set 혹은 우선순위 큐에서 꺼낼 때 $O(logN)$ 만큼의 시간복잡도를 가집니다.

이때 뒤집기 연산은 사실상 초기화됨을 유의합니다.

G. Maximize MEX

`greedy`, `parametric search`

출제진 의도 – **Hard**

- ✓ 제출 27번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: **X**
- ✓ 출제자: 강효규 `djs100201`

G. Maximize MEX

- 주어진 문제의 정답은 단조성을 가집니다.
- 즉 적절히 연산을 시행하여 x를 남길 수 있었다면, $x-1$을 남길 수 있습니다.
- 어떻게 증명할 수 있을까요?

G. Maximize MEX

- ✓ N 은 2 보다 크거나 같으므로, 하나의 원소를 남기려면 최소 1 번의 연산은 해야 합니다.
- ✓ 마지막 연산이후 x 라는 하나의 원소가 남게 되었다고 생각해 봅시다.
- ✓ x 라는 원소가 남기 위해서는 이전에, 0 이상 $x-1$ 이하의 원소는 모두 가지고 있는 상태입니다.
- ✓ 이때 $x-1$ 들만 모아서 연산을 적용하면 0 이상 $x-2$ 이하의 원소와 x 보다 큰 원소들만 가지고 있는 상태가 됩니다.
- ✓ 그 이후 전체 원소에 연산을 적용하면 $x-1$ 을 만들 수 있습니다.

G. Maximize MEX

- ✓ 단조성을 보였으므로 이분 탐색을 통해 전체 문제를 해결해 봅시다.
- ✓ x라는 한 원소만을 남길 수 있는가? 라는 결정 문제로 바꾸어 봅시다.
- ✓ 이 결정문제를 풀기 전에 다음과 같은 사실을 확인하고 넘어갑시다.

G. Maximize MEX

- ✓ 0 이라는 단일 원소만이 있는 집합에 연산을 적용해봅시다.
- ✓ 1 이라는 단일 원소 집합이 됩니다.
- ✓ 1 이라는 단일 원소만이 있는 집합에 연산을 적용해봅시다.
- ✓ 0 이라는 단일 원소 집합이 됩니다.
- ✓ 따라서 이 문제에서 0 과 1 은 같은 원소입니다.

- 이제 결정 문제를 해결해 봅시다.
- x 이상의 원소들은 전부 각각 0 으로 바꿔줍니다.
- 다음 원소가 필요한 개수를 나타내는 $need$ 라는 변수를 설정해 줍시다.
- 처음에는 0 부터 $x-1$ 까지의 원소가 1 개 필요하므로, $need$ 변수를 1 로 초기화 합니다.
- $x-1$ 부터 2 까지 원소가 $need$ 개 이상있는지 순회해 봅시다.

G. Maximize MEX

- ✓ 만약 현재 살펴보고 있는 원소가 $need$개 이상이라면, 남은 원소를 0으로 바꾸고 넘어갑니다.
- ✓ 만약 현재 살펴보고 있는 원소가 $need$개 이하라면, 어떨까요?
- ✓ 현재 원소의 개수가 x개라면, $need$에 $need - x$ 만큼을 더해 줍니다.
- ✓ i라는 원소를 만드는 방법은, 0부터 $i - 1$에 연산을 시행하는 방법밖에 없기 때문입니다.
- ✓ 모든 순회 이후, $2 * need$와 0의 개수와 1의 개수의 합을 비교하면 결정 문제를 풀 수 있습니다.

G. Maximize MEX

- ✓ 순회과정에서, 오버플로우가 날 수 있기 때문에 적절히 조심해서 처리해 줍시다.
- ✓ 결정문제를 $\mathcal{O}(N)$에 풀 수 있어 전체 문제가 $\mathcal{O}(N \log N)$의 시간복잡도에 해결됩니다.
- ✓ 위 풀이 외에도 적절히 구현하면 $\mathcal{O}(N)$에 문제를 풀 수 있습니다.

H. 슈퍼 블랙잭

`dp`, `parametric_search`, `prefix_sum`
출제진 의도 – **Challenging**

- 제출 3번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: **X**
- 출제자: 박수현`shiftpsh`

$0 \leq x$에 대해 다음과 같은 점화식을 생각해 봅시다.

$$f(x) = (\text{현재 } x\text{점일 때, } [S, E] \text{ 점을 달성하기 위해 뽑아야 하는 카드 수의 기댓값})$$

- ✓ $E < x$라면, 버스트되었으므로 카드를 전부 버리고 처음부터 시작해야 합니다. 따라서 이 경우 $f(x) = f(0)$입니다.
- ✓ $S \leq x \leq E$라면, 승리했으므로 카드를 더 뽑을 필요가 없습니다. 따라서 이 경우 $f(x) = 0$입니다.
- ✓ $0 \leq x < S$라면?

덱이 하나만 있고 여기 적혀 있는 수 A_1 이 있다고 생각해 봅시다. 여기서는 1 에서 A_1 사이의 수가 같은 확률로 나올 수 있습니다.

따라서 x 점인 상황에서 A_1 이 적힌 덱의 카드를 고를 경우, 기댓값은

$$1 + \sum_{i=1}^{A_1} \left[\frac{1}{A_1} \times f\left(x+i\right) \right]$$

가 됩니다.

H. 슈퍼 블랙잭

덱이 여러 개 있는 상황을 생각해 봅시다. 이전 슬라이드와 같은 방법으로 생각하면, i 번째 덱을 고르면

$$1+\sum_{j=1}^{A_i}\left[\frac{1}{A_i}\times f\left(x+j\right)\right]$$

의 기댓값을 가질 것입니다.

최선의 전략은 i 에 대해 위 식의 값이 최소가 되는 덱을 고르는 것입니다. 따라서

$$f\left(x\right)=\min_{1\le i\le D}\left[1+\sum_{j=1}^{A_i}\left[\frac{1}{A_i}\times f\left(x+j\right)\right]\right]$$

입니다.

H. 슈퍼 블랙잭

위 식은 x 마다 $\mathcal{O}\left(D \max_i D_i\right)$ 가 소요되므로 약간 느립니다. 식을 정리해서 최적화할 여지가 있는지 봅시다.

$$
\begin{aligned}
f(x) &= \min_{1 \le i \le D} \left[1 + \sum_{j=1}^{A_i} \left[\frac{1}{A_i} \times f(x+j) \right] \right] \\
&= 1 + \min_{1 \le i \le D} \left[\frac{1}{A_i} \sum_{j=1}^{A_i} f(x+j) \right]
\end{aligned}
$$

x의 값을 큰 값부터 작은 값으로 내려오면서 계산해 준다면 $\sum_{j=1}^{A_i} f(x+j)$ 부분은 suffix sum 과 같은 방법으로 최적화 가능해 보입니다. 따라서 정리하면

$$f(x) = \begin{cases} f(0) & \text{if } E < x \\ 0 & \text{if } S \le x \le E \\ 1 + \min\limits_{1 \le i \le D} \left[\dfrac{1}{A_i} \sum\limits_{j=1}^{A_i} f(x+j) \right] & \text{if } 0 \le x < S \end{cases}$$

가 되고, $\mathcal{O}(D)$로 줄어듭니다. 우리가 원하는 답은 $f(0)$입니다.

하지만 이 모든 계산은 $f\left(0\right)$이 미리 주어져야 가능합니다. 따라서 $E < x$인 경우의 $f\left(x\right)$의 값을 미리 f_0이라고 가정하고, 그에 따른 $f\left(0\right)$의 값을 비교하는 방법을 생각해 볼 수 있습니다.

이렇게 계산된 $f\left(x\right)$는 f_0의 값이 증가함에 따라 함께 증가하며, 단조성을 띕니다. 이는 귀납적으로 증명 가능합니다.

따라서 f_0보다 $f\left(0\right)$의 값이 크면 f_0을 줄이고, 아니라면 f_0을 키우는 방법으로 f_0과 $f\left(0\right)$이 수렴하게 하면 됩니다. 이는 이분 탐색 등의 방법으로 계산 가능합니다.

DP 테이블을 채우는 데 $\mathcal{O}\left(SD\right)$가 걸리므로 총 $\mathcal{O}\left(SD \times \log\left(\text{범위} \times \varepsilon^{-1}\right)\right)$의 시간이 걸립니다.

이분 탐색의 범위는 이론적으로 최대인 $f\,(x)$의 값으로 잡습니다.

(이하 2022-11-28 수정)

- ✓ D가 작을수록 f는 커집니다. D가 관여하는 부분은 $\min$뿐입니다.
- ✓ $E - S$가 작을수록 f는 커집니다.

이에 착안해 $D = 1, S = E$로 두고 적당한 상한을 정해 봅시다. 최선의 전략은 아니지만, 0 점에서 시작해 $S - A_1$ 점을 달성한 뒤 이 시점에서 S점 달성에 성공할 경우 종료, 실패할 경우 같은 전략으로 재시도하는 전략을 생각해 봅시다.

A_1을 뒤집었을 때 얻을 수 있는 수의 기댓값은 $\frac{A_1 + 1}{2}$입니다. 따라서 $S - A_1$ 점 달성까지는 평균 $(S - A_1) / \left(\frac{A_1 + 1}{2}\right)$ 번의 뒤집음이 필요합니다.

여기서 카드를 하나 뒤집어 정확히 S 점을 달성할 확률은 $\frac{1}{A_1}$ 입니다. 따라서, 이 전략의 기댓값의 상한은 $S - A_1$ 점 달성한 시점에서 카드를 한 장 더 뒤집는 행동을 A_1 번 할 때입니다.

따라서 상한은 아래와 같이 설정하면 충분합니다.

$$\begin{aligned}\left[(S-A_1)\,/\left(\frac{A_1+1}{2}\right)+1\right]\times A_1 &= 2\,(S-A_1)\times\frac{A_1}{A_1+1}+1\times A_1\\ &< 2\,(S-0)\times\frac{A_1+1}{A_1+1}+A_1\\ &= 2S+A_1\le 2S+E\le 3E\end{aligned}$$

Open Contest

문제		의도한 난이도	출제자
1	효구와 호규 (Hard)	**Medium**	조원빈 wbcho0504
2	AND vs OR	**Challenging**	강효규 djs100201

1. 효구와 호규 (Hard)

`ad_hoc`, `graph traversal`
출제진 의도 – **Medium**

- ✓ 제출 00번, 정답 00명 (정답률 00.000%)
- ✓ 처음 푼 사람: **000**, 00분
- ✓ 출제자: 조원빈 `wbcho0504`

1. 효구와 호규(Hard)

- 효규와 호규(Easy) 문제에서 매 번 없애야 하는 두 카드를 출력해야 합니다.
- 풀이는 Easy버전과 같습니다.
- 구현이 다소 까다로울 수 있어 Open Contest로 빠지게 되었습니다.

2. AND vs OR

`segment tree`, `offline query`
출제진 의도 – **Challenging**

- ✓ 제출 00번, 정답 00명 (정답률 00.000%)
- ✓ 처음 푼 사람: **000**, 00분
- ✓ 출제자: 강효규`djs100201`

2. AND vs OR

- ✓ 연속 부분 수열 중에 가치가 양수인 구간을 특수한 구간이라고 불러 봅시다.
- ✓ 주어진 수열에서 특수한 구간은 매우 적을 것이라는 관찰이 필요합니다.
- ✓ 우선 특수한 구간이 $\mathcal{O}\left(N \log 10^{18}\right)$ 개 정도라는 관찰을 해 봅시다.

2. AND vs OR

- ✓ 특수한 구간의 왼쪽 좌표를 l 로 고정해 봅시다.
- ✓ 구간 $[l, r]$ 이 특수한 구간이라면 a_r 은 $(a_{l+1} \mid a_{l+2} \mid \cdots \mid a_{r-1})$ 보다 큽니다.
- ✓ a_R 은 $(a_{l+1} \mid a_{l+2} \mid \cdots \mid a_{r-1})$ 에 켜지지 않은 비트가 켜져 있습니다.
- ✓ 따라서 r 을 $r+1$ 로 확장시키면, OR 연산값의 비트는 1 개 이상 더 켜집니다.
- ✓ 고정된 l 에 대해서 최대 $\mathcal{O}\left(\log 10^{18}\right)$ 개 정도의 r 값을 가질 수 있습니다.
- ✓ 따라서 특수한 구간은 $\mathcal{O}\left(N \log 10^{18}\right)$ 개 정도입니다.

2. AND vs OR

- ✓ 하지만 $\mathcal{O}\left(N \log 10^{18}\right)$ 개는 꽤 큰 양이므로 시간 내에 통과하기 어려울 수 있습니다.
- ✓ 특수한 구간이 $\mathcal{O}\left(N\right)$ 개 정도라는 관찰을 해 봅시다.

2. AND vs OR

- 다음과 같은 두 관찰을 해 봅시다.
- $(a_l \ \& \ a_r) \leq min(a_l, a_r)$
- $max(a_{l+1}, a_{l+2}, \cdots, a_{r-1}) \leq (a_{l+1} \,|\, a_{l+2} \,|\, \cdots \,|\, a_{r-1})$
- 따라서 모든 특수한 구간은 양 끝의 최솟값이 사이 구간의 최댓값보다 큽니다.

- 다음과 같은 두 관찰을 해 봅시다.
- $(a_l \ \& \ a_r) \leq min(a_l, a_r)$
- $max(a_{l+1}, a_{l+2}, \cdots, a_{r-1}) \leq (a_{l+1} \,|\, a_{l+2} \,|\, \cdots \,|\, a_{r-1})$
- 따라서 모든 특수한 구간은 양 끝의 최솟값이 사이 구간의 최댓값보다 큽니다.
- 양 끝의 최솟값이 사이 구간의 최댓값보다 큰 구간을 매우 특수한 구간이라고 해 봅시다.

2. AND vs OR

- 매우 특수한 구간은 $\mathcal{O}(N)$ 개 정도입니다.
- ioi 2019 Rectangles, 제1회 블롭컵의 blobpopcorn 에서도 출제된 논리입니다.
- 모든 i 에 대해서 a_i 가 구간 $[l, r]$ 의 사이의 원소들의 최댓값이라고 해 봅시다.
- 그렇다면 l 과 r 이 고정됩니다.
- $j < i$ 이면서, $a_j > a_i$ 인 최대 j 가 l, $j < r$ 이면서, $a_r > a_j$ 인 최소 j 가 r 이 됩니다.
- 그 이외에 경우에는 모두 모순이 발생합니다.

2. AND vs OR

- 고정된 i에 대해 매우 특수한 구간 $[l, r]$은 고정됩니다.
- 따라서 매우 특수한 구간은 $\mathcal{O}(N)$개 정도입니다.
- 우리는 모든 매우 특수한 구간들에 대해서 실제로 특수한 구간인지를 확인해줘야 합니다.
- 이는 세그먼트 트리로 매우 빠르게 구할 수 있습니다.
- 따라서 우리는 특수한 구간 $O(N)$개와 그 가치를 $\mathcal{O}(NlogN)$에 모두 구할 수 있습니다.

2. AND vs OR

- ✓ 이제 쿼리마다 정답을 처리해야 합니다.
- ✓ 오프라인 쿼리와 세그먼트 트리로 적절하게 처리한다면 $\mathcal{O}\left((Q+N)logN\right)$ 정도에 전체 문제를 처리할 수 있습니다.