제2회 디미고 프로그래밍 챌린지 해설

Official Solution

by

대회 운영진 일동

2025년 3월 13일

문제		의도한 난이도	출제자
A	구멍	**Easy**	goodpjw2008
B	자세히 보아야 예쁘다	**Easy**	ian0704
C	대칭제곱수	**Easy**	deom
D	끊임없는 정렬과 창조함으로	**Normal**	goodpjw2008
E	시간표	**Normal**	ldhhello
F	자습실과 쿼리	**Normal**	ldhhello
G	8번 출구	**Hard**	ecode
H	누가 이름 안 적고 나갔어	**Hard**	seungchan0325

문제		의도한 난이도	출제자
I	디미 그래프	Hard	ldhhello
J	트리와 뽀미	Hard	ldhhello
K	IZ*ONE Sequence	Hard	ian0704
L	건물 폭파	Challenging	ecode
M	수열과 띄엄띄엄 쿼리	Challenging	goodpjw2008
N	디미교도소	Challenging	deom
O	비트 뒤집기와 쿼리	Challenging	seungchan0325

A. 구멍

`case_work implementation`
출제진 의도 – **Easy**

- 본 대회 – 제출 86번, 정답 37명 (정답률 44.186%)
- Open Contest – 제출 278번, 정답 182명 (정답률 66.187%)
- 처음 푼 사람(본 대회): **2hoyean**, 7분
- 처음 푼 사람(Open Contest): **ychangseok**, 1분
- 출제자: `goodpjw2008`

A. 구멍

- 공백을 포함한 문장의 경우 C언어의 fgets, C++의 getline, Python의 input 등 여러 방법으로 입력받을 수 있습니다.
- 각 문자의 구멍 개수는 A, a, b, D, d, e, g, O, o, P, p, Q, q, R, @는 1개, B는 2개로 계산하여 총합을 구하면 됩니다.

B. 자세히 보아야 예쁘다

`sorting` `greedy`

출제진 의도 – **Easy**

- 본 대회 – 제출 99번, 정답 35명 (정답률 35.354%)
- Open Contest – 제출 328번, 정답 162명 (정답률 51.524%)
- 처음 푼 사람(본 대회): **jeonsonghyun**, 7분
- 처음 푼 사람(Open Contest): **ychangseok**, 2분
- 출제자: `ian0704`

B. 자세히 보아야 예쁘다

- 재원이가 i번 친구를 A_i 시간 이상 보게 된다면, 퇴학 처분을 받게 됩니다.
- i번 친구를 볼 수 있는 최대 시간은 $A_i - 1$ 시간입니다.
- 만약 재원이가 $\sum_{i=1}^{N}(A_i - 1) + 1$의 시간만큼 친구들을 보게 된다면, 비둘기집 원리에 의해 어떤 한 친구 k에 대해 A_k 이상 본 경우가 반드시 존재하게 됩니다.
- 따라서, M이 $\sum_{i=1}^{N}(A_i - 1)$를 초과하면 재원이는 퇴학당하게 됩니다.

C. 대칭제곱수

`math`

출제진 의도 – **Easy**

- 본 대회 – 제출 211번, 정답 23명 (정답률 10.900%)
- Open Contest – 제출 288번, 정답 146명 (정답률 52.431%)
- 처음 푼 사람(본 대회): **coding_dana**, 14분
- 처음 푼 사람(Open Contest): **nflight11**, 1분
- 출제자: deom

C. 대칭제곱수

- 정수 N 이 대칭제곱수인지 판별하는 문제입니다.
- $\lfloor \sqrt{N} \rfloor^2 = N$ 인지 판별하고, 뒤집은 수에 대해서도 같은 판별을 통해 문제를 해결할 수 있습니다.



D. 끊임없는 정렬과 창조함으로

`sort implementation`
출제진 의도 – **Normal**

- 본 대회 – 제출 31번, 정답 17명 (정답률 58.065%)
- Open Contest – 제출 114번, 정답 87명 (정답률 77.193%)
- 처음 푼 사람(본 대회): **magic_spirit**, 21분
- 처음 푼 사람(Open Contest): **ychangseok**, 5분
- 출제자: `goodpjw2008`

D. 끊임없는 정렬과 창조함으로

- 삽입 쿼리의 경우 삽입할 위치 이후의 값들을 한 칸 뒤로 밀고, 이 연산으로 비게 된 위치에 값을 삽입함으로서 $O(N)$에 수행할 수 있습니다.
- 정렬 쿼리의 경우 배열의 최대 크기가 3000이므로 $O(NlogN)$ 이하의 정렬 기법으로 배열을 정렬하면 문제를 해결할 수 있습니다.

E. 시간표

`implementation`
출제진 의도 – **Normal**

- 본 대회 – 제출 71번, 정답 16명 (정답률 22.535%)
- Open Contest – 제출 193번, 정답 71명 (정답률 36.788%)
- 처음 푼 사람(본 대회): **coding_dana**, 34분
- 처음 푼 사람(Open Contest): **fermion5**, 3분
- 출제자: `ldhhello`

E. 시간표

- s를 다음과 같이 정의합시다.

$$s_i = \begin{cases} 1 & i\text{번 과목을 좋아함} \\ -1 & i\text{번 과목을 싫어함} \\ 0 & \text{otherwise} \end{cases}$$

- 이후 반복문을 돌며 현재까지 s와 동일한 값이 몇 번 연속으로 나왔는지 기록하는 변수를 관리하여 문제를 해결할 수 있습니다.

F. 자습실과 쿼리

`prefix_sum`
출제진 의도 – **Normal**

- 본 대회 – 제출 81번, 정답 8명 (정답률 09.877%)
- Open Contest – 제출 152번, 정답 35명 (정답률 23.684%)
- 처음 푼 사람(본 대회): **shinhuichan**, 310분
- 처음 푼 사람(Open Contest): **ychangseok**, 21분
- 출제자: `ldhhello`

F. 자습실과 쿼리



- 각 학생들은 왼쪽 또는 오른쪽 한 방향으로만 움직이는 것이 항상 최선입니다.
- 따라서 i번 학생의 위치가 P_i라고 할 때 $[1, P_i]$ 구간 또는 $[P_i, N]$ 구간의 모든 벽이 파괴됩니다.
- 여기서 벽이 파괴되는 구간이 왼쪽 또는 오른쪽 끝에 항상 붙어 있기 때문에 잠재적으로 벽이 있을 수 있는 구역은 하나의 연속한 구간이 되는 것을 알 수 있습니다.

F. 자습실과 쿼리

- i번 구역의 벽에 남은 내구도를 A_i로 정의합시다. 만약 i번 구역에 벽이 없다면 $A_i = 0$이라고 합시다.
- 또한 아직 벽이 파괴되지 않은 구간을 $[s, e]$라고 합시다. 이때 s, e의 초기 값은 각각 $1, N$입니다.
- 이때 $P_i \notin [s, e]$인 학생의 망치질 횟수는 0이고, 그렇지 않다면 망치질 횟수는 조건에 따라 $\sum_{k=s}^{P_i} A_k$, $\sum_{k=P_i}^{e} A_k$ 중 하나가 됩니다.
- 만약 i번 학생이 왼쪽으로 탈출했다면 s 값을 P_i로 업데이트하고, 오른쪽으로 탈출했다면 e 값을 P_i로 업데이트하여 문제를 해결할 수 있습니다.

G. 8번 출구

greedy
출제진 의도 – **Hard**

- 본 대회 – 제출 30번, 정답 4명 (정답률 13.333%)
- Open Contest – 제출 65번, 정답 29명 (정답률 46.154%)
- 처음 푼 사람(본 대회): **k_san**, 616분
- 처음 푼 사람(Open Contest): **terrasphere**, 14분
- 출제자: ecode

G. 8번 출구

- 1번 출구에 서 있을 때의 면역력을 m이라고 정의합시다.
- 1번 출구에서 i번 출구까지 이동하기 위해서는 $A_1 + A_2 + \cdots + A_{i-1}$의 면역력을 소비해야 합니다.
- 이는 1번 출구에 서 있을 때 $A_1 + A_2 + \cdots + A_{i-1} \leq m$이 성립하는 모든 i에 대하여 i번 출구까지는 1번 출구로 돌아오지 않고 이동할 수 있음을 의미합니다.

G. 8번 출구

- 1 번 출구에서 i 번 출구까지 이동한 후 다시 돌아온다고 할 때, m 은 $A_i - (A_1 + A_2 + \cdots + A_{i-1})$ 만큼 증가합니다.
- 즉, $A_1 + A_2 + \cdots + A_{i-1} \leq m$ 이 성립하는 모든 i 에 대하여 $A_i - (A_1 + A_2 + \cdots + A_{i-1})$ 의 값이 가장 큰 출구에서 돌아오는 것이 최적입니다.

G. 8번 출구

- $A_1 + A_2 + \cdots + A_{i-1} \leq m$ 이 성립하는 i 의 개수를 늘려가면서 m 값을 가장 많이 증가시킬 수 있는 i 를 저장합니다.
- $A_1 + A_2 + \cdots + A_N \leq m$ 이 성립하여 N 번 출구를 통과하기까지 최적의 출구를 통해 1 번 출구로 몇번 돌아가야 하는지 카운트하여 답을 도출할 수 있습니다.
- 이동 가능한 i 의 개수에 따라 N 번 계산하므로 $\mathcal{O}(N)$ 시간에 문제를 해결할 수 있습니다.

H. 누가 이름 안 적고 나갔어

`graphs` `graph_traversal` `bfs`

출제진 의도 – **Hard**

- 본 대회 – 제출 39번, 정답 5명 (정답률 20.513%)
- Open Contest – 제출 110번, 정답 36명 (정답률 32.727%)
- 처음 푼 사람(본 대회): **shinhuichan**, 345분
- 처음 푼 사람(Open Contest): **mujigae**, 47분
- 출제자: seungchan0325

H. 누가 이름 안 적고 나갔어

- ✓ 진우는 다음 두 가지 방법으로 승찬이를 찾을 수 있습니다.
  1. 진우가 직접 승찬이를 찾는다.
  2. 진우가 선생님께 도움을 요청하면, 선생님이 승찬이를 대신 찾아준다.
- ✓ 위 두 가지 경우로 나누어 문제를 해결할 수 있습니다.

H. 누가 이름 안 적고 나갔어

- 진우의 위치를 j, 승찬이의 위치를 s, 어느 한 선생님의 위치를 t라고 합시다.
- 위치 u와 v의 거리를 $d(u, v)$라고 합시다.
- 1번 경우에 걸리는 시간은 $2 \cdot d(j, s)$입니다.
- 2번 경우에 걸리는 시간은 $2 \cdot d(j, t) + d(s, t)$입니다.
- 따라서 답은 $\min(2 \cdot d(j, s), \min_{t}(2 \cdot d(j, t) + d(s, t)))$입니다.

H. 누가 이름 안 적고 나갔어



- ✓ s, j 각각을 시작 위치로 BFS를 하여 모든 t에 대해 $d(j,t), d(s,t)$을 구할 수 있습니다.
- ✓ BFS의 시간복잡도는 $\mathcal{O}(V+E)$로 알려져 있음으로 $\mathcal{O}(N \cdot M)$ 시간에 문제를 해결할 수 있습니다.

I. 디미 그래프

`graphs` `graph_traversal` `implementation` `disjoint_set`

출제진 의도 – **Hard**

- 본 대회 – 제출 30번, 정답 6명 (정답률 20.000%)
- Open Contest – 제출 55번, 정답 29명 (정답률 52.727%)
- 처음 푼 사람(본 대회): **shinhuichan**, 615분
- 처음 푼 사람(Open Contest): **fermion5**, 24분
- 출제자: `ldhhello`

- 다음은 가능한 방법 중 하나입니다.
- 우선 $N \neq M$ 이라면 답은 **NO** 입니다.
- DFS를 통해 사이클에 포함되는 정점을 모두 찾습니다.
- 이들 정점 중 차수가 3 초과인 정점이 있거나, 차수가 3 인 정점이 하나가 아니라면 답은 **NO** 입니다.
- 이러한 차수가 3 인 정점을 c 라고 합시다. c 에서 출발해 사이클에 포함되지 않는 정점 방향으로 그래프를 순회합니다.
- 그러한 정점 중 차수가 2 초과인 정점이 있다면 답은 **NO** 입니다.
- 이제 지금까지 방문한 정점의 총 개수가 N 이라면 답은 **YES**, 그렇지 않다면 **NO** 입니다.

- 또는 정점의 차수만 세어도 문제를 풀 수 있습니다.
- 다음 조건을 모두 만족하는 그래프는 디미 그래프입니다.
  1. 정점이 모두 연결되어 있다.
  2. 차수가 3인 정점이 오직 하나이다.
  3. 차수가 1인 정점이 오직 하나이다.
  4. 그 외의 모든 정점의 차수는 2이다.
- 분리 집합 등의 방법으로 위 조건을 모두 판정하면 답을 구할 수 있습니다.

J. 트리와 뽀미

`dp` `trees`

출제진 의도 – **Hard**

- 본 대회 – 제출 12번, 정답 6명 (정답률 50.000%)
- Open Contest – 제출 38번, 정답 24명 (정답률 65.789%)
- 처음 푼 사람(본 대회): **shinhuichan**, 497분
- 처음 푼 사람(Open Contest): **ychangseok**, 34분
- 출제자: `ldhhello`

J. 트리와 뽀미



- 다음과 같이 DP를 정의해 문제를 해결할 수 있습니다.
- DP_t = 시간 t에 승찬이가 뽀미를 만날 때 시간 1에서 시간 t까지 승찬이가 뽀미를 만나는 횟수의 최댓값
- dist(u, v)를 정점 u에서 v까지의 거리로 정의할 때, 다음과 같은 점화식이 성립합니다.
- $DP_t = \max(DP_{t'}) + 1$ where $t' < t, \text{dist}(t, t') \leq t - t'$
- dist(u, v)를 $\mathcal{O}(N^2)$에 전처리하거나, 또는 LCA를 이용해 $\mathcal{O}(logN)$에 온라인으로 처리하면 각각 $\mathcal{O}(N^2 + T^2)$, $\mathcal{O}(T^2 logN)$ 시간에 DP를 계산할 수 있습니다.
- 이때 답은 $\max_{t=1}^{T} DP_t$ 입니다.

- 또는 다음과 같은 DP로 문제를 해결할 수 있습니다.
- $DP_{t,v}$ = 시간 t에 승찬이가 v번 장소에 있을 때 시간 1에서 시간 t까지 승찬이가 뽀미를 만나는 횟수의 최댓값
- v번 장소에서 인접한 장소들을 c라고 할 때, 다음과 같은 점화식이 성립합니다.
- $$DP_{t,v} = \begin{cases} \max(DP_{t-1,v}, DP_{t-1,c}) & C_t \neq v \\ \max(DP_{t-1,v}, DP_{t-1,c}) + 1 & C_t = v \end{cases}$$
- 위 DP는 $\mathcal{O}(NT)$에 계산할 수 있고, 이때 답은 $\max_{v=1}^{N} DP_{T,v}$ 입니다.

K. IZ*ONE Sequence

`constructive` `ad_hoc`

출제진 의도 – **Hard**

- 본 대회 – 제출 17번, 정답 5명 (정답률 29.412%)
- Open Contest – 제출 77번, 정답 18명 (정답률 23.377%)
- 처음 푼 사람(본 대회): **shinhuichan**, 784분
- 처음 푼 사람(Open Contest): **fermion5**, 45분
- 출제자: `ian0704`

K. IZ*ONE Sequence

- 이 문제는 다양한 풀이가 있을 수 있습니다.

- K가 왼쪽 끝에 고정된 상태에서 K가 삭제되려면, **오른쪽 끝에 있는 수가 $K+1$인 경우** 밖에 없습니다.
- 따라서 K를 왼쪽 끝에 고정하고 $K+1$을 먼저 없애는 방식을 시도해볼 수 있습니다.
- $K+1$을 가장 먼저 배열에서 삭제시키기 위해 K로 시작하고 $K+2$로 끝나는 구성을 생각할 수 있습니다.

- 하지만 이 구성을 사용할 때 주의할 점이 있습니다.

- $K \geq N - 1$ 인 경우, 오른쪽 끝에 $K + 2$ 을 배치할 수 없습니다.
- $K = N$ 인 경우에는 K 와 어떤 수의 평균을 내림한 값이 K 가 되는 것이 불가능하기 때문에, K 를 왼쪽 끝에 고정하기만 하면 해결됩니다.

- $N = K - 1$인 경우에는 K와 $K + 2$로 $K + 1$을 없애는 것이 불가능하므로, $K + 1$**이 오른쪽 끝에서 왼쪽 끝 값과의 평균을 내림한 값이 수열에 존재하지 않아**서 사라지도록 만들어야 합니다.
- 이러한 구성을 위한 예시로는 $1, N - 1, \cdots, 2, N$과 같이 구성하는 방식이 있습니다. 이 구성 방식에서는 1과 N의 평균을 내림한 값이 사라지며 이후 N이 사라지고, 왼쪽 끝에는 1이, 오른쪽 끝에는 2가 남아 최종적으로 1이 사라지게 되어 왼쪽 끝에 $N - 1$이 남고 N을 제거한 상태를 만들 수 있습니다
- 하지만 $N \leq 4$일 때는 이렇게 구성하는 것이 불가능하며, 해가 존재하지 않음을 알 수 있습니다.



L. 건물 폭파

`trees` `dp_tree`

출제진 의도 – **Challenging**

- 본 대회 – 제출 6번, 정답 2명 (정답률 33.333%)
- Open Contest – 제출 19번, 정답 9명 (정답률 47.368%)
- 처음 푼 사람(본 대회): **k_san**, 1998분
- 처음 푼 사람(Open Contest): **jeoffrey0522**, 60분
- 출제자: ecode

L. 건물 폭파

- 모든 $i\,(1 \le i \le N-1)$에 대하여 i번 정점과 $i+1$번 정점이 연결되어 있는 일자 트리를 생각해봅시다.
- 위 트리에서 전달된 폭발의 강도가 0인 정점이 존재하지 않도록 l번 정점과 r번 정점에 각각 강도 L, R의 폭발을 일으키는 상황을 가정해보겠습니다.
- 모든 $i\,(1 \le i \le N)$에 대하여 i번 정점에 전해지는 폭발의 강도는 $\max(L-|l-i|,0)+\max(R-|r-i|,0)$입니다.

L. 건물 폭파

- 이번에는 m 번 정점에 강도 $L+R$ 의 폭발을 일으키는 상황을 가정해보겠습니다.
- 모든 $i\,(1 \le i \le N)$ 에 대하여 i 번 정점에 전해지는 폭발의 강도는 $\max(L+R-|m-i|, 0)$ 입니다.

L. 건물 폭파

- 모든 $i\,(1 \le i \le N)$에 대하여 $\max(L - |l - i|, 0) + \max(R - |r - i|, 0) \le \max(L + R - |m - i|, 0)$이 성립하는 m을 $l \le m \le r$ 범위에서 무조건 찾을 수 있음이 보장됩니다.
- 이는 폭발을 일으키는 정점의 개수를 줄이는 것이 절대 손해가 아님을 의미합니다.
- 즉, 하나의 정점에서 한번의 폭발을 일으키는 것이 최적이며 일자 트리에서 일반적인 형태의 트리로 확장해도 이는 유효합니다.
- 도출 과정에 따른 자세한 증명은 생략합니다.

L. 건물 폭파

- 최소의 폭발 강도로 모든 건물을 무너뜨릴 수 있는 정점을 찾기 위해 DP를 이용할 수 있습니다.
- $dp[i]$를 모든 건물을 무너뜨리기 위해 i번 정점에 일으켜야할 폭발의 최소 강도로, $d_{i,j}$를 i번 정점과 j번 정점 사이의 간선 개수로 정의하겠습니다.
- $dp[i] = \min(A_1 + d_{i,1}, A_2 + d_{i,2}, \cdots, A_N + d_{i,N})$이 됩니다.

L. 건물 폭파

- 모든 $i\,(1 \leq i \leq N)$에 대하여 $dp[i]$의 최솟값이 문제의 정답입니다.
- 그러나 모든 $dp[i]$를 직접 탐색하면 $\mathcal{O}\left(N^2\right)$ 시간이 소요됩니다.
- 탐색 시간을 줄이기 위해서 전방향 트리 DP(트리 리루팅)를 사용하면 $\mathcal{O}\left(N\right)$ 시간에 문제를 해결할 수 있습니다.



M. 수열과 띄엄띄엄 쿼리

`segtree` `lazyprop`

출제진 의도 – **Challenging**

- 본 대회 – 제출 5번, 정답 2명 (정답률 40.000%)
- Open Contest – 제출 63번, 정답 7명 (정답률 11.111%)
- 처음 푼 사람(본 대회): **k_san**, 1551분
- 처음 푼 사람(Open Contest): **jh01533**, 121분
- 출제자: goodpjw2008

M. 수열과 띄엄띄엄 쿼리

- 이 문제는 인접한 연속된 배열에 작업을 하는 것이 아니라, 띄엄띄엄 떨어져 있는 값들을 일괄적으로 업데이트 및 쿼리를 해야 하는 문제입니다.
- 이러한 업데이트 및 쿼리는 일반적인 구간 트리로 처리하기 어렵습니다.
- 그렇다면 띄엄띄엄 구간을 처리할 수 있도록, 범위 쿼리의 특성을 최대한 살려서 문제를 풀어봅시다.



- A_1, A_2, A_3, A_4에 업데이트를 하고, A_1, A_3의 합을 구해야 하는 상황이라고 가정해봅시다.
- 이 경우 홀수 인덱스를 관리하는 구간 트리, 짝수 인덱스를 관리하는 구간 트리로 나눠서 작업을 해주면 됩니다.
- 같은 아이디어로 d가 6까지 늘어난다고 해도 구간 트리를 $\mathrm{lcm}(1,2,3,4,5,6) = 60$개 만들고 $0 \le j \le 59$인 j번째 구간 트리가 $i \equiv j \pmod{60}$인 인덱스 i들을 관리하도록 하면
- 구간 업데이트, 쿼리는 각 구간 트리당 $O(\log N/60)$에 해결할 수 있고, 구간 트리는 60개이므로 총 시간복잡도 $O(60Q\log(N/60))$에 해결할 수 있습니다.

N. 디미교도소

`segtree`
출제진 의도 – **Challenging**

- 본 대회 – 제출 11번, 정답 0명 (정답률 00.000%)
- Open Contest – 제출 9번, 정답 2명 (정답률 22.222%)
- 처음 푼 사람(본 대회): **N/A**
- 처음 푼 사람(Open Contest): **jh01533**, 194분
- 출제자: deom

- 먼저 모든 죄수가 탈출할 수 없는 경우에 대해 살펴봅시다.
- i번 죄수의 목적지 E_i가 $i < E_i$이면서 j번 죄수의 목적지 E_j가 $j < E_j$인 경우를 생각해봅시다.
- 여기서 $i < j$일 때 $E_j < E_i$라면 j번 죄수가 이동하는 경로가 i번 죄수의 이동경로에 간섭을 받아 j번 죄수가 E_j로 가지 못하게 됩니다.
- 그 반대의 조건도 성립함을 알 수 있습니다.

N. 디미교도소

- 굴 a와 b사이에 샛길을 하나 추가한다면 a번 죄수와 b번 죄수의 위치가 교환된다고 볼 수 있습니다.
- 일반적인 사다리타기 규칙으로 볼 때 최소 개수의 샛길은 순열 E에서 역순쌍의 개수와 같다는 것을 보일 수 있습니다.
- 이 문제에는 탈옥 규칙 3,4가 예외적으로 작용하는 것처럼 보입니다.

- 규칙 3에 의해 이동방향은 한방향으로 고정되기 때문에 결국 목적지에 도착하기 위해서는 반드시 $|i - E_i|$ 개의 샛길이 사용된다는 것을 파악할 수 있습니다.
- $i = E_i$ 인 경우를 생각해봅시다.
- $j < i < E_j$ 인 j 번 죄수가 있다면 i 번 죄수의 마지막 위치는 $i - 1$ 이 됩니다.
- 만약 $E_k < i < k$ 인 경우가 존재한다면 i 번 죄수는 규칙 4에 의해 i 굴에서 탈출할 수 있습니다.
- 그러한 k 가 없다고 하여도 결국 역순쌍의 개수가 최소 개수가 됨을 알 수 있습니다.
- 따라서 세그먼트 트리를 사용해 $\mathcal{O}(N \log N)$ 에 문제를 해결할 수 있습니다.

O. 비트 뒤집기와 쿼리

`data_structures segtree trie lazyprop smaller_to_larger`
출제진 의도 – **Challenging**

- 본 대회 – 제출 5번, 정답 0명 (정답률 00.000%)
- Open Contest – 제출 3번, 정답 1명 (정답률 33.333%)
- 처음 푼 사람(본 대회): **N/A**
- 처음 푼 사람(Open Contest): **jthis**, 79분
- 출제자: seungchan0325

- 리프 노드 $0, 1, \cdots, 2^{20}-1$을 가지는 세그먼트 트리를 구성합니다.
- 이때 세그먼트 트리는 비트 문자열의 트라이와 같습니다.
- 리프 노드 i에 정수 i의 개수를 저장합시다.
- 노드 u에 u가 관리하는 범위 $[s, e]$ 사이의 정수 개수를 저장합니다.
- $C = 2^{20}$이라고 합시다.
- 2번 쿼리는 세그먼트 트리에서의 이분탐색으로 $\mathcal{O}(\log C)$에 처리할 수 있습니다.

- $l = 0, r = 2^{20} - 1$ 인 경우를 살펴봅시다.
- 루트 노드의 높이는 19, 노드 u 의 높이는 u 의 부모 노드의 높이 -1 이라고 합시다.
- 높이가 k 인 노드의 왼쪽 자식과 오른쪽 자식을 바꾸면 됩니다.
- 루트 노드에서부터 k 번째 비트가 뒤집혔다는 정보를 느리게 전파하여 해결할 수 있습니다.
- 이 경우 시간 복잡도는 $\mathcal{O}(1)$ 입니다.

- 노드 u가 관리하는 범위를 $[s, e]$, u의 높이를 d라고 합시다.
- $l = s, r = e$인 경우를 살펴봅시다.
- 이때 두 가지 경우로 나눌 수 있습니다.
  1. $d > k$
  2. $d \leq k$
- 1번의 경우에는 $l = 0, r = 20^{20} - 1$인 경우와 동일하게 해결할 수 있습니다.
- 이 경우 시간 복잡도는 노드 u를 찾는 시간 복잡도인 $\mathcal{O}(\log C)$입니다.

- 2번의 경우에는 다음과 같이 해결할 수 있습니다.
  - u의 조상 중 높이가 k인 노드를 v, u를 v 기준으로 대칭한 위치의 노드를 w라고 합시다.
  - u의 서브 트리를 w의 서브 트리에 병합합니다.
- 노드 u, v, w를 찾는 시간 복잡도는 $\mathcal{O}(\log C)$입니다.

- u, w의 서브트리를 각각 a, b라고 합시다.
- a를 b에 병합하는 데 걸리는 시간은 $\mathcal{O}(\min(|a|, |b|))$입니다.
- 모든 수가 병합되는데 걸리는 시간은 $\mathcal{O}(N \cdot \log C \cdot \log N)$입니다.
- 따라서 시간복잡도는 $\mathcal{O}(Q \cdot \log C + N \cdot \log C \cdot \log N)$입니다.
- 관리하는 구간에 정수가 없는 노드 삭제하지 않으면 시간 복잡도가 보장되지 않습니다.