2025 UDPC

Official Solutions

UDPC 2025 - 출제진 및 검수진

- ✓ 출제진
  - 곽민성 minsung05
  - 권찬우 kwoncycle
  - 김도훈 99asdfg
  - 김현서 nukec
  - 박신욱 bnb2011
  - 우은규 ekwoo
  - 장래오 leo020630
  - 정윤성 slah007
  - 최진우 andrew1010
  - 한동규 queued_q
- ✓ 검수진
  - jinhan814
  - jthis
  - playsworld16
  - tony9402
  - utilforever

Junior Division	문제	의도한 난이도	출제자
A	CPDU	**Easy**	장래오 leo020630
B	포린드롬	**Easy**	권찬우 kwoncycle
C	Arkain 대시보드	**Easy**	한동규 queued_q
D	C)	**Medium**	장래오 leo020630
E	가디언 엔젤 슬라임	**Medium**	장래오 leo020630
F	선물 보내기	**Medium**	박신욱 bnb2011
G	UDP 문자열	**Medium**	김도훈 99asdfg
H	드론 라이트 쇼	**Hard**	한동규 queued_q
I	트리오	**Hard**	장래오 leo020630

Senior Division	문제	의도한 난이도	출제자
A	CPDU	**Easy**	장래오 leo020630
B	포린드롬	**Easy**	권찬우 kwoncycle
C	Arkain 대시보드	**Easy**	한동규 queued_q
D	해밍 거리	**Medium**	한동규 queued_q
E	가디언 엔젤 슬라임	**Medium**	장래오 leo020630
F	선물 보내기	**Medium**	박신욱 bnb2011
G	드론 라이트 쇼	**Hard**	한동규 queued_q
H	지역번호 정하기	**Hard**	장래오 leo020630
I	C)와 쿼리	**Hard**	장래오 leo020630

1A/2A. CPDU

`implementation, string`

출제진 의도 – **Easy**

- 통계:
  - Junior: 제출 36번, 정답 22번, 정답률 61.111%
  - Senior: 제출 22번, 정답 22번, 정답률 100.000%
- First Solve:
  - Junior: **yjun**POSTECH, 0분
  - Senior: **벌레캠프호날두**POSTECH, 0분
- 출제자: 장래오`leo020630`

1A/2A. CPDU

- 문자열 N 개를 입력받은 후 각 문자열이 알파벳 C로 시작하는지 확인하면 됩니다.

1B/2B. 포린드롬

`bruteforcing`, `math`
출제진 의도 – **Easy**

- ✓ 통계:
  - Junior: 제출 48번, 정답 17번, 정답률 35.417%
  - Senior: 제출 62번, 정답 22번, 정답률 33.817%
- ✓ First Solve:
  - Junior: **outfinity**[UNIST], 5분
  - Senior: **과메기단**[UNIST], 4분
- ✓ 출제자: 권찬우[`kwoncycle`]

1B/2B. 포린드롬

- 먼저 0 부터 9 까지의 수들은 모두 포린드롬 수입니다.
- 자리수가 2 이상인 수가 포린드롬 수이려면, 각 자리수의 숫자가 모두 같아야 합니다.
- 자리수가 t 인 포린드롬 수 N 의 i 번째 숫자를 a_i 라고 합시다.
- 이때, $a_1a_2...a_t$ 가 팰린드롬이어야 하므로 $a_i = a_{t+1-i}$ 여야 하며, 또한 $a_1a_2...a_{t-1}$ 도 팰린드롬이어야 하므로 $a_{i-1} = a_{t+1-i}$ 이어야 합니다.
- 즉, 각 i 에 대해 $a_{i-1} = a_i$ 이며, $a_1 = a_2 = ... = a_t$ 가 성립합니다.
- 따라서 특정 자리수를 가지는 포린드롬은 많아야 10 개 존재하며, 문제의 제한 내로 들어오는 포린드롬의 개수는 최대 82 개입니다. 이 숫자들 중 입력 N 이하인 것의 개수를 세면 됩니다.

1C/2C. Arkain 대시보드

`implementation`, `hash_set`, `sorting`
출제진 의도 – **Easy**

- ✓ 통계:
  - Junior: 제출 37번, 정답 14번, 정답률 37.838%
  - Senior: 제출 45번, 정답 19번, 정답률 42.222%
- ✓ First Solve:
  - Junior: **outfinity**[UNIST], 9분
  - Senior: **과메기단**[UNIST], 11분
- ✓ 출제자: 한동규[`queued_q`]

1C/2C. Arkain 대시보드

- ✓ 먼저 각 컨테이너마다 마지막으로 사용한 시점을 알아야 합니다.
- ✓ i번째 사용 로그를 입력받을 때마다, 그 컨테이너를 마지막으로 사용한 시점을 i로 업데이트합시다.
  - `last_use[container] = i`
- ✓ 이때 `container`는 정수 인덱스가 아니라 문자열이므로 해시 테이블을 사용해야 합니다.
  - C++에서는 `map` 또는 `unordered_map`,
  - Python에서는 `dict` (dictionary)를 사용하면 됩니다.

1C/2C. Arkain 대시보드

- ✓ 모든 컨테이너를 `last_use`가 작아지는 순서로 정렬합시다.
- ✓ 모든 `(last_use[container], container)` 쌍을 pair (C++)나 tuple (Python)로 묶어 정렬한 뒤 순서를 뒤집으면 됩니다.
- ✓ 이 밖에도, 로그를 역순으로 확인하면서 한 번 확인했던 컨테이너는 건너뛰는 식으로 정렬 순서를 구하는 방법도 있습니다.

1C/2C. Arkain 대시보드

- ✓ 이제 고정된 컨테이너와 그렇지 않은 컨테이너를 분리해야 합니다.
- ✓ 정렬된 목록을 순서대로 확인하면서, 고정된 컨테이너라면 `pinned` 배열에, 그렇지 않으면 `unpinned` 배열에 추가합시다.
- ✓ 이때 각각이 고정된 컨테이너인지 여부를 빠르게 판단해야 합니다.
  - 단순히 고정된 컨테이너 배열에서 컨테이너 이름을 탐색하는 것은 시간 복잡도가 $\mathcal{O}(K)$ 이므로 느립니다.
  - 주의: C++의 `find` 함수나 Python의 `in` 연산자를 사용해도 마찬가지입니다.

- ✓ 고정된 컨테이너 목록도 해시 테이블에 저장하면 이 문제를 해결할 수 있습니다.
  - `is_pinned[container] = true`
- ✓ 해시 테이블에 값을 저장하고 불러오는 것은 시간 복잡도가 $\mathcal{O}(1)$로 빠릅니다.
- ✓ 컨테이너가 고정되어 있는지는 다음과 같이 확인할 수 있습니다.
  - C++: `is_pinned[container]`
  - Python: `container in is_pinned`
- ✓ 이 밖에도, `set`이나 `unordered_set`과 같은 해시 집합을 사용해도 됩니다.
- ✓ `pinned` 배열과 `unpinned` 배열을 합쳐 출력하면 문제를 해결할 수 있습니다.
- ✓ 전체 시간복잡도는 구현에 따라 $O(N \log N)$ 또는 $O(N)$이 됩니다.

2D. C)

`greedy`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 39번, 정답 13번, 정답률 33.333%
- First Solve:
  - Junior: **outfinity**[UNIST], 17분
- 출제자: 장래오[`leo020630`]

2D. C)

- C를 닫는 괄호로 사용할 경우 2의 비용이 들며, U는 어느 방향으로 사용해도 1의 비용이 듭니다.
- 따라서 닫는 괄호로 사용하는 C의 개수를 최소화해야 합니다.
- 만약 닫는 괄호로 사용하는 C의 개수 k를 결정하면 가장 뒤에 위치한 k개의 C를 닫는 괄호로 사용하는 것이 최적입니다.
- 이러면 닫는 괄호로 사용하는 U의 개수 역시 $\frac{|S|}{2} - k$로 결정되며 역시 가장 뒤에 위치한 U들을 닫는 괄호로 사용하는 것이 최적입니다.

2D. C)

- 다음과 같은 과정을 반복합시다.
- 초기에 모든 C는 여는 괄호, 모든 U는 닫는 괄호라 가정합시다.
- 여는 괄호를 -1, 닫는 괄호를 $+1$으로 정의합시다.
- 문자열을 뒤쪽부터 순회하며 Suffix Sum이 0 미만이 되는 순간, 그 C는 닫는 괄호로 사용되어야만 합니다.
- 이러한 방식으로 닫는 괄호로 사용하는 C의 개수를 결정할 수 있으며, 실제 문자열도 앞 슬라이드에서 설명한 방식으로 구성할 수 있습니다.

1D. 해밍 거리

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

- 통계:
  - Senior: 제출 37번, 정답 9번, 정답률 24.324%
- First Solve:
  - Senior: **fm9779**[UNIST], 35분
- 출제자: 한동규[queued_q]

1D. 해밍 거리

- 구하려는 두 수를 X, Y 라고 합시다. $(A \leq X \leq Y \leq B)$
- A, B 의 상위 비트부터 보았을 때, 처음으로 달라지는 위치를 찾습니다. 해당 자리까지의 공통 비트열을 S 라고 합시다.
- 해당 자리까지 X, Y 의 상위 비트열도 S 와 동일해야 합니다.
- 그 아래의 자리는 X, Y 의 모든 비트가 다르게 만드는 것이 가능합니다!

1D. 해밍 거리

- 다음과 같이 구성하면 됩니다.
  - $X = S0111\ldots1$
  - $Y = S1000\ldots0$
- A와 B의 S 다음으로 오는 비트가 서로 다르므로 각각 0과 1일 것이고,
- X, Y는 각각 $S0$으로 시작하는 수 중 가장 큰 값, $S1$로 시작하는 수 중 가장 작은 값이므로 항상 $A \leq X \leq Y \leq B$를 만족합니다.
- 이를 적절히 구현하면 $O(\log B)$ 시간에 문제를 해결할 수 있습니다.

1E/2E. 가디언 엔젤 슬라임

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

- ✓ 통계:
  - Junior: 제출 43번, 정답 10번, 정답률 23.256%
  - Senior: 제출 36번, 정답 8번, 정답률 22.222%
- ✓ First Solve:
  - Junior: **outfinity**[UNIST], 37분
  - Senior: **sg549**[DGIST], 48분
- ✓ 출제자: 장래오[leo020630]

1E/2E. 가디언 엔젤 슬라임

- 각 슬라임은 두 가지 경우 중 하나에 속하게 됩니다.
  - 1. X 이상 크기에 도달하지 못한다.
  - 2. $S < E$인 두 정수 S, E가 존재해서 $[S, E]$ 구간에 속하는 날짜에는 크기가 X 이상이다.
- 두 가지 경우 판단, 후자일 경우 S와 E 계산 모두 간단한 수식 계산으로 할 수 있습니다.
- S와 E를 모두 구했다면 이를 모두 정렬해 누적합을 계산하는 것으로 특정 날짜 구간에서 크기가 X 이상인 슬라임의 수를 구할 수 있습니다.
- 각 날짜 구간에 대해서 크기가 X 이상인 슬라임의 수가 3마리 이상이라면 정답에 더해 줍시다.
- 시간 복잡도는 정렬에 의존하며, $\mathcal{O}(N \log N)$입니다.
- 가능한 날의 수가 int 범위를 초과할 수 있음에 유의합시다.

1F/2F. 선물 보내기

`graph_traversal`
출제진 의도 – **Medium**

- ✓ 통계:
  - Junior: 제출 22번, 정답 6번, 정답률 27.273%
  - Senior: 제출 23번, 정답 5번, 정답률 21.739%
- ✓ First Solve:
  - Junior: **outfinity**[UNIST], 49분
  - Senior: **과메기단**[UNIST], 67분
- ✓ 출제자: 박신욱[bnb2011]

1F/2F. 선물 보내기

- 문제의 조건을 다음과 같이 해석합니다.
  - i번 선물을 윤이에게 보내는 경우 0, 포닉스에게 보내는 경우 1을 i번 정점의 값으로 둔다고 생각합시다.
  - 그러면 U, D, P의 각 조건은 a_i번 정점과 b_i번 정점에 할당된 값의 합이 각각 $0, 1, 2$가 되어야 함과 같습니다.
- 그래프의 각 연결 컴포넌트에서 한 정점의 값을 고정하면 그와 연결된 다른 모든 정점의 값 역시 자동으로 정해집니다.
- 정할 수 있는 값은 0 또는 1 중 하나이므로, 두 경우를 모두 시도해보고 조건을 만족하는 경우의 수를 전체 답에 곱해주면 됩니다.
- 시간복잡도는 $\mathcal{O}(N+M)$입니다.

2G. UDP 문자열

`dp`, `combinatorics`
출제진 의도 – **Hard**

- ✓ 통계:
  - Junior: 제출 28번, 정답 8번, 정답률 28.571%
- ✓ First Solve:
  - Junior: **equinox**[POSTECH], 72분
- ✓ 출제자: 김도훈[99asdfg]

2G. UDP 문자열

- ✓ 완전 UDP 문자열은 어떤 두 UDP 문자열을 이어붙여도 만들어져선 안됩니다.
- ✓ UDP 문자열의 정의에 의해, 앞의 부분 문자열이 UDP 문자열이라면 나머지 문자열도 UDP 문자열이 됩니다.
- ✓ 따라서, 위 조건은 $1 \leq i \leq 3N-1$ 인 모든 i 에 대해 $[1, i]$ 구간의 부분 문자열이 UDP 문자열이 아니라는 조건과 동일합니다.
- ✓ 또한, UDP 문자열의 정의 상 반드시 모든 UDP 문자열의 길이는 3 의 배수입니다.
- ✓ 위 조건을 활용해서 DP 테이블을 만들어 봅시다.

2G. UDP 문자열

- $DP[3i] : [1, 3i]$ 구간이 처음으로 부분 문자열이 되는 UDP 문자열의 경우의 수
- $f(x)$: 크기가 $3x$ 인 UDP 문자열의 개수
- 로 정의하고, $DP[0] = 1$ 로 정의하면 다음과 같은 계산식을 구할 수 있습니다.
- $f(x) = \frac{3x!}{(x!)^3}; dp[x] = f(x) - dp[1]f(x-1) - \cdots - dp[x-1]f(1)$
- $f(x)$ 값은 단순 계산으로 $\mathcal{O}\left(N^2\right)$ 로 미리 계산해둘 수 있으므로, $dp[1]$ 값부터 차례대로 채워 나가면 총 시간복잡도 $\mathcal{O}\left(N^2\right)$ 로 문제를 해결할 수 있습니다.
- $f(x)$ 값을 계산하는 도중 값이 너무 커질 수 있으므로, 모듈로 역원을 사용하여 계산을 빠르게 진행할 수 있습니다.

1G/2H. 드론 라이트 쇼

`offline_queries, dp, graph_traversal`
출제진 의도 – **Hard**

- ✓ 통계:
  - Junior: 제출 8번, 정답 6번, 정답률 75.000%
  - Senior: 제출 8번, 정답 4번, 정답률 50.000%
- ✓ First Solve:
  - Junior: **van.li**[UNIST], 80분
  - Senior: **벌레캠프호날두**[POSTECH], 102분
- ✓ 출제자: 한동규[queued_q]

1G/2H. 드론 라이트 쇼

- 서버의 명령을 전부 순서대로 시뮬레이션하면 시간복잡도가 $\mathcal{O}(MQ)$ 로 매우 느립니다.
- 모든 명령 정보를 미리 입력받을 수 있으므로, 그때그때 처리하는 대신 오프라인 쿼리의 아이디어를 활용합시다.
- 명령에 순서대로 $1, 2, \ldots, Q$ 번호를 붙이겠습니다.

1G/2H. 드론 라이트 쇼

- ✓ 우선 문제를 단순화해서 아래 방향으로 전파하는 명령만 입력으로 들어온다고 가정합니다.
- ✓ x 번 드론의 위 방향으로 직접 연결된 드론들을 $y_1, \ldots, y_k$ 라고 합시다.
- ✓ y_i 번 드론은 명령을 수행하고 나면 x 번 드론에게 명령을 전파할 것입니다.
- ✓ 따라서 x 번 드론이 마지막으로 **수행한** 명령은,
  - x 번 드론이 마지막으로 **서버로부터 받은** 명령과,
  - y_i 번 드론들이 마지막으로 **수행한** 명령들 중 가장 번호가 큰 명령입니다.
- ✓ 이는 N 번부터 1 번까지 드론을 순회하며 DP를 통해 구할 수 있습니다.

1G/2H. 드론 라이트 쇼

- ✓ 이제 위 방향으로 전파하는 명령도 있는 있는 상황에서 문제를 풀어 봅시다.
- ✓ 두 방향에 대해 각각 따로 DP를 수행하면, x번 드론이 수행한 가장 마지막 아래 방향 명령과 가장 마지막 위 방향 명령의 번호를 구할 수 있습니다.
- ✓ 둘 중에 더 번호가 큰 명령의 불빛 색깔이 x번 드론의 색깔이 됩니다.
- ✓ 전체 시간복잡도는 $\mathcal{O}(M+Q)$입니다.
- ✓ 각 방향에 대해 쿼리의 역순으로 그래프 탐색을 하는 방법으로도 문제를 해결할 수 있습니다.

1H. 지역 코드 정하기

`dp_bitfield`, `pigeonhole_principle`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 2번, 정답 0번, 정답률 0.000%
- First Solve:
  - Senior: -
- 출제자: 장래오 `leo020630`

1H. 지역 코드 정하기

- ✓ 몇 개의 문자열을 골라 문자 단위로 재배열해 회문을 만들 수 있는 조건에 대해 생각해 봅시다.
- ✓ 모든 문자가 짝수번 등장하거나, 홀수번 등장하는 문자가 정확히 하나 있으면 가능합니다.
- ✓ 문자열에서 중요한 정보는 각 문자의 등장 횟수에 대한 홀짝성이기 때문에, 각 문자열을 0에서 $1023 = 2^{10} - 1$까지의 이진수로 표현할 수 있습니다.
- ✓ 문자열 여러 개를 합치는 것은 이진수들 간의 Bitwise XOR 연산으로 해석할 수 있습니다.

1H. 지역 코드 정하기

- 같은 이진수 값을 가진 문자열 2개를 쓰면 항상 회문을 만들 수 있습니다.
- 이 경우 외에는 사용하는 문자열의 이진수 값이 모두 다른 경우만 고려해주면 되므로, 이를 예외로 처리합시다.
- 남은 과정은 DP로 해결할 수 있습니다.
- $dp[i][j]$ = i까지의 이진수 값을 가진 문자열 중 몇 개를 적절히 골라 XOR한 값이 j일 때 길이 합의 최솟값과 같이 정의하면 됩니다.
- DP 테이블의 크기가 $2^{10} \times 2^{10} = 2^{20}$이며 전이를 $\mathcal{O}(1)$에 할 수 있으므로 시간 복잡도는 $\mathcal{O}(2^{20})$입니다.
- 이진수 값이 0인 문자열의 사용과 역추적 처리에 유의하며 구현하면 됩니다.

11. 트리오

`trees`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 16번, 정답 2번, 정답률 12.500%
- First Solve:
  - Senior: **outfinity**[UNIST], 140분
- 출제자: 장래오[leo020630]

11. 트리오

- 먼저 없앨 수 있는 간선을 모두 구해 봅시다. 크게 두 가지 풀이로 가능합니다.
- 1. 관찰 + LCA 응용
- A, B, C의 역배열을 P, Q, R이라 합시다.
- $1 \leq i \leq N$인 각 i에 대해 P_i, Q_i, R_i의 세 정점을 모두 포함하는 가장 작은 서브트리에 속한 간선은 잘릴 수 없습니다. 이에 속한 간선을 자르는 순간 i가 서로 다른 서브트리에 속하는 경우가 생기기 때문입니다.
- 따라서 각 i에 대해 사용할 수 없는 간선을 표시해주면 됩니다.
- 이는 각 간선에 대해 이 간선이 잇는 부모 정점과 자식 정점을 서로 합쳐주는 식으로 가능합니다.
- 이 과정에서 두 정점의 최소 공통 조상을 구해야 하며, 각 간선은 최대 1번만 체크되므로 시간 복잡도는 $\mathcal{O}(N \log N)$입니다.

- 2. 집합을 직접 관리
- DFS 과정에서 서브트리에 속한 A_i, B_i, C_i의 세 집합을 모두 관리합시다.
- 집합을 합치는 것은 Small to Large 테크닉이라는 잘 알려진 방법으로 구현할 수 있습니다.
- 단, 집합을 직접 비교한다면 시간 초과를 받게 됩니다.
- 각 집합의 배타적 논리합을 관리해 이 집합이 빔을 확인하거나, 각 원소의 등장 횟수를 관리하면 됩니다.
- 시간 복잡도는 $\mathcal{O}\left(N \log^2 N\right)$ 입니다.

- 집합을 직접 관리하지 않는 방법도 있습니다.
- 2^{60} 정도 크기의 정수 M 을 정한 후, 각 수에 $[0, M]$ 범위의 랜덤한 정수를 할당합시다.
- 두 집합의 일치 여부를 곧 두 집합의 XOR 값이 일치하는지로 판별할 수 있습니다. 이러한 기법을 XOR Hashing 또는 Zobrist Hashing이라 합니다.
- 시간 복잡도는 $\mathcal{O}(N \log N)$ 입니다.

11. 트리오

- 이제 트리를 세 부분으로 나누어 봅시다.
- 먼저, 나눌 수 있는 간선이 하나 이하라면 불가능하므로 -1을 출력합니다.
- 그렇지 않다면, 이분 탐색으로 답을 찾을 수 있습니다.
- 모든 서브트리의 크기가 X 이상이도록 나눌 수 있는 최대 서브트리의 수가 3 이상인가? 라는 결정 문제를 해결합시다. 이는 서브트리의 크기가 X 이상이면 바로 잘라주는 전략을 이용한 트리 DP로 해결할 수 있습니다.
- 결정 문제의 답이 YES인 X 중 최댓값이 답이 되며, 시간 복잡도는 $\mathcal{O}(N \log N)$ 입니다.
- 그 외에도 Small to Large 방법을 이용해 두 개의 자를 간선을 모두 고려하는 것으로도 해결할 수 있습니다.

11. C)와 쿼리

`segtree`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 0번, 정답 0번, 정답률 -%
- First Solve:
  - Senior: -
- 출제자: 장래오`leo020630`

- 기본적으로 C) 문제의 풀이에 사용되는 성질은 모두 사용됩니다.
- 닫는 괄호로 사용되는 C의 개수를 비용으로 정의합시다. 또한, 닫는 괄호로 사용되는 C를 **닫는 C**와 같이 표기하겠습니다.
- **Observation 1.** 가장 먼저 등장하는 닫힌 C 이후에 등장하는 모든 U는 닫힌 U이다.
- **증명** 닫힌 C 뒤에 열린 U가 등장한다면 바꿔주는 것이 항상 이득입니다.
- **reduce**라는 행동을 정의합시다. 이는 가장 먼저 닫힌 C를 열고 마지막으로 열린 U를 닫는 것이 가능한지 판단하고, 가능하면 시행하는 행동입니다. reduce에 성공할 경우 비용은 1 감소합니다.

C가 U로 바뀌는 경우

- ✓ 1. 바뀌는 문자가 닫힌 C였을 경우
- ✓ 바뀐 후에도 닫힌 U인 상태로 두어야 합니다. 이때 비용이 1 감소합니다.
- ✓ 2. 바뀌는 문자가 열린 C였을 경우
- ✓ 우선 열린 U로 둔 후, 앞쪽에 닫힌 U가 있는 경우 가장 앞의 닫힌 U와 위치를 바꿉니다. 이후 reduce를 시도합니다.

U가 C로 바뀌는 경우

- ✓ 1. 바뀌는 문자가 열린 U였을 경우
- ✓ 바뀐 후에도 열린 C인 상태로 두어야 합니다.
- ✓ 2. 바뀌는 문자가 닫힌 U였을 경우
- ✓ 우선 닫힌 C로 둔 후, 뒤쪽에 열린 C가 있는 경우 가장 뒤의 열린 C와 위치를 바꿉니다. 이때 비용이 1 증가합니다. 이후 reduce를 시도합니다.

- ✓ 각 쿼리에서 비용은 최대 1 증가합니다. 따라서 reduce의 총 횟수가 $\mathcal{O}(N+Q)$임을 알 수 있습니다.
- ✓ 누적합을 저정하는 Lazy Segment Tree 등의 자료구조를 이용해 모든 과정을 시행할 수 있습니다.
- ✓ 시간 복잡도는 $\mathcal{O}((N+Q)\log N)$입니다.
- ✓ 이외에도 Segment Tree의 각 원소를 잘 정의하는 방법으로 문제를 해결할 수 있습니다.