제 4회 보라매컵 풀이

by

Official Solutions

문제		의도한 난이도	출제자
A	국방시계	**Very Easy**	chattermin, amel
B	나무와 그림자 easy	**Easy**	sjimed, amel
C	특식 배분	**Easy**	moonrabbit2
D	소등 시간	**Medium**	serin
E	배틀쉽	**Medium**	kaorin
F	휴가 계획	**Hard**	serin
G	나무와 그림자 hard	**Hard**	sjimed, amel



A. 국방시계

`math`

출제진 의도 – **Very Easy**

- 제출 54번, 정답자 21명 (정답률 51.85%)
- 처음 푼 사람: **버스충**, 2분
- 출제자: `chattermin`, `amel`

A. 국방시계



- 00시 00분부터 24시 00분까지를 $24 \times 60 = 14400$분으로 생각할 수 있습니다.
- 현재 총 N일 중 M일을 복무하였으므로 14400분의 $\frac{M}{N}$ 만큼 지났습니다.
- 분 단위까지만 출력하면 되기 때문에 연산 결과를 버림 하여 지난 분을 계산하면 되겠습니다.
- 시간은 60으로 나눈 몫, 분은 60으로 나눈 나머지를 출력하면 됩니다.



B. 나무와 그림자 easy

`sorting`
출제진 의도 – **Easy**

- 제출 51번, 정답자 12명 (정답률 23.53%)
- 처음 푼 사람: **18생들**, 11분
- 출제자: `sjimed, amel`

B. 나무와 그림자 easy



- 여러 가지 접근 방법이 있습니다.
- 가장 떠올리기 쉽고 풀기 쉬운 방법을 소개하겠습니다.
- 먼저 나무에 의해 그림자가 질 때 햇빛을 좌표평면 상의 직선으로 생각할 수 있습니다.
- 위치 x_i, 높이 h_i 일 때 직선의 방정식은 $t(x - x_i) + (y - h_i) = 0$ 입니다.
- y 절편인 $tx_i + h_i$ 만 저장해주면 그림자를 표현할 수 있습니다.

B. 나무와 그림자 easy



- 먼저 모든 나무를 위치 기준으로 정렬합니다.
- 왼쪽부터 보는데, 지금까지 본 나무들에 의해 생기는 그림자 중 가장 긴 S 를 저장할 것입니다.
  - 이는 위에서 살펴봤듯 y 절편을 저장하는 방식으로 구현할 수 있습니다.
- i 번째 나무 T_i 를 볼 때, 먼저 S 에 의해 T_i 에 지는 그림자의 길이를 구합니다.
  - 그림자 S 는 위치 x_i 에서 높이 $S - tx_i$ 를 가집니다.
  - 그림자가 지지 않을 수도, 나무 전체에 그림자가 질 수도 있다는 점에 유의합시다.
- S 보다 T_i 에 의해 생기는 그림자가 더 긴 지 비교한 뒤, 더 길다면 S 를 바꿔줍니다.
- 이를 $i = 1, \cdots, N$ 까지 반복하여 정답을 구합니다.

C. 특식 배분

`dp`, `prefix_sum`
출제진 의도 – **Easy**

- 제출 70번, 정답자 8명 (정답률 11.43%)
- 처음 푼 사람: **국대와 아이들**, 11분
- 출제자: `moonrabbit2`

C. 특식 배분

- E_i 를 특식이 i 개 남아있을 때 기댓값이라고 합시다.
- 문제의 조건에 따라, $E_i = 1 + \frac{1}{K_i} \sum_{x=1}^{K_i} E_{i-x}$ 의 DP 식을 세울 수 있습니다.
- 그러나 이를 naïve하게 계산하면 $\mathcal{O}(N^2)$ 의 시간복잡도를 가져 시간 초과를 받습니다.
- 누적합을 사용하여 $\sum_{x=1}^{K_i} E_{i-x}$ 부분을 $\mathcal{O}(1)$ 에 계산할 수 있습니다.

C. 특식 배분



- 더 구체적으로, $S_i = \sum_{x=1}^{i} E_x$ 라 합시다.
- 그러면 $E_i = \begin{cases} 1 + \dfrac{1}{K_i}(S_{i-1} - S_{i-K_i-1}) & \texttt{if } K_i < i \\ 1 + \dfrac{S_{i-1}}{K_i} & \texttt{if } K_i = i \end{cases}$ 가 됩니다.
- 그리고 $S_i = S_{i-1} + E_i$ 로 S 배열을 update 해줄 수 있습니다.
- 따라서 $\mathcal{O}(N)$ 의 시간복잡도로 문제를 해결할 수 있습니다.



D. 소등 시간

`ad_hoc, case_work`
출제진 의도 – **Medium**

- ✓ 제출 37번, 정답자 4명 (정답률 10.81%)
- ✓ 처음 푼 사람: **국대와 아이들**, 40분
- ✓ 출제자: `serin`

D. 소등 시간



- 먼저 첫 번째 열의 전구가 모두 꺼지도록 연산을 적절히 수행해 줍시다.
- 전구가 가장 많이 켜진 열 C을 봅시다.
- 열 C에 켜져 있는 전구의 개수에 따라 경우를 나눕시다.

D. 소등 시간



- 만약 3개 이상이 켜져있다면, 정답은 0입니다.
- 만약 2개가 켜져있다면, 반드시 켜진 전구가 포함된 두 행 중 정확히 하나에 연산을 수행하여야 합니다. 따라서 2개의 후보가 존재합니다.
- 만약 1개가 켜져있다면, 더 이상 연산을 수행하지 않거나 켜진 전구가 포함된 행에 연산을 수행하는 2개의 후보가 존재합니다.
- 만약 모든 전구가 꺼져있다면, 아무 한 행에 연산을 수행해도 되기 때문에 정답은 $N+1$입니다.

D. 소등 시간



- 가장 처음에 첫 번째 열의 전구가 모두 꺼지도록 연산을 수행해 주는 것은 $\mathcal{O}(NM)$ 에 가능합니다.
- 하나의 후보에 대해서 가능한 지 확인하는 것도 $\mathcal{O}(NM)$ 에 쉽게 가능합니다.
- 확인해봐야 하는 후보가 최대 2 개이므로 $\mathcal{O}(NM)$ 의 시간복잡도에 문제를 해결할 수 있습니다.



E. 배틀쉽

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

- 제출 9번, 정답자 3명 (정답률 33.33%)
- 처음 푼 사람: **여기에팀명을입력**, 76분
- 출제자: `kaorin`

E. 배틀쉽



- 벽이 아닌 칸은 서로 이동이 가능하기 때문에 답은 항상 YES 입니다.
- 또한 한 번의 이동마다 하나의 새로운 칸에 말이 놓여지기 때문에 정확히 $NM - K - 1$ 번의 행위가 필요합니다.
- 말을 한꺼번에 들고 가야 하기 때문에, 나중에 몇 개를 사용할 지 미리 계산하여야 합니다.
- dfs tree를 사용하면 재귀적인 과정을 통해 이 값을 구해낼 수가 있습니다.

E. 배틀쉽



- (R, C)를 루트로 삼아서 dfs를 진행합니다. 이때 부모노드를 기준으로 어떤 방향으로 이동했는지를 자식노드에게 넘겨줍니다.
- dfs를 하는 과정에서 상하좌우로 전부 방문했거나, 더 이상 이동이 불가능한 노드가 생기게 됩니다.
- 해당 노드를 루트로 하는 서브트리의 크기를 구하면 부모노드에서 얼마만큼의 말이 자식노드로 이동했는지를 구할 수 있습니다.
- 이를 반복하면서 반환된 결과들을 순서대로 저장한 다음에 결과를 거꾸로 출력하면 문제의 답이 됩니다.

E. 배틀쉽



- NM 개의 칸 모두에서 정확히 1 번씩 dfs가 호출되고, 각 칸에서 이동에 $\mathcal{O}(N+M)$ 이 걸립니다.
- 따라서 $\mathcal{O}\left(NM\left(N+M\right)\right)$ 의 시간복잡도에 문제를 해결할 수 있습니다.



F. 휴가 계획

`mst`, `lca`
출제진 의도 – **Hard**

- ✓ 제출 14번, 정답자 3명 (정답률 21.43%)
- ✓ 처음 푼 사람: **18생들**, 60분
- ✓ 출제자: `serin`

F. 휴가 계획



- ✓ 먼저 최단 시간이 걸리는 경로는 유일합니다.
- ✓ 또한 그 경로는 모두 minimum spanning tree 상에 존재함을 쉽게 보일 수 있습니다.
  - 크루스칼 알고리즘을 생각해 봅시다.
  - 만약 i 번째 간선이 잇는 u_i, v_i 사이를 $1, \cdots, i-1$ 번째 간선만 사용하여 이동할 수 있다면, 이동시간이 2^i 분보다 작으므로 i 번째 간선은 사용할 필요가 없습니다.
  - 따라서 크루스칼 알고리즘의 정당성이 확보됩니다.
- ✓ minimum spanning tree 역시 유일합니다.

F. 휴가 계획



- tree를 구성했다면 거치는 정점의 개수는 lca로 쉽게 구해줄 수 있습니다.
- mst를 구성하는데 $\mathcal{O}((N+M)\alpha(N))$, lca를 구성하는데 $\mathcal{O}(N \log N)$, 쿼리를 수행하는데 $\mathcal{O}(Q \log N)$ 이 걸립니다.
- 따라서 $\mathcal{O}((N+Q) \log N)$ 의 시간복잡도에 문제를 해결할 수 있습니다.



G. 나무와 그림자 hard

`segment_tree`, `lazy_prop`, `offline_query`
출제진 의도 – **Hard**

- 제출 2번, 정답자 1명 (정답률 50.00%)
- 처음 푼 사람: **국대와 아이들**, 115분
- 출제자: `sjimed`, `amel`

G. 나무와 그림자 hard



- 접근 방법에 따라 난이도가 많이 달라질 수 있습니다.
- 먼저 그림자가 나무를 통과한다고 생각하고 모든 그림자를 x 축에 눕혀 봅시다.
- 그림자가 겹치지 않으면 나무 위에 그림자가 지지 않고, 겹치면 나무 위에 그림자가 집니다.
- 햇빛의 기울기가 -1 이기 때문에, 그림자가 겹친 길이만큼 나무 위에 그림자가 지게 됩니다.
- 따라서 x 축 상에서 그림자가 겹친 길이의 합을 구하면 됩니다. 물론 여러 번 겹치면 여러 번 더해야 합니다.

G. 나무와 그림자 hard



- 겹치는 길이를 구하는 것은 어렵기 때문에, 각 나무의 길이의 합에서 전체 그림자의 길이를 빼줍시다.
- x 축의 범위가 -10^9 부터 10^9 로 매우 길기 때문에, offline query로 좌표압축을 해줍시다.
- 나무가 추가되고 삭제되는 것은 lazy propagation으로 구현할 수 있습니다.
  - 일반적인 lazy_prop과 달리 존재하는지 안하는지를 prop해줄 수 있습니다.
- 각 시행 후 segment tree의 루트 노드의 값을 조회하여 전체 그림자의 길이를 구할 수 있습니다.
- 따라서 $\mathcal{O}(N \log N)$ 의 시간복잡도에 문제를 해결할 수 있습니다.