제2회 대전과학고등학교 프로그래밍 경진대회 DSHStack 풀이

Official Solution

by

제2회 DSHStack 출제진 및 검수진 with ♥

2025년 11월 7일



출제진

- ✓ koi312500
- ✓ c8216
- ✓ potterlovejh
- ✓ lime7486
- ✓ quadratech188
- ✓ ohwphil
- ✓ wider93



검수진

- ✓ gmroh06
- ✓ jms020820
- ✓ kckyoung2
- ✓ lycoris1600
- ✓ realpsdoingdamyoo
- ✓ sjhi00
- ✓ tlsdydaud1
- ✓ utilforever

예비 소집

문제		의도한 난이도	출제자
PA	라면 끓여 먹자 야호	**Easy**	koi312500
PB	A Graph of Fire and Ice (Easy)	**Medium**	c8216, gmroh06
PC	Paint It Anything Other Than White	**Hard**	quadratech188
PD	A Graph of Fire and Ice (Hard)	**Hard**	c8216, gmroh06

본 대회 / Open Contest

문제		의도한 난이도	출제자
A	대전과학고등학교를 사랑하십니까?	**Easy**	koi312500
B	아 마이마이 하고 싶다	**Easy**	koi312500
C	마법소녀 코이	**Easy**	koi312500
D	슈퍼 학생	**Easy**	c8216
E	이름 짓기	**Medium**	lime7486
F	아직은 어색해	**Medium**	wider93

본 대회 / Open Contest

문제		의도한 난이도	출제자
G	홍수	**Medium**	ohwphil
H	Between	**Medium**	quadratech
I	구간 NOT 과 단일 NOT	**Medium**	c8216
J	물리를 잘하는 시시포스는 오늘도 우울	**Medium**	lime7486
K	독서실 자리 바꾸기	**Hard**	c8216
L	볼록껍질과 쿼리	**Hard**	potterlovejh

PA. 라면 끓여 먹자 야호

`math`
출제진 의도 – **Easy**
출제자 : `koi312500`

교내 대회 예비 소집

- ✓ 제출 162번, 정답 26팀 (정답률 16.667%)
- ✓ 처음 푼 팀: **나서스스택쌓기**, 1분

PA. 라면 끓여 먹자 야호

- ✓ 0.81 배를 비교해서 계산하면 되는 간단한 문제이다.
- ✓ 하지만, $N \times 0.81$ 을 M 과 비교하면 소수점 연산 이슈로 계산값이 달라지는 문제가 발생한다.
- ✓ 그렇기 때문에 $N \times 81$, $M \times 100$ 을 비교하면 된다.
- ✓ Q. C/C++의 경우 N, M 을 float형이나 long double형으로 처리하면 문제가 발생하지 않는다. 이유가 무엇인지 고민해보자.

PB. A Graph of Fire and Ice (Easy)

`bruteforcing`, `dfs`
출제진 의도 – **Medium**
출제자 : `c8216`, `gmroh06`

교내 대회 예비 소집

- ✓ 제출 16번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: —, —분

PB. A Graph of Fire and Ice (Easy)



- 어떤 간선이 같은 타입을 연결하는 것을 허용할지 모든 간선에 대해 시뮬레이션하자. 이때, $\mathcal{O}(M)$ 이 필요하다.
- 몇 개의 간선을 제거할지 시뮬레이션하자. 가중치가 x 인 간선을 제거하려면 가중치가 x 보다 작은 모든 간선을 제거해야 하므로, 제거할 간선의 개수를 결정하면 어떤 간선들을 제거할지 결정된다. 이때, $\mathcal{O}(M)$ 이 필요하다.
- 제거된 간선을 제외한 나머지 그래프가 연결 그래프이고, 허용된 간선을 제외한 나머지 그래프가 이분 그래프인지 판별하면 된다. $\mathcal{O}(N+M)$ 이 필요하다.
- 모든 경우의 수에 대해 제거할 간선의 최솟값을 구하면 된다. 전체 시간 복잡도는 $\mathcal{O}(M^2(N+M))$ 이다.

PC. Paint It Anything Other Than White

segtree
출제진 의도 – **Hard**
출제자 : quadratech188

교내 대회 예비 소집

- ✓ 제출 3번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: —, —분

PC. Paint It Anything Other Than White



- ✓ ’여러 색을 합성하면 흰색이 아니다’라는 명제는 ’여러 색에 (R, G) 성분만 들어있거나, (G, B) 성분만 들어있거나, (B, R) 성분만 들어있다’라는 명제와 동치이다.
- ✓ 따라서 색에 (R, G) 성분만 들어있는 여부, (G, B) 성분만 들어있는 여부, (B, R) 성분만 들어있는 여부를 배열로 저장한다.
- ✓ 문제가 원하는 최대 연속 구간은 위 3개 배열의 최대 연속 구간 중 최댓값이다.
- ✓ 최대 연속 구간은 금광세그로 구할 수 있다. (업데이트 $\mathcal{O}(\log N)$, 쿼리 $\mathcal{O}(\log N)$)
- ✓ 총 $\mathcal{O}(N + K \log N)$ 의 시간복잡도로 해결 가능하다.

PD. A Graph of Fire and Ice (Hard)

`articulation`, `binary_search`, `difference_array`
출제진 의도 – **Hard**
출제자 : `c8216`, `gmroh06`

교내 대회 예비 소집

- ✓ 제출 0번, 정답 0팀 (정답률 -.—%)
- ✓ 처음 푼 팀: —, —분

PD. A Graph of Fire and Ice (Hard)



- ✓ 제거할 간선의 수로 이분 탐색이 가능하다.
- ✓ 이제, $\mathcal{O}(\log M)$을 사용해서 문제를 결정 문제로 바꾸었다. 즉, 하나의 간선을 제외한 나머지 그래프가 이분 그래프인지 판별하는 문제가 되었다.
- ✓ 단, 여기서 제거된 간선을 제외한 나머지 그래프가 연결 그래프가 되도록 탐색 범위를 잘 설정해야 한다.
- ✓ 그래프가 이분 그래프이려면 홀수 사이클이 존재하면 안 된다. 즉, 그래프에서 하나의 간선을 제거해서 모든 홀수 사이클을 없앨 수 있는지 판별하면 된다.
- ✓ 이는 dfs 트리에서 back edge를 이용해 쉽게 판별할 수 있다.
- ✓ dfs 트리에서의 parity를 결정하자. 같은 parity의 정점을 연결하는 back edge를 홀수 back edge, 다른 parity의 정점을 연결하는 back edge를 짝수 back edge라 할 것이다.

PD. A Graph of Fire and Ice (Hard)

- ✓ dfs 트리에 속하지 않는 간선을 허용한다면 홀수 back edge가 하나여야 한다.
- ✓ dfs 트리에 속한 간선을 허용한다고 하자. 어떤 정점 v가 있을 때, v와 v의 부모를 연결하는 간선을 허용할 것이다.
- ✓ v의 서브 트리 내부의 두 정점을 연결하는 홀수 back edge가 존재하면 불가능하다.
- ✓ v의 서브 트리 외부의 두 정점을 연결하는 홀수 back edge가 존재하면 불가능하다.
- ✓ 모든 홀수 back edge가 v의 서브 트리 내부의 정점과 외부의 정점을 연결해야 가능하다.
- ✓ 단, v의 서브 트리 내부의 정점과 외부의 정점을 연결하는 짝수 back edge가 존재한다면 홀수 back edge와 홀수 사이클을 이루므로 불가능하다.

PD. A Graph of Fire and Ice (Hard)



- ✓ 짝수 back edge는 단절선 알고리즘을 이용해 확인할 수 있다. 짝수 back edge에 대해서만 업데이트하면 간선이 짝수 back edge에 대해 단절선인지 알 수 있다.
- ✓ 홀수 back edge는 차분 배열을 이용해 확인할 수 있다. 전체 홀수 back edge의 개수와 간선을 가로지르는 홀수 back edge의 개수와 비교하면 모든 홀수 back edge가 지나는 간선인지 판별할 수 있다.
- ✓ 전체 시간 복잡도는 $\mathcal{O}((N+M)\log M)$ 이다.

A. 대전과학고등학교를 사랑하십니까?

`implementation`
출제진 의도 – **Easy**
출제자 : `koi312500`

교내 대회

- ✓ 제출 49번, 정답 29팀 (정답률 67.347%)
- ✓ 처음 푼 팀: **나서스스택쌓기**, 1분

Open Contest

- ✓ 제출 153번, 정답 124명 (정답률 81.046%)
- ✓ 처음 푼 사람: **seawon0808**, 1분

A. 대전과학고등학교를 사랑하십니까

- ✓ 질문에 잘 대답하면 된다.
- ✓ 대전과학고등학교를 사랑해보자.

B. 아 마이마이 하고 싶다



`sorting`
출제진 의도 – **Easy**
출제자 : `koi312500`

교내 대회

- ✓ 제출 87번, 정답 27팀 (정답률 31.034%)
- ✓ 처음 푼 팀: **북**, 4분

Open Contest

- ✓ 제출 173번, 정답 109명 (정답률 64.740%)
- ✓ 처음 푼 사람: **seawon0808**, 2분

B. 아 마이마이 하고 싶다



- 도착하는 사람은 지문에 따라 가장 빠른 기체를 선택하게 된다.
- 가장 빠른 기체를 선택하기 위해서는, 모든 기체의 총 대기 시간을 확인하면 된다. 이는 $\mathcal{O}(M)$ 의 시간을 필요로 하다.
- 이 과정을 도착하는 사람마다 반복하면, 모든 사람에 대해서 확인해야한다. 이는 $\mathcal{O}(N)$ 이므로 결과적으로 $\mathcal{O}(NM)$ 의 시간을 필요로 한다.
- $1 \leq M, N \leq 1\,000$ 이므로 $\mathcal{O}(NM)$ 은 주어진 시간 안에 돌아간다.
- 마이마이는 재미있는 리듬게임이라고 생각합니다. 한번 기회가 되신다면 해보시는건 어떠신가요? :)

C. 마법소녀 코이

`ad_hoc, constructive`
출제진 의도 – **Easy**
출제자 : `koi312500`

교내 대회

- ✓ 제출 136번, 정답 14팀 (정답률 10.294%)
- ✓ 처음 푼 팀: **고전파**, 12분

Open Contest

- ✓ 제출 60번, 정답 29명 (정답률 48.333%)
- ✓ 처음 푼 사람: **nflight11**, 3분

C. 마법소녀 코이



- 정화의 방패를 만들 수 있는 단 하나의 조합만 찾으면 된다.
- 그렇기 때문에 각 경우에 대해서 K 를 만들 수 있는 방법을 하나만 찾으면 된다.
- 특정한 자연수 N 을 둬보자.
- 이때, 만약 K 가 $2N+1$ 라면 $(N+3)-(N+2)-2$ 이라는 마법 공식을 통해 K 에너지를 가진 정화의 방패를 만들 수 있다.
- 반대로, 만약 K 가 $2N$ 이면 $(N+5)-(N+4)-3$ 이라는 마법 공식을 통해 K 에너지를 가진 정화의 방패를 만들 수 있다.
- 제곱수를 이용하는 다음 방법처럼, 각 상황에서 K 에너지를 가진 정화의 방패를 만들 방법을 찾으면 된다.

D. 슈퍼 학생

`sorting`, `greedy`
출제진 의도 – **Easy**
출제자 : c8216

교내 대회

- ✓ 제출 59번, 정답 14팀 (정답률 23.729%)
- ✓ 처음 푼 팀: **고전파**, 30분

Open Contest

- ✓ 제출 21번, 정답 13명 (정답률 61.905%)
- ✓ 처음 푼 사람: **leinad2**, 14분

D. 슈퍼 학생



- ✓ 먼저 하루 내에서 이동 거리를 최소화하는 방안을 생각해 보자.
- ✓ 어떤 날에 1층에서 출발해서 여러 층을 오르내릴 때, 제일 높은 층의 번호를 X라 하자. 이때 이동거리를 최소화하는 방법은 1층에서 X층까지 한 번 올라갔다 내려오는 것이다. 따라서 그날 이동 거리의 최솟값은 $2(X-1)$이다.
- ✓ 유연한 수업이 없고 고정된 수업만 있는 상황을 생각해 보자. (즉, $a=0$)
- ✓ i요일의 고정된 수업 중 가장 높은 층의 번호를 G_i라고 하면, 그날 이동 거리의 최솟값은 $2(G_i-1)$이고, 일주일 전체 이동거리의 최솟값은 $\sum_{i=1}^{w} 2(G_i-1)$이다.

D. 슈퍼 학생

- 고정된 수업이 없고 유연한 수업만 있는 상황을 생각해 보자. (즉, $b = 0$)
- 먼저 유연한 수업들의 층 번호를 내림차순으로 정렬하여 $F'_1, F'_2, \cdots, F'_a$ 로 재배치한다고 하자.
- 이때, 하루에 최대 M 개의 수업을 들을 수 있으므로, 유연한 수업을 층 번호가 높은 것부터 M 개씩 묶는 것이 최선이다. 즉, 첫날은 $F'_1, F'_2, \cdots, F'_M$, 둘째 날은 $F'_{M+1}, \cdots$ 이런식으로 배정한다.
- 각 날의 최소 이동 거리는 가장 큰 층 번호만 영향을 주므로, 최소 전체 이동 거리는 $\sum_{i=1}^{w} 2(F'_{M\times(i-1)+1} - 1)$ 이다. ($i > a$ 인 i에 대해서는, $F'_i = 1$ 라고 하자.)
- 이는 비둘기집의 원리를 이용해 증명할 수 있다. 하루에 최대 M개의 수업만 넣을 수 있으므로, 가장 높은 층들을 하루에 모아야 최댓값의 개수를 최소로 줄일 수 있다.

D. 슈퍼 학생

- ✓ 이제 고정된 수업과 유연한 수업이 모두 있는 상황을 생각해 보자.
- ✓ 하루에 고정된 수업이 b개가 있으므로, 하루에 넣을 수 있는 유연한 수업은 $M - b$개다.
- ✓ 모든 요일에 대해 i요일의 고정된 수업 중 가장 높은 층의 번호인 G_i를 구하고, 이를 내림차순으로 정렬하여 $G'_1, G'_2, \cdots, G'_w$로 재배치한다고 하자.
- ✓ 유연한 수업도 내림차순으로 정렬하여 $F'_1, F'_2, \cdots, F'_a$로 재배치하고 ($i > a$인 i에 대해서는, $F'_i = 1$), 추가적으로 $F''_i = F'_{(M-b)\times(i-1)+1}$라고 하자.
- ✓ 재배열 부등식에 의해, $x_1, x_2, \cdots, x_w$가 $F''_1, F''_2, \cdots, F''_w$를 적당히 재배열한 것이라고 할 때, $\sum_{i=1}^{w} 2(\max(x_i, G'_i) - 1)$을 최소화하는 방법은 $x_i = F''_i$로 설정하는 것임을 알 수 있다.
- ✓ 따라서, 문제의 정답은 $\sum_{i=1}^{w} 2(\max(F''_i, G'_i) - 1)$ 이다.

E. 이름 짓기

ad_hoc
출제진 의도 – **Medium**
출제자 : lime7486

교내 대회

- ✓ 제출 124번, 정답 9팀 (정답률 7.258%)
- ✓ 처음 푼 팀: **고전파**, 20분

Open Contest

- ✓ 제출 124번, 정답 9명 (정답률 7.258%)
- ✓ 처음 푼 사람: **leinad2**, 18분

E. 이름 짓기

- 간단한 **DP 문제**이다.
- 크기가 $N \times 26$인 배열을 만든다. (모두 0으로 초기화)
- n번째 행은 **글자의 수가 n자인 이름의 수**를 의미한다.
- i번째 열은 **i번째 알파벳으로 끝나는 경우**를 나타낸다. (예: a: 1번째, b: 2번째)
- 예를 들어, 허용된 문자열 중 'ab' 가 있다면 $dp[n][0]$에 $dp[n-1][1]$을 더해준다.
- 모든 항의 합을 구하면, 만들 수 있는 이름의 총 개수가 된다.

F. 아직은 어색해

implementation
출제진 의도 – **Medium**
출제자 : wider93

교내 대회

- ✓ 제출 32번, 정답 3팀 (정답률 9.375%)
- ✓ 처음 푼 팀: **순정**, 58분

Open Contest

- ✓ 제출 31번, 정답 13명 (정답률 41.935%)
- ✓ 처음 푼 사람: **makon**, 21분

F. 아직은 어색해



- 길이 M 의 배열 min_dist를 준비한다. 우리의 목표는, 배열의 i 번째 원소는 사람이 현재 앉아 있는 모든 자리와, i 번째 자리 사이의 거리의 최솟값을 담게 하는 것이다.
- 먼저 min_dist의 i 번째 원소를 1 번 자리와 i 번 자리 사이의 거리로 초기화한다.
- 현재까지의 min_dist 값을 이용하여 현재 학생이 앉을 자리를 결정할 수 있다. 그런 다음에, 현재 학생이 앉은 자리와의 거리를 이용하여 min_dist 배열의 모든 원소를 $\mathcal{O}(M)$ 시간 안에 갱신할 수 있다.
- 이러한 갱신은 $\mathcal{O}(N)$ 번 이루어지기 때문에, 시간복잡도는 $\mathcal{O}(NM)$ 이다.
- 매 학생마다, 자신보다 이전에 착석한 모든 학생의 위치를 기반으로 min_dist를 재계산하는 풀이의 시간복잡도는 $\mathcal{O}(N^2M)$ 로, 시간 초과를 받는다.

G. 홍수

graphs, graph_traversal, ad_hoc
출제진 의도 – **Medium**
출제자 : ohwphil

교내 대회

- ✓ 제출 35번, 정답 10팀 (정답률 28.571%)
- ✓ 처음 푼 팀: **고전파**, 40분

Open Contest

- ✓ 제출 24번, 정답 18명 (정답률 75.000%)
- ✓ 처음 푼 사람: **cinador**, 14분

G. 홍수



- 크게 2가지 방향의 풀이가 있다.
- 방법 1: 그래프 탐색을 활용하는 풀이
- 어떤 하수구가 물이 고이지 않는 하수구라면, 이 하수구와 인접한 하수구 중에서 높이가 현재 하수구보다 높은 하수구 역시 물이 고이지 않는 하수구다.
- 따라서 그래프 탐색을 이용하여, 막히지 않은 하수구에서 출발해서 물이 고이지 않는 하수구를 모두 찾을 수 있다.
- 시간복잡도는 $\mathcal{O}(N + M)$ 이다.

G. 홍수

- 방법 2: 그래프 탐색을 활용하지 않는 풀이
- 길이 N 인 check 배열을 만들고, false로 초기화한다.
- 높이가 낮은 하수구부터 보면서, 만약 이 하수구가 막히지 않은 하수구가 아니면서 check 값이 false라면 이 하수구는 물이 고이는 하수구다. 따라서 "flood"를 출력하고 종료한다.
- 만약 그렇지 않다면, 이 하수구와 인접한 하수구 중 더 높은 하수구의 check 값을 모두 true로 설정해 준다. 마지막 하수구까지 확인했는데도 "flood"를 출력하지 않았다면, "no flood"를 출력한다.
- 어떤 하수구의 check 값이 true라는 것은, 문제에서의 조건 2.에 의해서 물이 빠질 수 있다는 것을 의미하기 때문에 이 알고리즘은 올바르게 작동한다.
- 시간복잡도는 역시 $\mathcal{O}(N+M)$ 이다.

H. Between

`graphs`, `bfs`, `dfs`, `ad_hoc`, `shortest_path`
출제진 의도 – **Medium**
출제자 : `quadratech188`

교내 대회

- ✓ 제출 16번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: —, —분

Open Contest

- ✓ 제출 17번, 정답 8명 (정답률 47.059%)
- ✓ 처음 푼 사람: **leinad2**, 31분

H. Between

- ✓ 모든 점으로부터 BFS/DFS를 시작해 임의의 두 점 사이의 최단 경로 길이를 행렬로 저장한다. $\mathcal{O}(NM)$
- ✓ 삼각부등식: $A, X_1, \cdots, X_k, B$가 최단경로인 것은 $\overline{AX_1} + \overline{X_1X_2} + \cdots + \overline{X_kB} = \overline{AB}$인 것과 동치이다. ($\mathcal{O}(k)$)
- ✓ 다만, 문제에서 주어진 p_k를 배열해 경로를 만드는 방법은 $k!$개, 너무 느리다.
- ✓ 만약 어떤 경로가 최단경로라면, 시작점으로부터의 거리는 계속 증가해야 한다. 따라서 시작점으로부터의 거리 기준으로 오름차순 정렬한 경우에 대해서만 확인해주면 된다.
- ✓ 쿼리당 $\mathcal{O}(k \log k)$, 총 $\mathcal{O}(NM) + \sum_{i=1}^{Q} \mathcal{O}(k \log k) \leq \mathcal{O}(NM + K \log K)$

I. 구간 NOT 과 단일 NOT

dp
출제진 의도 – **Medium**
출제자 : c8216

교내 대회

- ✓ 제출 10번, 정답 1팀 (정답률 10.000%)
- ✓ 처음 푼 팀: **고전파**, 64분

Open Contest

- ✓ 제출 16번, 정답 6명 (정답률 37.500%)
- ✓ 처음 푼 사람: **ruykun**, 30분

I. 구간 NOT 과 단일 NOT



- ✓ 풀이를 시작하기에 앞서, 문제에서 제시한 연산 중 "한 문자열의 1 번째 문자부터 a 번째 문자까지를 반전하는 연산"은 **구간 NOT** 연산, "두 문자열의 b 번째 문자를 동시에 반전하는 연산"은 **단일 NOT** 연산이라고 지칭하도록 하자.
- ✓ 먼저, 두 연산(구간 NOT 연산, 단일 NOT 연산)이 모두 "비트를 뒤집는 연산"이라는 점을 관찰해 보자.
- ✓ 어떤 위치에서든, 그 자리에 연산이 총 몇 번 적용되었는지는 중요하지 않고, 홀수 번 적용되었는지, 짝수 번 적용되었는지만 중요하다. 따라서 적용된 연산의 순서를 어떻게 바꾸어도 최종 결과와 총 비용은 변하지 않는다.
- ✓ 따라서 구간 NOT 연산은 적용 순서가 "뒤집는 마지막 위치 a 가 단조 감소하는 순서"가 되도록 배치하고, 단일 NOT 연산은 "뒤집는 위치 b 가 단조 감소하는 순서"를 유지하면서 구간 NOT 연산 중간 적절한 위치에 배치되도록 연산 순서를 고정한 상황에서 최소 비용을 찾아도 된다.



- ✓ 이 규칙에 따라 연산 순서를 고정한 경우, 어떤 시점에 "지금 처리 중인 구간 NOT 연산의 뒤집는 마지막 위치가 A"라면, 두 문자열의 $A+1$~N 번 문자는 더 이상 구간 NOT 연산으로는 뒤집을 수 없고, 오직 단일 NOT 연산으로만 뒤집을 수 있다.
- ✓ 이제 오른쪽에서 왼쪽으로 내려오면서, "각 문자열에서 지금 위치에 구간 NOT 연산이 적용된 횟수가 홀수/짝수 번인지"를 상태로 들고 가는 DP 배열 및 DP 점화식을 정의하자.
- ✓ 구체적으로, "$dp[i][j][k]$: 두 문자열 s_1, s_2 의 k~N 번 문자를 모두 1로 만들기 위한 최소 비용 (단, s_1, s_2 의 k 번 문자에 각각 구간 NOT 연산이 적용된 횟수를 2로 나눈 나머지가 각각 i, j)" 으로 정의한다.



- ✓ 연산을 한 번도 적용하지 않은 초기 s_1, s_2 를 각각 $s_{1,initial}, s_{2,initial}$ 이라 하고, s_1, s_2 의 k 번 문자에 각각 구간 NOT 연산이 적용된 횟수를 2 로 나눈 나머지를 각각 i, j 라고 하면 현재 s_1, s_2 에서 $s_1[k] = s_{1,initial}[k] \oplus i, s_2[k] = s_{2,initial}[k] \oplus j$ 다. (단, $\oplus$ 는 비트 XOR 연산)
- ✓ k 번 문자에 적용되는 구간 NOT 연산을 모두 완료한 상태에서, $s_1[k] \neq s_2[k]$ 이면 최종적으로 "모든 문자를 1 로 만드는 것"은 불가능하고, $s_1[k] = s_2[k] = 1$ 이면 k 번 문자에 대한 단일 NOT 연산이 필요없으며, $s_1[k] = s_2[k] = 0$ 이면 k 번 문자에 대한 단일 NOT 연산을 한 번 해야 한다 (동시에 $dp[i][j][k]$ 에 c_2 를 더해야 한다).



- ✓ dp 점화식에서, $dp[i'][j'][k+1]$ 에서 $dp[i][j][k]$ 로 전이하는 상황을 생각하자.
- ✓ 만약 $i' \neq i$ 면 "s_1 에서 1~k 번 문자를 뒤집는 구간 NOT 연산"을 한 번 더 한 것이므로, 전이 시 c_1 을 더한 값을 전이해야 한다. 마찬가지로 $j' \neq j$ 면 전이 시 c_1 을 더한 값을 전이해야 한다. ($i' \neq i, j' \neq j$ 를 모두 만족한다면 전이 시 $2c_1$ 를 더해야 한다.)



- ✓ 지금까지의 논리를 종합하면, dp 점화식은 다음과 같다. (!는 논리 NOT 연산)
- ✓ $dp[A][B][k] =$
  $\min(dp[A][B][k+1], dp[!A][B][k+1]+c_1, dp[A][!B][k+1]+c_1, dp[!A][!B][k+1]+2c_1)+c_2$
- ✓ $dp[!A][!B][k] =$
  $\min(dp[!A][!B][k+1], dp[A][!B][k+1]+c_1, dp[!A][B][k+1]+c_1, dp[A][B][k+1]+2c_1)$
- ✓ (단, $dp[0][0][N+1]=0, dp[0][1][N+1]=dp[1][0][N+1]=dp[1][1][N+1]=INF$)

J. 물리를 잘하는 시시포스는 오늘도 우울



ad_hoc
출제진 의도 – **Medium**
출제자 : lime7486

교내 대회

- ✓ 제출 12번, 정답 2팀 (정답률 16.667%)
- ✓ 처음 푼 팀: **고전파**, 87분

Open Contest

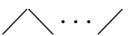
- ✓ 제출 12번, 정답 2명 (정답률 16.667%)
- ✓ 처음 푼 사람: **leinad2**, 55분

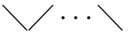
- ✓ 최고점을 하나 선택하고,
  - 왼쪽: 모든 에스컬레이터를 ／ 방향으로 설치
  - 오른쪽: 모든 에스컬레이터를 ＼ 방향으로 설치
- ✓ 그러면 임의의 평지 ↔ 임의의 평지 이동이 가능하다.
  - 임의의 평지 → 최고점 이동 가능
  - 최고점 → 임의의 평지 이동 가능
- ✓ 목표: **가능한 모든 배치 중 이 방법이 항상 최소 길이임을 증명**.

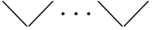
J. 물리를 잘하는 시시포스는 오늘도 우울

- ✓ 높이가 정수인 모든 수평선을 기준으로, 비탈길을 구간 $[h, h+1]$ 단위로 분해한다.
  - 예: 높이 $2 \to 5$ 비탈길 $\Rightarrow [2,3], [3,4], [4,5]$ 세 구간으로 분리
- ✓ 구간 $[h, h+1]$에 포함된 비탈길은 다음 네 패턴 중 하나:

(1) 

(2) 

(3) 

(4) 

- ✓ 비탈길이 0개여도 (4)의 특수한 형태로 본다.

J. 물리를 잘하는 시시포스는 오늘도 우울

- ✓ 아래 세 경우의 평지는 **특정 비탈길에 반드시 에스컬레이터 설치 필요**.
  - 왼쪽: ＼, 오른쪽: ／ → 둘 중 하나는 필수
  - 왼쪽에 ＼ 없음 → 오른쪽 ／ 필수
  - 오른쪽에 ／ 없음 → 왼쪽 ＼ 필수
- ✓ 이를 이용하면 각 패턴별로 최소 설치량을 계산할 수 있다.

- 패턴별 최소 설치 개수:

패턴	필수 설치 개수
(1) k 짝수	$\frac{k}{2}+1$
(2), (3) k 홀수	$\frac{k+1}{2}$
(4) k 짝수	$\frac{k}{2}$

- 처음 제시한 "최고점 기준 좌/우 일방 설치 방법"은 **각 패턴에서 정확히 이 최소량만 설치함**을 확인할 수 있다.
- 따라서 전체 에스컬레이터 길이의 합이 모든 가능한 배치 중 **항상 최소**임이 증명된다.

K. 독서실 자리 바꾸기



segment_tree, lis
출제진 의도 – **Hard**
출제자 : c8216

교내 대회

- ✓ 제출 5번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: —, —분

Open Contest

- ✓ 제출 2번, 정답 1명 (정답률 50.000%)
- ✓ 처음 푼 사람: **leinad2**, 90분

K. 독서실 자리 바꾸기



- ✓ 풀이 1: 위상 정렬의 아이디어와 세그먼트 트리를 이용한 풀이 (처음 출제자가 의도했던 풀이)
- ✓ 풀이에서, 편의상 $i = b_i$ 인 학생의 이동 방향은 시계 방향이라고 할 것이다.
- ✓ 먼저 두 학생의 이동 경로를 보고, "같은 라운드에 둘 수 없는 경우"가 어떤 경우인지 생각해 보자.
- ✓ 두 학생의 이동 방향(시계 방향/반시계 방향)이 같을 때는, 한 학생의 경로가 다른 학생의 경로를 완전히 포함할 때 두 학생이 같은 라운드에 배치될 수 없다.
- ✓ 두 학생의 이동 방향이 다를 때는, 두 학생의 이동 경로가 교차할 때 같은 라운드에 배치될 수 없다.

K. 독서실 자리 바꾸기



- ✓ 각 학생마다 일대일로 대응되는 정점이 존재하는 그래프를 만든다. 처음 좌석이 x번 좌석인 학생을 x번 학생, 그래프에서 x번 학생에 대응되는 정점을 정점 x라 하자.
- ✓ 임의의 i번, j번 학생에 대해 ($i \neq j$),
  - i번, j번 학생이 모두 시계 방향으로 이동하고, i번 학생의 이동 경로가 j번 학생의 이동 경로를 포함하는 경우, 그래프에서 단방향 간선 '$i \to j$'을 추가한다.
  - i번, j번 학생이 모두 반시계 방향으로 이동하고, j번 학생의 이동 경로가 i번 학생의 이동 경로를 포함하는 경우, 그래프에서 단방향 간선 '$i \to j$'을 추가한다.
  - i번 학생이 시계 방향, j번 학생이 반시계 방향으로 이동하고, 두 학생의 이동 경로가 조금이라도 겹치는 경우, 그래프에서 단방향 간선 '$i \to j$'을 추가한다.
- ✓ 이렇게 만들어진 그래프는 방향 비순환 그래프(DAG)다. 또한 임의의 정점 u, v, w에 대해 간선 $u \to v$과 간선 $v \to w$이 존재한다면 반드시 간선 $u \to w$이 존재한다. (즉, 전이성 만족)

K. 독서실 자리 바꾸기



- ✓ 한편, 길이 N 의 배열 round 를 정의하고, $1 \le i \le N$ 인 모든 i 에 대해 $\text{round}[i]$ 를 다음 점화식에 따라 구하자.
  - 정점 i 로 진입하는 간선이 존재하는 정점의 집합 A_i 에 대해,

$$\text{round}[i] = \begin{cases} 1 & (A_i = \varnothing) \\ 1 + \max\limits_{x \in A_i} \text{round}[x] & (A_i \neq \varnothing) \end{cases}$$

- ✓ 그래프가 DAG이므로, 이러한 $\text{round}[i]$ 의 점화식은 유효하다.
- ✓ 이렇게 각 $\text{round}[i]$ 를 구한 뒤, i 번 학생을 $\text{round}[i]$ 번째 라운드에 배정하는 방법은 최소 라운드로 규칙을 준수하는 배정 방법 중 하나다.
- ✓ 이 사실은 다음 방법으로 증명할 수 있다.

K. 독서실 자리 바꾸기



- ✓ 각 정점의 round 값은, 그 정점과 직접적으로 연결된 모든 다른 정점의 round 값과 다르다. 따라서 이러한 배정 방법은 규칙을 준수한다.
- ✓ $\text{round}[i] - 1$ 은 "정점 i에 도달하는 최대 경로 길이"와 같고, 그래프의 전이성에 의해 임의의 경로 안에 있는 각 정점에 대응되는 학생들의 라운드 번호는 모두 달라야 한다.
- ✓ 따라서 그래프를 위상 정렬한 순서에 따라 각 학생의 라운드를 순차적으로 배정할 때 i번 학생의 라운드를 배정하기 위해서는 적어도 $\text{round}[i]$ 개의 라운드가 필요하고, 어떤 순서로 학생들을 배정하든 모든 학생을 배정하기 위해서는 적어도 $\max_{1 \le i \le N} \text{round}[i]$ 개의 라운드가 필요하다.
- ✓ 따라서 이러한 배정 방법은 최소 라운드로 규칙을 준수하는 배정 방법 중 하나다.
- ✓ 그러나 위상 정렬만을 이용해 $\max_{1 \le i \le N} \text{round}[i]$ 을 구하는 방법의 시간복잡도는 $\mathcal{O}(N^2)$ 으로, 시간 초과를 받는다.

K. 독서실 자리 바꾸기



- $\max_{1 \le i \le N} \text{round}[i]$ 을 보다 효율적으로 구하기 위해, 먼저 1 번부터 N 번까지의 원형 좌석 배열을 1 번부터 $2N$ 번까지의 일직선 좌석 배열로 펼친다.
- 다음으로, 이 직선 좌석 배열의 부분 구간 $[S_i, E_i]$ 을 만드는데, S_i 와 E_i 는 i 번 학생의 이동 방향에 따라 다음과 같이 결정된다.

$$S_i = \begin{cases} i & (\text{이동 방향이 시계 방향}) \\ b_i & (\text{이동 방향이 반시계 방향}) \end{cases}, \qquad E_i = \begin{cases} b_i & (\text{이동 방향이 시계 방향}) \\ i & (\text{이동 방향이 반시계 방향}) \end{cases}$$

- 단, $S_i \le E_i$ 면 구간 $[S_i + N, E_i + N]$ 도 만들어 i 번 학생의 이동 경로를 두 개의 선형 구간으로 확장시키고, $S_i > E_i$ 면 $E_i := E_i + N$ 으로 바꾸어 한 개의 선형 구간으로 확장시킨다.

K. 독서실 자리 바꾸기

- ✓ 이렇게 각 학생에 대응되는 원형 좌석 배열에서의 이동 구간을 직선 좌석 배열에서 한 개 또는 두 개의 구간으로 확장해 두고, 각 선형 구간에 대응되도록 round 배열을 만든 뒤 최댓값 세그먼트 트리를 이용해 명시적으로 정점과 간선을 만들지 않더라도 기존 $\mathcal{O}(N^2)$ 풀이에서의 $\text{round}[i]$ 점화식을 동일하게 계산할 수 있다.
- ✓ (선형 구간으로 펼치는 과정에서 어떤 학생의 이동 경로는 두 개의 선형 구간으로 확장되므로, 이 경우 두 확장 구간에 각각 대응되는 round 값을 따로 게산해야 한다. 다만 이 두 구간의 round 값의 최댓값이 $\mathcal{O}(N^2)$ 풀이에서 원래 정점(원형 구간)에 배정되던 $\text{round}[i]$ 와 항상 같다는 사실을 증명할 수 있다.)
- ✓ 구체적으로는 각 구간을 적절한 기준으로 정렬한 뒤 적절한 갱신 규칙에 따라 최댓값 세그먼트 트리에서 값을 갱신하면 되고, 이 풀이의 시간복잡도는 $\mathcal{O}(NlogN)$ 이다.

K. 독서실 자리 바꾸기

- ✓ 풀이 2: LIS를 이용한 풀이
- ✓ 문제 상황을 이분 그래프로 변환하자.
- ✓ 풀이 1 과 동일하게, 1 번부터 N 번까지의 원형 좌석 배열을 1 번부터 $2N$ 번까지의 일직선 좌석 배열로 펼친다.
- ✓ 원형 이동 구간을 선형 구간으로 펼치는 방식 역시 동일하다.
- ✓ 하지만, 이 풀이에서는 일직선 좌석 배열에서 각 좌석에 대응되는 정점을 두 개씩 만들어 두 정점에 서로 다른 색깔을 부여하고, 모든 선형 구간 $[S_i, E_i]$ 에 대해 한 색깔의 정점 중 S_i 번 좌석에 해당하는 정점에서 다른 색깔의 정점 중 E_i 번 좌석에 해당하는 정점으로 향하는 간선을 추가하는 방법으로 이분그래프를 만든다.
- ✓ 이 방법으로 문제를 모델링할 때, BOJ 5058번 (Nested Dolls) 문제와 동일한 방법으로 $\mathcal{O}(NlogN)$ 만에 해결 가능하다. 이 경우 풀이에서 LIS 알고리즘을 이용하게 된다.

L. 볼록껍질과 쿼리



`convex_hull`, `point_in_convex_polygon_check`, `binary_search`

출제진 의도 – **Hard**

출제자 : `potterlovejh`

교내 대회

- ✓ 제출 7번, 정답 0팀 (정답률 0.000%)
- ✓ 처음 푼 팀: —, —분

Open Contest

- ✓ 제출 2번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: —, —분

L. 볼록껍질과 쿼리



- ✓ M 개의 점 중 볼록다각형 내부에 있는 점들은 볼록껍질에 영향을 주지 않는다.
- ✓ 볼록다각형 외부의 한 점에 대하여 볼록다각형에 그은 접선의 접점을 Q_i, Q_j 라고 하자.($i < j$)
- ✓ Q_{i+1} 부터 Q_{j-1} 까지의 점들은 새로운 볼록껍질에 포함되지 않게 된다.
- ✓ M 개의 점들에 대하여 볼록다각형 내부 위치 여부, 볼록다각형에 그은 접선의 접점 번호를 전처리해주면 된다.

L. 볼록껍질과 쿼리



- ✓ 점 Q 가 볼록다각형 내부의 점인지는 $\mathcal{O}(\log N)$ 만에 구할 수 있다.
- ✓ 볼록다각형에서 기준점을 하나 잡으면, 나머지 점들은 반시계 방향 정렬되어있기 때문에 이분탐색으로 Q 가 어느 선분 방향에 있는지 찾을 수 있다.
- ✓ 만약 선분 P_iP_j 방향에 존재한다면, $CCW(P_i, Q, P_j)$ 의 부호로 점 Q 의 내부 여부를 판단할 수 있다.

L. 볼록껍질과 쿼리



- ✓ 볼록다각형을 이루는 점들은 정렬되어있기 때문에, 외부의 점 Q에 대하여 다음 점과의 방향성 $CCW(Q, P_i, P_{i+1})$는 단조성을 갖는다.
- ✓ $CCW(Q, P_i, P_{i+1})$의 부호가 바뀌는 점이 접점이 되고, 이분탐색으로 $\mathcal{O}(\log N)$만에 구할 수 있다.
- ✓ 부호가 바뀌는 점이 두 개이기 때문에 탐색 범위에서 부호가 같은 구간이 두 개가 생기는데, $CCW(Q, P_1, P_i)$로 구분할 수 있다.
- ✓ 접점을 알고 있는 상태에서 각 쿼리는 $\mathcal{O}(1)$만에 처리할 수 있다. 총 시간복잡도는 $\mathcal{O}(M \log N + K)$가 된다.



대회에 참가해주신 모든 분들께 감사합니다!

1. 본 대회 참가 팀 수: 32팀 (57명)
2. Open Contest 참가자 수: 130명