나는코더다 2020 송년대회 본선 풀이

Official Solutions

by

안지민 (retro3014), 이종영 (moonrabbit2), 최은수 (gs18115)

문제		의도한 난이도	출제자
A	거리의 기댓값	**Challenging**	moonrabbit2
B	다오와 디지니의 데이트	**Easy**	gs18115
C	경로 수정하기	**Medium**	moonrabbit2
D	타자 연습	**Easy**	moonrabbit2
E	둥둥섬 다리 재정비하기	**Hard**	moonrabbit2
F	복잡한 쿼리	**Hard**	moonrabbit2
G	쿼리와 수열	**Hard**	moonrabbit2
H	정기 모임 2	**Hard**	moonrabbit2
I	맛집 탐방	**Medium**	gs18115
J	과제 해결하기	**Challenging**	moonrabbit2
K	휴식	**Medium**	gs18115
L	Counting Stars	**Challenging**	moonrabbit2

A. 거리의 기댓값

`dp`

출제진 의도 – **Challenging**

- 제출 5번, 정답 2팀 (정답률 40.00%)
- 처음 푼 팀: **에어팟 감사합니다** (김석표, 정희승, 최승민), 252분
- 출제자: `moonrabbit2`

A. 거리의 기댓값

- $D_{u,v}$ 를 정점 u 와 정점 v 의 거리의 기댓값이라 정의합니다. 다음과 같은 방법으로 $D_{u,v}$ 값들을 계산할 수 잇습니다.
- $$D_{u,v} = \begin{cases} D_{v,u} & (u > v) \\ 0 & (u = v) \\ \displaystyle\sum_{w=1}^{v-1} \frac{a_w(D_{u,w} + c_w + c_v)}{a_1 + \cdots + a_{v-1}} & (u < v) \end{cases}$$
- 마지막의 식의 원리는 T_N 에서의 u 와 v 의 거리는 T_v 에서의 거리와 같으며, T_v 에서 v 는 리프이기에, v 와 연결된 정점 w 에 대해 u 와 v 의 거리는 (u 와 w 의 거리) + (w 와 v 의 거리) 라는 데에 있습니다.
- $\mathcal{O}\left(N^3 + Q\right)$ 에 해결할 수 있습니다.

A. 거리의 기댓값

IAMCODER

- 각 v에 대해 $K_v = \sum_{w=1}^{v-1} \frac{a_w(c_w + c_v)}{a_1 + \cdots + a_{v-1}}$ 는 일정함을 관찰합시다. 이 값들은 $\mathcal{O}(N \log P)$에 모두 계산 가능합니다. 여기서 $P = 10^9 + 7$이며, $\log P$는 역원을 구하는 데에 걸리는 시간입니다. 이 과정에서 부분합 $S_v = a_1 + \cdots + a_v$을 $\mathcal{O}(N)$에, $\frac{1}{S_v}$ 값들을 $\mathcal{O}(N \log P)$에 계산하게 됩니다.
- 이제 $E_{u,v} = \sum_{w=1}^{v} a_w D_{u,w}$를 이용해 DP를 $\mathcal{O}(N^2)$에 구할 수 있습니다. $u < v$일 때 $D_{u,v} = \frac{E_{u,v-1}}{S_{v-1}} + K_v$이며, $E_{u,v} = E_{u,v-1} + a_v D_{u,v}$입니다.

IAMCODER

- $E_{u,v-1}$ 과 $E_{u,v}$ 의 관계를 $D_{u,v}$ 를 풀어 쓰며 알아봅시다.
  $E_{u,v} = E_{u,v-1} + a_v D_{u,v} = E_{u,v-1} + \frac{a_v E_{u,v-1}}{S_{v-1}} + a_v K_v = (1 + \frac{a_v}{S_{v-1}}) E_{u,v-1} + a_v K_v$
  입니다.
- 각 v 마다 일차함수 $F_v(x) = (1 + \frac{a_v}{S_{v-1}})x + a_v K_v$ 를 정의하면, $E_{u,v} = F_v(E_{u,v-1})$ 입니다.
- 즉, 임의의 $u < v$ 인 v 에 대해 $E_{u,v}$ 는 $E_{u,u}$ 에 $F_{u+1}, \cdots, F_v$ 를 순서대로 적용한 값입니다.
  이는 세그먼트 트리를 이용해 $\mathcal{O}(\log N)$ 에 계산하거나, 모든 $F_v(x)$ 의 역이 존재한다는 점을 이용해 prefix의 합성함수들을 미리 계산하는 방식으로 $\mathcal{O}(1)$ 에 계산 가능합니다. 두 풀이 모두 두 일차함수의 합성함수 또한 일차함수임을 이용합니다.

IAMCODER

- 수식을 전개해보면 $E_{u,u} = \frac{2\sum_{k=1}^{u-1} a_k E_{k,k}}{S_{u-1}} + K_u S_{u-1}$ 임을 알 수 있어, 모든 u에 대한 $E_{u,u}$ 값을 $O(N)$에 계산 가능합니다.
- 각 u에 대한 $E_{u,u}$ 값을 알고, $F_{u+1}, \cdots, F_v$의 합성함수에 대입한 결과를 $O(1)$에 구할 수 있으므로 문제를 $\mathcal{O}(N \log P + Q)$에 해결 가능합니다.
- 이 외에도 기댓값의 선형성을 이용하는 풀이, 주어진 구조를 더 관찰해 조금 더 순수 수학적인 측면에서 접근하는 풀이 등이 있습니다.

IAMCODER

B. 다오와 디지니의 데이트

`greedy`
출제진 의도 – **Easy**

- 제출 31번, 정답 12팀 (정답률 45.16%)
- 처음 푼 팀 : **GumiBus882** (송준혁, 박재민, 문홍윤), 8분
- 출제자 : `gs18115`

IAMCODER

- i번 장소와 $i+1$번 장소를 잇는 길에 $w_i = \frac{h_i + h_{i+1}}{2}$의 가중치를 주고, 이를 i번 길이라고 표현합시다.
- i번 길을 지나는 횟수를 a_i라고 하고, 수열 a_i를 경로의 고유 수열이라고 합시다. 그러면 해야 할 일은, $\sum_{i=1}^{N-1} a_i w_i$를 최대로 하는 경로를 구하는 것입니다.

IAMCODER

- 경로의 고유 수열 a_i 가 만족해야 하는 조건을 살펴봅시다.
- i 번 길을 왼쪽 방향으로 지나는 횟수와 오른쪽 방향으로 지나는 횟수는 항상 같습니다. 따라서 a_i 는 항상 짝수임을 알 수 있습니다.
- 어떤 i 에 대해 $a_i > 0$ 이라면, 1번 정점에서 i 번 정점까지 도달할 수 있는 경로를 포함해야 하기 때문에 $1 \leq j < i$ 를 만족하는 모든 j 에 대해 $a_j > 0$ 입니다.
- 각 길을 지나는 것은 항상 1분의 시간을 소모하므로, $\sum_{i=1}^{N-1} a_i \leq T$ 를 만족합니다.

IAMCODER

- 앞의 세 조건을 만족하는 수열 a_i에 대해서, 항상 이를 고유 수열로 가지는 경로가 존재하며, 이는 다음과 같이 만들 수 있습니다.
- 다오와 디지니가 데이트를 시작하기 전, 각 항이 a_i와 동일하게 수열 b_i를 만듭니다.
- 매 분마다, 현재 x번 장소에 위치할 때, b_x가 1이고 b_{x+1}이 0보다 크면 $x+1$번 장소로, 그렇지 않다면 $\max(1, x-1)$번 장소로 갑니다.

B. 다오와 디지니의 데이트

- 최적인 경로의 고유 수열 a_i가 만족하는 조건을 알아봅시다.
- 어떤 i에 대해서, $a_i > 2$와 $a_{i+1} > 0$ 동시에 만족하는 경로가 있다면, 이보다 얻을 수 있는 행복도가 작지 않으면서, 그런 i가 없는 경로가 항상 존재함을 보일 수 있습니다.
- $a_j > 0$인 최대 j를 p라 합시다. 가정에 따라, $p > i$를 만족합니다.
- $1 \leq j \leq p$의 범위에서 a_j를 최대로 하는 j를 q라고 하고, 수열 c_i를 다음과 같이 정의합시다.

$$c_i = \begin{cases} 2, & 1 \leq i < q \\ 2 - 2q + \sum_{i=1}^{N-1} a_i, & i = q \\ 0, & i > q \end{cases}$$

- 수열 c_i 는 고유 수열의 3가지 조건을 모두 만족하기 때문에, 이를 고유 수열로 가지는 경로가 하나 이상 존재합니다.
- c_i 를 고유 수열로 가지는 경로는 a_i 를 고유 수열로 가지는 경로보다 나쁘지 않음을 알 수 있습니다. 따라서 우리는 $a_i > 2$와 $a_{i+1} > 0$을 동시에 만족하는 i가 없는 경로만 고려하여 최적 경로를 찾을 수 있습니다.
- 누적합을 통해 $\mathcal{O}(N)$ 시간에 최적 경로의 조건을 만족하는 모든 경로를 고려할 수 있습니다.

IAMCODER

C. 경로 수정하기

`ad_hoc`

출제진 의도 – **Medium**

- 제출 98번, 정답 11팀 (정답률 12.25%)
- 처음 푼 팀: **국대와 퇴물들** (이민준, 이민규, 임성재), 24분
- 출제자: `moonrabbit2`

IAMCODER

- 당연한 사실은 각 문자의 등장 횟수만이 중요하다는 것입니다. 각각 L_1, R_1, D_1, U_1 로 정의합시다.
- 새로운 문자열 S' 에 대해서도 마찬가지입니다. 새 문자열에서 각 문자의 등장 횟수를 L_2, R_2, D_2, U_2 로 정의합시다.
- S 에서 S' 로 수정하기 위한 최소 수정 횟수는 $N - \min(L_1, L_2) - \min(R_1, R_2) - \min(D_1, D_2) - \min(U_1, U_2)$ 입니다. 바꿀 필요가 없는 문자들을 최대한 유지시키면 됩니다.

C. 경로 수정하기

IAMCODER

- 우선 당연한 경우를 따져줍시다: N의 기우성이 $X+Y$의 기우성이 다르다면 답은 -1입니다. C++은 홀수 음수에 %2 계산을 하면 1이 아닌 -1이 나옴에 유의합시다.
- 그 외에도 $|N| < |X| + |Y|$인 경우 등이 있겠지만, 고려하지 않아도 문제를 해결하는 데에 지장이 없습니다.

C. 경로 수정하기

IAMCODER

- L_2 를 고정하면 R_2, D_2, U_2 가 자동적으로 결정되게 됩니다.
- $R_2 = X + L_2, D_2 = \frac{N - X - Y}{2} - L_2, U_2 = \frac{N - X + Y}{2} - L_2$ 입니다.
- L_2, R_2, D_2, U_2 가 모두 0 이상 N 이하여야 함에 유의합시다. L_2 의 최솟값 $S = \max(0, -X, -\frac{N + X + Y}{2}, -\frac{N + X - Y}{2})$ 이며 최댓값 $E = \min(N, N - X, \frac{N - X - Y}{2}, \frac{N - X + Y}{2})$ 입니다. $[S, E]$ 내의 정수는 모두 L_2 의 후보로 가능합니다. 앞서서 기우성에 대한 처리를 미리 했기 때문에 D_2, U_2 의 값이 유리수가 되는 경우는 없습니다.

C. 경로 수정하기

IAMCODER

- L_2에 대한 수정 횟수는 변곡점들을 기준으로 일차함수들이 이어진 형태입니다.
- 변곡점들과 더불어 S와 E까지만 L_2의 후보로 보아도 최솟값을 구할 수 있습니다. 변곡점들이 $[S, E]$ 밖에 있는 경우 고려하지 않아야 함에 유의합시다.
- 변곡점들은 $L_1, R_1 - X, \frac{N-X-Y}{2} - D_1, \frac{N-X+Y}{2} - U_1$ 입니다.
- 시간복잡도 $\mathcal{O}(N+Q)$에 문제를 해결할 수 있습니다. 이 풀이 외에도 삼분 탐색, 케이스 분석 등의 풀이들이 있습니다.

D. 타자연습

`dp`

출제진 의도 – **Easy**

- 제출 3번, 정답 2팀 (정답률 66.67%)
- 처음 푼 팀: **국대와 퇴물들** (이민준, 이민규, 임성재), 191분
- 출제자: `moonrabbit2`

D. 타자 연습

IAMCODER

- $D_{i_1,\cdots,i_N}$ 을 S_1 의 i_1 번째 문자부터 $|S_1|$ 번째 문자까지, S_N 의 i_N 번째 문자부터 $|S_N|$ 번째 문자까지만 고려했을 때 최소 길이라고 정의합니다.
- 문자열의 첫 글자를 A부터 D까지 시도해보며, 첫 글자와 일치하는 문자열들의 경우 suffix의 길이를 1 줄이는 방법으로 다음 상태를 구할 수 있습니다.
- 사전순 최소인 문자열을 구하는 것은 DP를 역추적하며 A부터 D중 가능한 글자 중 가장 앞선 것을 사용하면 됩니다. 또는 문자열 자체를 DP로 저장해도 됩니다.
- $\mathcal{O}\left(N \times 10^N\right)$ 등에 해결 가능합니다.

IAMCODER

E. 둥둥섬 다리 재정비하기

`tree`, `segment_tree`
출제진 의도 – **Hard**

- 제출 12번, 정답 2팀 (정답률 16.67%)
- 처음 푼 팀: **GumiBus882** (송준혁, 박재민, 문홍윤), 29분
- 출제자: `moonrabbit2`

IAMCODER

- 하나의 섬에 대해서만 쿼리가 주어질 때 $\mathcal{O}(N \log N + Q)$에 해결해 봅시다. 이 섬을 루트로 잡는다면 각 다리에 대해 다리 밑의 서브트리의 크기만큼 루트까지의 경로의 길이의 합에 기여함을 알 수 있습니다.
- 결국 재정비하는 다리들은 기여도가 가장 큰 a개의 다리들입니다. 정렬 후 쿼리당 $\mathcal{O}(1)$에 계산 가능합니다.

E. 둥둥섬 다리 재정비하기

IAMCODER

- 루트에서 자식으로 루트를 이동해 봅시다. 서브트리의 방향성이 바뀌는 도로는 두 섬을 잇는 도로 하나 뿐입니다. dfs 과정에서 섬에 처음 들어갈 때와 나갈 때 해당 간선의 방향성을 바꿔 서브트리의 크기를 바꿔줍시다.
- 결국 현재 섬을 루트로 했을 때의 서브트리 크기들을 관리할 수 있게 됩니다. 세그먼트 트리로 값의 추가/제거 및 k 번째 값까지의 합을 구하는 연산을 구현하여 시간복잡도 $\mathcal{O}\left((N+Q)\log N\right)$ 에 문제를 해결할 수 있습니다.
- 상수가 작은 $\mathcal{O}\left(N\log N + Q\log^2 N\right)$ 등의 시간복잡도를 가진 풀이들도 통과되었습니다.

F. 복잡한 쿼리

`graph`, `linear_algebra`
출제진 의도 – **Hard**

- 제출 1번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: **없음**
- 출제자: `moonrabbit2`

IAMCODER

- 그래프 G의 임의의 스패닝 트리 T에 대해, T에 포함되지 않은 G의 간선 (u, v)와 T에서 u와 v를 잇는 경로를 이용해 단순 사이클들을 만들 수 있으며, 이 단순 사이클들의 XOR로 모든 사이클을 표현할 수 있습니다. 이 원리를 cycle basis라 합니다.
- T에서 u와 v의 경로 위의 간선들의 XOR 값에 cycle basis 상의 사이클들의 값들을 XOR 해줌으로써 모든 가능한 경로의 XOR 값을 표현 가능합니다.
- T에서 루트에서 u까지의 경로의 간선들의 XOR 값 D_u를 정의하면, T에서 u와 v의 경로 위 간선들의 XOR 값은 D_u XOR D_v 입니다.

- XOR 연산을 Z_2^{30} 에서의 덧셈 연산으로 볼 수 있습니다. cycle basis 상의 사이클들의 값들의 집합 C 를 이용해 REF 를 만들고, $L = D_u$ XOR D_v 에 최상위 비트부터 그리디하게 XOR 여부를 결정해 하나의 $d(u, v)$ 를 $\mathcal{O}(30)$ 에 계산 가능합니다.
- 이제 주어진 쿼리들을 답해봅시다. 기존의 REF 로는 상당히 어렵지만, RREF 를 만든다면 각 비트별로 독립적으로 해결하는 데에 큰 도움이 됩니다. leading one에 대해 나머지 벡터들은 0 을 값으로 가지기 때문입니다. 예시로, 이제 $d(u, v)$ 를 계산할 때 굳이 최상위 비트부터 고려할 필요가 없습니다.

F. 복잡한 쿼리

IAMCODER

- 우선, 모든 D_i XOR D_j 의 XOR을 계산합니다. $r-l+1$ 이 짝수이면 D_l XOR $\cdots$ XOR D_r 이고, 홀수이면 0입니다. 이제 추가적으로 RREF 상의 각 값을 몇 번 XOR하는지 계산하면 됩니다.
- 각 비트별로 문제를 해결합니다. RREF 상의 한 값이 답에 몇 번 XOR 되는지 계산하려면 결국 D_i XOR D_j 의 B 번째 비트가 켜져 있는 (i, j) 의 개수를 빠르게 세면 됩니다. D_i 의 B 번째 비트가 켜져 있는 i 의 개수, 꺼져 있는 i 의 개수를 부분합으로 구하면 쿼리를 $\mathcal{O}(30)$ 에 계산 가능합니다.
- 총 시간복잡도 $\mathcal{O}(30(N+M+Q))$ 에 해결 가능합니다.

IAMCODER

- 쿼리의 시간복잡도를 $\mathcal{O}(1)$로 개선할 수 있습니다. 모든 D_i 값들을 미리 RREF 상의 값들과 XOR해 가능한 최소로 만들어준다면, 모든 D_i XOR D_j 의 B번째 비트가 RREF 상의 leading one으로 존재하는 모든 B에 대해 0이기 때문입니다.
- 모든 RREF 상의 값들의 XOR을 마지막에 XOR 할지만 $r - l + 1$을 4로 나눈 나머지에 따라 결정해주면 됩니다.
- 총 시간복잡도 $\mathcal{O}(30(N + M) + Q))$에 해결 가능합니다.

G. 쿼리와 수열

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

- 제출 0번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: **없음**
- 출제자: `moonrabbit2`

IAMCODER

- $D_{l,r}$ 을 $[l, r]$ 범위의 인덱스들의 A_i 값을 채우며, $[l, r]$ 에 완전히 포함된 쿼리들만 고려할 때의 최댓값이라 정의합니다.
- $D_{l,r} = \max_{l \le k \le r} (D_{l,k-1} + D_{k+1,r} + cost(l, r, k))$ 입니다. $cost(l, r, k)$ 는 $[l, r]$ 에 완전히 포함된 쿼리들 중, k 를 지나는 쿼리들의 답은 모두 A_k 라 가정했을 때 최적의 비용입니다.
- 식만 봤을 때는 그렇게 직관적으로 다가오지는 않습니다.

IAMCODER

- 답은 $D_{1,N}$ 이하입니다: 최적의 배열에서 k를 항상 $[l, r]$ 내에서 A_i가 최대인 i로 고를 경우, 답이 됩니다.
- 답은 $D_{1,N}$ 이상입니다: 최댓값을 고르는 것은 구간에서 임의의 값을 고를 수 있을 경우에 최적의 전략입니다. DP 과정이 결국 하나의 위치를 골라 그 위치를 포함하는 쿼리의 결과를 몰아주는 것을 반복하는 것이므로, 답보다 좋은 결과를 낼 수 없습니다.
- 결국 $D_{1,N}$이 우리가 원하는 답이 됩니다.

IAMCODER

- 이제 필요한 것은 $cost(l, r, k)$ 를 빠르게 계산하는 것입니다. $[l, r]$ 에 완전히 포함되며 k 를 지나는 쿼리들의 개수 X 는 부분합을 이용해 계산할 수 있습니다.
- 결국 $V_{k,i}X - C_{k,i}$ 의 최댓값을 구하면 됩니다. 이는 CHT로 $\mathcal{O}(\log K_k)$ 에 가능해, 문제를 $\mathcal{O}\left((N^3 + K)\log K\right)$ 에 해결 가능합니다.
- 전처리로 $cost$ 값들을 구한다면 $\mathcal{O}\left(N^3 + KN + K\log K\right)$ 도 가능합니다. $Q_{i,j}$ 값들이 모두 0 이상임을 이용해 l 과 k 를 고정하고 r 을 1씩 증가시키며 CHT 상의 교점을 이동하면 됩니다.
- 모든 X 값들을 미리 정렬하는 방법으로 $\mathcal{O}\left(N^3 \log N + K\log K\right)$ 도 가능합니다.

H. 정기 모임 2

`tree`

출제진 의도 – **Hard**

- 제출 1번, 정답 1팀 (정답률 100.00%)
- 처음 푼 팀: **국대와 퇴물들** (이민준, 이민규, 임성재), 199분
- 출제자: `moonrabbit2`

H. 정기 모임 2

IAMCODER

- 트리 T 상에서 u와 v의 경로 최댓값이, 배열 A 상에서 u와 v의 위치 기준 구간 최댓값이 되는 길이 $N-1$의 배열을 만들 수 있습니다.
- 간선들을 가중치 순으로 정렬한 후, 첫 간선부터 보면서 현재 간선 e가 연결하는 컴포넌트의 배열 A_1과 A_2에 대해, A_1, e, A_2를 순서대로 이은 배열이 새 컴포넌트의 배열이 되는 것을 반복하면 됩니다.
- 다양한 방법으로 $\mathcal{O}(N \log N)$에 구현 가능합니다.

IAMCODER

- 이 배열 A 상에서 문제를 해결해도 답이 동일함은 자명합니다.
- A 상의 연속한 살아있는 정점들끼리 묶어 X 개의 그룹을 만드는 방법들 중 최적의 방법이 있습니다. 꼬임을 풀어주어 최댓값이 증가하는 경우가 없기 때문입니다.
- 이 문제는 살아있는 연속한 정점들 사이에 $X-1$ 개의 "끊음"을 만들어 준다고 해석해도 좋습니다. 끊는 경우 가장 가중치가 높은 것을 끊는 것이 좋을 것입니다.

H. 정기 모임 2

IAMCODER

- 결국 배열 상에 정점 1부터 i까지만 살아있을 때, 살아있는 연속한 두 정점 사이의 구간 최댓값 $i-1$개 중 X번째로 큰 값이 하나의 X에 대한 답이 됩니다. $i-1 < X$인 경우에는 0입니다.
- 즉 1부터 K까지의 X에 대한 답의 합은 가장 큰 $\min(i-1, K)$개 값의 합이 됩니다.
- 다양한 방법으로 $\mathcal{O}(N \log N)$에 구현 가능합니다. 현재 구간들을 i를 늘리면서/줄이면서 관리하면 됩니다. 자료구조들을 적절히 활용합시다.
- 이 풀이 외에도 트리에서 바로 해결하는 다양한 풀이들이 있습니다.
- K가 고정이 아니어도 E번 문제와 같은 세그먼트 트리를 이용해 해결할 수 있습니다.
- prefix가 아닌 구간 쿼리에 대해서도 Mo를 섞으면 해결할 수 있습니다.

I. 맛집 탐방

`scc, dp`
출제진 의도 – **Medium**

- 제출 68번, 정답 5팀 (정답률 8.82%)
- 처음 푼 팀 : **에어팟 감사합니다** (김석표, 정희승, 최승민), 61분
- 출제자 : `gs18115`

IAMCODER

- 각 맛집을 정점으로 보고 각 길을 간선으로 보아 N 개의 정점과 M 개의 간선을 가지는 유향 그래프를 만들 수 있습니다.
- 각 정점을 SCC로 묶고, 각 SCC에 번호를 1부터 순서대로 매깁시다.
- 한 SCC를 하나의 정점으로 보면, 그래프는 DAG가 되어, 위상정렬을 할 수 있습니다.

- dp_i를 i번 SCC를 끝으로 하는 경로 중, 서로 다른 정점을 가장 많이 포함하는 경로가 포함하는 서로 다른 정점의 수로 정의하면, 위상정렬의 순서대로 각 dp_i의 값을 $\mathcal{O}(N+M)$의 시간복잡도에 구할 수 있습니다.
- i번 SCC에 속하는 정점 개수를 nV_i, 그래프의 간선 집합을 E라 하고, $dp_0 = 0$이라 정의하면, dp_i의 점화식은 다음과 같습니다.

$$dp_i = nV_i + \max_{(j,i)\in E \text{ or } j=0} dp_j$$

- dp_i의 최댓값이 N과 같은지의 여부에 따라서, 모든 맛집을 지나는 경로가 존재하는지의 여부를 구할 수 있습니다.

IAMCODER

- 정점의 경우와 비슷하게, dp_i 를 i 번 SCC를 끝으로 하는 경로 중, 서로 다른 간선을 가장 많이 포함하는 경로가 포함하는 서로 다른 간선의 수로 정의하면, 위상정렬의 순서대로 각 dp_i 의 값을 $\mathcal{O}(N+M)$ 의 시간복잡도에 구할 수 있습니다.
- i 번 SCC에 속하는 간선 개수를 nE_i, 그래프의 간선 집합을 E 라 하고, $dp_0 = -1$ 이라 정의하면, dp_i 의 점화식은 다음과 같습니다.

$$dp_i = 1 + nE_i + \max_{(j,i) \in E \text{ or } j=0} dp_j$$

- 마찬가지로, dp_i 의 최댓값이 M 과 같은지의 여부에 따라서, 모든 기념품 상점을 지나는 경로가 존재하는지의 여부를 구할 수 있습니다.

- 모든 맛집을 지나는 경로가 존재한다는 명제를 P, 모든 기념품 상점을 지나는 경로가 존재한다는 명제를 Q, 모든 맛집과 모든 기념품 상점을 동시에 지나는 경로가 존재한다는 명제를 R이라 합시다.
- SCC를 한 정점으로 하고, 기존 그래프의 간선을 각 정점을 포함하는 SCC 사이의 간선으로 바꿔 만든 그래프를 생각해 봅시다.
- 기존 그래프에서의 경로는, 바뀐 그래프 상에서도 경로로 나타납니다. Q가 참이라면, 바뀐 그래프는 경로 하나와, 그 경로에 포함되지 않은 loop를 포함하여 어떤 간선에도 연결되어 있지 않은 정점 여러 개로 나타납니다.

- 위에서 언급한 "경로에 포함되지 않은 어떤 간선에도 연결되어 있지 않은 정점"이 한 개 이상 존재한다면, 경로에 해당하는 정점과 그 정점은 연결되어 있지 않으므로, P는 거짓입니다.
- R이 참이라면, P와 Q가 참이고, 이에 따라 바뀐 그래프는 경로 하나로 나타납니다.
- 바뀌기 전의 그래프에서의 경로를 생각해 보면, 이 경로는 모든 간선과 정점을 동시에 지납니다. 따라서 바뀐 그래프가 경로 하나로 나타나면 R이 참임을 알 수 있습니다.

- 위의 논리에 따라서, R과 $P \wedge Q$가 동치임을 알 수 있습니다. P와 Q는 이미 참인지 확인했으므로, R이 참인지는 쉽게 구할 수 있습니다.
- 각 과정이 $\mathcal{O}(N+M)$의 시간복잡도를 가지기 때문에, 전체 시간복잡도 역시 $\mathcal{O}(N+M)$입니다.

J. 과제 해결하기

`graph`, `greedy`
출제진 의도 – **Challenging**

- 제출 18번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: **없음**
- 출제자: `moonrabbit2`

- 과제들을 정점으로 하고 두 과제 간에 겹치지 않는다면 간선으로 이은 그래프를 생각해 봅시다. 즉 이 그래프는 구간 그래프의 여 그래프 (complement graph) 입니다.
- 이 그래프에서 최대 매칭을 구했다고 합시다. 최적의 전략은 매칭에 포함된 두 정점들에 같은 학생을 배정하는 것을 반복한 후, 아직 학생이 남아 있다면 임의의 남아있는 과제들을 각각 남아있는 학생에게 배정하는 것입니다.
- 이제 이 그래프에서 최대 매칭을 빠르게 구하는 방법을 알아봅시다,

- 구간 그래프는 채색 수와 최대 크기 클리크의 크기가 동일합니다. 구간 그래프의 클리크는 해당하는 구간들이 같은 점을 공유하는 정점들입니다.
- 그래프를 채색한 것은 결국 몇 개의 독립 집합으로 나누는 것입니다. 채색 수만큼의 독립 집합을 만들었다고 합시다.
- 이렇게 만들었을 때 중요한 사실은 크기 1의 독립 집합에 포함된 정점들은 모든 최대 크기 클리크에 포함되어 있다는 것입니다. 최대 크기 클리크들에선 현재 겹쳐있는 구간의 수와 독립 집합의 수가 같기 때문에 각 독립 집합에서 하나씩의 정점이 포함되어 있어야 하기 때문입니다.

IAMCODER

- 각각 크기 2 이상의 두 개의 독립 집합 S_1, S_2에 대해, S_1 상의 정점 I_1, I_2, S_2 상의 정점 I_3, I_4을 잡읍시다. I_1이 I_2 앞에, I_3가 I_4 앞에 있다고 합시다.
- (I_1, I_4)와 (I_2, I_3) 중 성공적인 매칭이 반드시 존재함을 어렵지 않게 보일 수 있습니다.
- 이 사실은 크기 2 이상의 독립 집합들의 크기 합 S에 대해, 적어도 $\lfloor \frac{S}{2} \rfloor$ 크기의 매칭을 구할 수 있다는 뜻입니다. 짝수 크기 독립 집합들은 독립 집합 내에서 모든 매칭을 구하고, 홀수 크기 집합들 둘을 잡아 새 매칭을 하나 구하고 짝수 크기 독립 집합으로 바꾸는 것을 최대한 반복하면 됩니다.

IAMCODER

- 크기 1의 독립 집합은 모든 최대 크기 클리크에 포함되어 있으므로, 서로 다른 크기 1의 독립 집합 상의 두 정점들 간에는 반드시 매칭이 불가능합니다.
- 즉 크기 1인 독립 집합들 상의 정점들과 크기 2 이상인 독립 집합들 상의 정점들 간에만 서로 간선을 이어 이분 매칭 X 를 구했다면, 적어도 X 개의 정점들은 최대 매칭에 포함되지 않습니다. 즉 최대 매칭의 상한은 $\lfloor \frac{N-X}{2} \rfloor$ 입니다.

J. 과제 해결하기

- 매칭에 포함되지 않았던 X 개의 크기 1인 독립 집합들 상의 정점들은 크기 1의 독립 집합에 포함되게, 나머지 정점들은 크기 2 이상의 독립 집합에 포함되게 할 수 있습니다, 이 방법으로 크기 1의 독립 집합에 포함된 정점의 수를 최소화하면 됩니다.
- 이 사실이 의문스러울 수 있는데, 기존의 독립 집합들에서 매칭에 포함되며 크기 1인 독립 집합에 포함된 정점이 있다면 그 정점에 매칭된 다른 정점을 해당 독립 집합으로 옮기는 것을 반복하면 됩니다. 옮기는 정점은 모든 최대 크기 클리크에 포함되지 않으므로 옮긴 후에도 기존의 독립 집합의 크기가 0이 되는 일이 없으며, 한 정점이 두 정점과 매칭되지는 않으므로 이 과정에서 사이클이 발생하는 경우도 없습니다. 항상 최종적으로 매칭에 포함된 두 정점이 같은 독립 집합에 있게 되는 것은 아님에 유의합시다.
- 앞선 논의에 따라 결국 크기 $\lfloor \frac{N-X}{2} \rfloor$ 의 매칭을 구할 수 있고, 이것이 최적입니다.

- 이제 일반 그래프 최대 매칭 알고리즘의 도움 없이 최대 매칭을 구하는 방법을 알았습니다.
- 두 가지 일이 남았습니다: 이분 매칭을 빠르게 계산하는 것과 이분 매칭을 바탕으로 크기 1의 독립 집합에 포함된 정점의 수를 최소화하기.

IAMCODER

- 이분 매칭을 빠르게 구하기 위해서는 임의의 최대 크기 클리크를 잡아야 합니다. 클리크에 포함된 구간들이 위치 p를 지난다고 합시다.
- p 왼쪽의 구간들의 집합 L, p를 지나는 구간들의 집합 (즉, 최대 크기 클리크) C, p 오른쪽의 구간들의 집합 R을 정의합시다.
- L을 끝점 순으로 정렬하고 R을 시작점 순으로 정렬합시다.
- C의 각 정점은 L의 prefix와 R의 suffix에 매칭이 가능합니다.

IAMCODER

- R 뒤에 L 을 이어붙이거나 L 과 R 을 뒤집고 L 뒤에 R 을 이어붙이면 결국 C 의 각 정점은 이 새로운 집합의 구간에 매칭이 가능합니다.
- 이 집합의 정점들을 가장 앞부터 순회하며 각 정점마다 현재 살아있는 구간들 중 끝점이 가장 앞서는 구간에 그리디하게 매칭을 하는 것이 최적입니다. 시작점 기준 정렬 + 우선순위 큐 등의 방법을 사용하면 됩니다.
- 이분 매칭을 $\mathcal{O}(N \log N)$ 에 구할 수 있게 되었습니다!

IAMCODER

- 이분 매칭을 바탕으로 크기 1의 독립 집합에 포함된 정점의 수를 최소화합시다. 앞서 설명한 방법을 실제로 구현하면 됩니다.
- 크기 1의 독립 집합들을 큐 등으로 관리하고, 독립 집합들을 std::set 등으로 관리하면 $\mathcal{O}(N \log N)$에 가능합니다. 정점의 이동은 매칭에 포함된 정점들 각각에 최대 한 번 일어나기 때문입니다. 결국 문제를 $\mathcal{O}(N \log N)$에 해결할 수 있습니다.

- 이 문제의 풀이는 Frédéric Gardi의 논문 Efficient algorithms for disjoint matchings among intervals and related problems(2003) 을 바탕으로 일부 부분을 수정한 것입니다.
- 검색이 허용되는 대회였지만 문제가 해결되는 것을 예상하진 않았고, 그리디 풀이들로 유혹하기 위한 문제였습니다 :)

K. 휴식

`game`, `ad_hoc`
출제진 의도 – **Medium**

- 제출 138번, 정답 7팀 (정답률 5.07%)
- 처음 푼 팀: **국대와 퇴물들** (이민준, 이민규, 임성재), 84분
- 출제자: `gs18115`

- 후공인 민준이는, 매 턴 S의 변화로 1과 2 두 개의 값 중 하나를 선택할 수 있습니다. 따라서, $X + N \leq p \leq X + 2N$을 만족하는 M의 배수 p가 있으면, 승민이가 얻는 점수는 0점입니다.
- 그런 p가 없는 경우는 $X + 2N + 1$이 M의 배수인지 여부에 따라서 두 가지의 경우로 나눌 수 있습니다.

IAMCODER

- $X + 2N + 1$이 M의 배수일 경우, $X + 2N + 1 = M$이거나, $X + 2N + 1 = 2M$이 됩니다. 각 경우 승민이가 얻는 점수는 $X + N - 1$, $X + N - 1 - M$이 되며, 승민이와 민준이가 택할 수 있는 전략 중 하나는 다음과 같습니다.
- 승민이의 전략은, 첫 턴에 0을 올리고, 다음 턴부터는 민준이가 0을 올렸다면 1을, 1 또는 2를 올렸다면 0을 올리는 것입니다. 이 경우 S가 $X + N - 1$ 이상 $X + 2N$ 이하의 수가 되기 때문에, 승민이는 목표 점수 이상의 점수를 획득할 수 있습니다.
- 민준이의 전략은, 승민이의 전략과 달리 턴 전체를 기준으로 구성됩니다. 승민이가 첫 턴에 1을 올렸다면 모든 턴에 2를 올리고, 그렇지 않다면 첫 턴에 0을 올리고 나머지 턴에 1을 올리는 것입니다. 두 경우 모두 승민이의 점수를 목표 점수 이하로 만들 수 있습니다.

IAMCODER

- $X + 2N + 1$이 M의 배수가 아닐 경우, 승민이의 전략이 약간 달라집니다. 승민이는 첫 턴에 0 대신 1을 올려도 상관이 없기 때문에, 승민이는 S를 $X + N$ 이상으로 만들어, $X + N \mod M$ 이상의 점수를 얻을 수 있습니다.
- 민준이의 전략 역시 약간 달라집니다. 민준이는 모든 턴에 1을 올릴 수 있고, 승민이가 정확히 $X + N \mod M$의 점수를 얻도록 할 수 있습니다. 따라서 이 경우 승민이는 $X + N \mod M$의 점수를 얻게 됩니다.

IAMCODER

- 간단한 Casework이기 때문에 각 게임 당 시간복잡도는 $\mathcal{O}(1)$ 입니다.
- 이분탐색을 사용하는 풀이도 존재하며, 각 게임 당 $\mathcal{O}(\log M)$ 의 시간복잡도를 가집니다.

L. Counting Stars

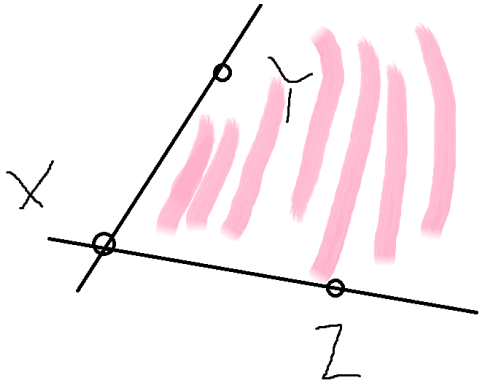
`geometry`
출제진 의도 – **Challenging**

- 제출 1번, 정답 0팀 (정답률 0.00%)
- 처음 푼 팀: **없음**
- 출제자: `moonrabbit2`

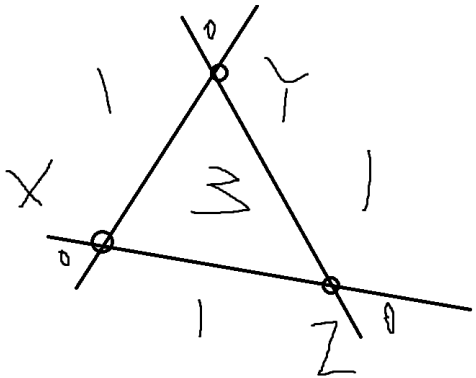
IAMCODER

- (X, Y, Z) 별자리에 대해, 바로 답을 구하는 것은 어렵지만 상대적으로 구하기 쉬운 여러 값들에서 답을 도출해 낼 수 있습니다.

L. Counting Stars

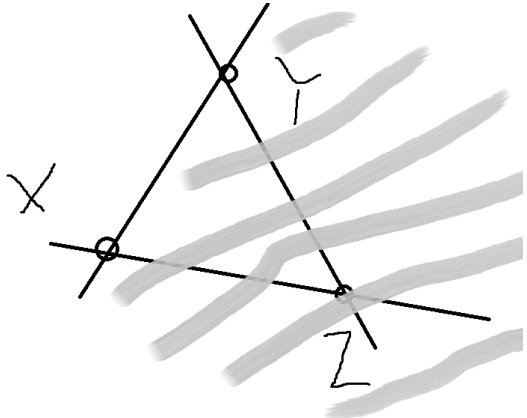


- X 를 중심으로 Y 와 Z 의 각도 범위 내의 별들의 c_i 값의 합을 구해봅시다. X, Y, Z 는 고려 대상에서 제외하는 것이 편합니다.

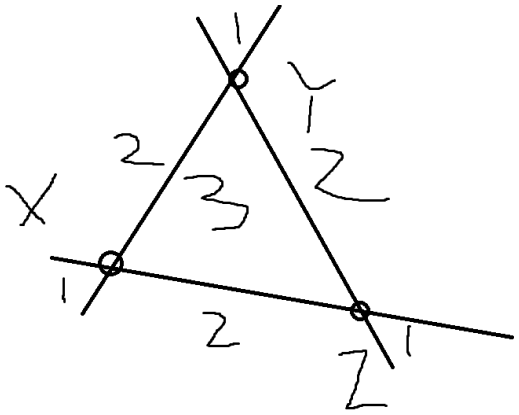


- Y 와 Z 를 중심으로 비슷한 값들을 계산한 후 더하면 각 영역별로 더해지는 횟수는 다음과 같습니다. 이 값을 S_1 이라고 정의합시다.

L. Counting Stars

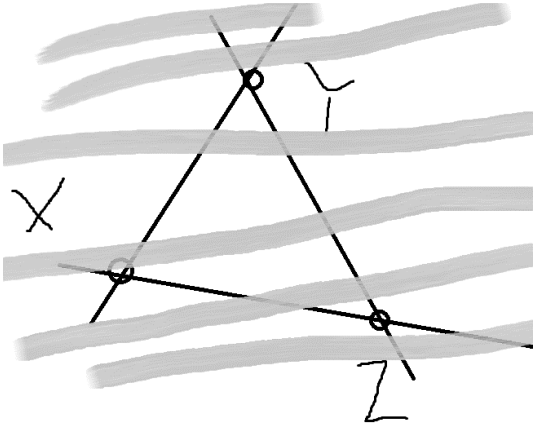


- 이번엔 X 와 Y 를 잇는 직선의 삼각형 방향 쪽 별들의 c_i 값들의 합을 구해봅시다. 마찬가지로 X, Y, Z 는 고려 대상에서 제외합니다.



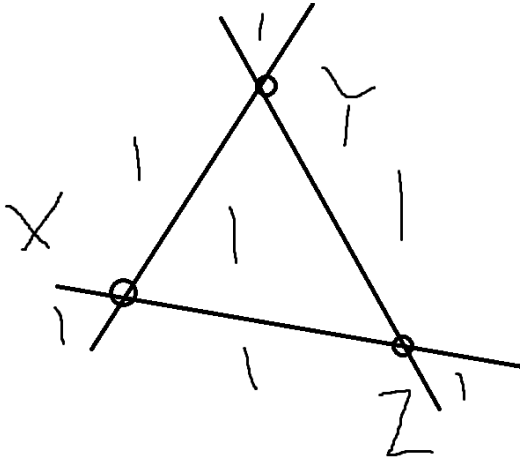
- 다른 직선들을 기준으로 비슷한 값들을 계산한 후 더하면 각 영역별로 더해지는 횟수는 다음과 같습니다. 이 값을 S_2 라고 정의합시다.

L. Counting Stars

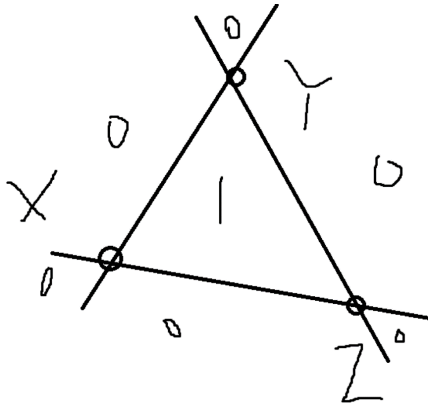


- 마지막으로 X, Y, Z 를 제외한 현재 모든 별들의 c_i 값들의 합을 구해봅시다.

L. Counting Stars



- 각 영역별로 더해지는 횟수는 다음과 같습니다. 이 값을 S_3 라고 정의합시다.



- $S_1 - S_2 + S_3$ 에서 각 영역별로 더해지는 횟수는 다음과 같습니다. 놀랍게도 원하는 영역 내의 별들만 세지는 것을 알 수 있습니다.

IAMCODER

- 결국 $S_1 - S_2 + S_3$ 에 $c_X + c_Y + c_Z$ 를 추가로 더하면 원하는 답이 됩니다. 또는 S_3 에서는 X, Y, Z 를 제외하지 않는 것도 방법입니다.
- S_3 을 구하는 것은 현재까지 별들의 c_i 값들의 합을 저장하면서 진행하면 굉장히 쉽습니다.

IAMCODER

- S_1 과 S_2 는 오프라인으로 계산합니다. 각 별을 중심으로 다른 별들을 각도순 정렬한 후 스위핑하는 방식으로 모든 값들을 계산할 수 있습니다.
- 별자리를 관측할 때 현재 기준 존재하는 별들만 고려하기 때문에, 시간 prefix에 대한 쿼리가 필요해 펜윅 트리 등이 필요합니다.
- 시간복잡도 $\mathcal{O}\left((N^2+Q)\log N\right)$ 에 해결할 수 있습니다.