나는코더다 2022 입부 시험 풀이

Official Solutions

by

나는코더다 39기

2022년 3월 21일

문제		의도한 난이도	출제자
A	카드 색칠	**Easy**	박영우
B	개표	**Easy**	박영우
C	Split the SSHS	**Medium**	문정후
D	센터가 돋보여야 해	**Medium**	이동현
E	용암 점프	**Hard**	문정후
F	지역 순회	**Hard**	박종경
G	균형 발전	**Challenging**	박영우

A. 카드 색칠

ad-hoc

출제 의도 – **Easy**

- 제출 176번, 정답 31명 (정답률 32%)
- 처음 푼 사람: 김건표, 4분
- 참가자 평균 점수: ???점
- 출제자: 박영우

A. 카드 색칠

문제 요약

- 일렬로 놓인 N 장의 카드가 있습니다.
- 어떤 것은 빨강, 초록, 파랑 중 하나의 색으로 색칠되어 있고, 아무 색으로도 색칠되어 있지 않은 카드도 있습니다.
- 색칠되지 않은 모든 카드에 대해 빨강, 초록, 파랑 중 하나의 색으로 색칠을 하여 모든 연속한 두 카드가 다른 색이 되도록 해야 합니다.

A. 카드 색칠

부분 문제 1 (29점)

- 초기에 아무 카드도 색칠되어 있지 않습니다.
- 짝수 번째 칸에는 빨간색, 홀수 번째 칸에는 파란색을 색칠하면 됩니다.
- 색칠이 불가능한 경우는 없습니다.

A. 카드 색칠

부분 문제 2 (54점)

- 초기에 빨간색 카드만 색칠되어 있습니다.
- 색칠이 불가능한 경우는 초기에 빨간색이었던 카드 중에서 인접한 카드가 있는 경우입니다.
- 초기에 빨간색인 카드가 인접하지 않는다면 항상 가능합니다.
- 홀수 번째 카드 중 색칠되지 않은 카드는 파란색, 짝수 번째 카드 중 색칠되지 않은 카드는 초록색으로 색칠하면 됩니다.

A. 카드 색칠

만점 풀이 (100점)

- 불가능한 경우는 초기에 색깔이 같은 카드가 인접한 경우입니다.
- 카드를 앞에서부터 보면서 색칠되지 않은 카드가 있다면 알맞은 색으로 색칠합니다.
- 사용 가능한 색이 총 세 개이고 인접한 색칠된 카드로 인해 사용이 불가능한 색은 최대 두 개이므로, 각각의 카드를 색칠할 때 반드시 사용이 가능한 색이 있습니다.

B. 개표

`ad-hoc`, `math`

출제 의도 – **Easy**

- 제출 328번, 정답 18명 (정답률 18%)
- 처음 푼 사람: 김건표, 13분
- 참가자 평균 점수: ???점
- 출제자: 박영우

B. 개표

문제 요약

- ✓ $N+1$ 명의 후보가 있는 투표에서 개표하고 있고, 정후는 $N+1$ 번 후보입니다. 다음 두 가지 연산을 처리해야 합니다.

1. x 번 후보의 표가 y 표 더 집계됩니다.
2. 정후에게 x 표, 정후가 아닌 후보들에게 y 표가 추가로 더 집계될 때 정후가 당선이 될 가능성이 있는지의 여부를 구합니다.

B. 개표

부분 문제 1 (10점)

- 정후 외의 후보는 한 명입니다.
- 따라서, 정후의 질문이 들어올 때마다 정후가 가정한 각 후보의 표 수를 정확하게 알 수 있습니다.

B. 개표

부분 문제 2 (79점)

- A_i 를 i 번 후보의 표 수로 정의합니다.
- 우선 정후가 x 표를 받아도 정후의 표 수 보다 같거나 많은 표를 받은 후보가 있다면, 즉 $A_i \geq A_{N+1} + x$ 인 $i\,(1 \leq i \leq N)$ 가 존재한다면 답은 “NO”입니다.
- 각 정후의 질문에 대해서 정후가 당선될 가능성이 존재하려면 정후를 제외한 N 명의 후보들에게 y 개의 표를 적당히 배분하여 $A_{N+1} + x$ 미만이 되어야 합니다.
- 정후의 질문이 들어올 때마다, 1 번 후보부터 N 번 후보까지 보면서 각각의 후보가 정후의 표보다 표 수가 적게 되기 위해 최대 몇 개의 표를 받을 수 있는지를 계산하고, 전부 더합니다.
- 만약 위에서 더한 값이 y 이상이라면 답이 “YES”이고, 그렇지 않다면 “NO”입니다.
- 시간 복잡도는 2번 쿼리마다 $\mathcal{O}\,(N)$, 전체 $\mathcal{O}\,(QN)$ 입니다.

B. 개표

만점 풀이 (100점)

- ✓ 부분 문제 2에서 사용한 답이 “YES”일 조건을 수식으로 나타내 봅시다.

1. 모든 $i\,(1 \le i \le N)$에 대해 $A_i < A_{N+1} + x$
2. $\sum_{i=1}^{N} (x + A_{N+1} - A_i - 1) \ge y$

- ✓ 1번 조건은 배열 전체 값과 배열의 최댓값을 관리해주면 $\mathcal{O}(1)$에 확인할 수 있습니다.
- ✓ 2번 수식을 변형하면 $N \times (x + A_{N+1} - 1) - \sum_{i=1}^{N} A_i \ge y$가 되고, 배열 전체의 합을 관리해 주면 역시 $\mathcal{O}(1)$에 확인할 수 있습니다.

C. Split the SSHS

`tree`, `tree_set`

출제 의도 – **Medium**

- 제출 32번, 정답 5명 (정답률 16%)
- 처음 푼 사람: 김기범, 68분
- 출제자: 문정후

C. Split the SSHS

부분 문제 1 (10점)

- ✓ 이 문제에서 정의된 '서울과학고등학교'와 같은 그래프를 일반적으로 트리라고 부릅니다. 건물은 '정점', 길은 '간선'입니다.
- ✓ 각 정점에 인접한 정점 중 1번 정점에 가장 가까운 정점을 부모라고 합시다.
- ✓ 조각은 다음과 같이 셀 수 있습니다.

1. 각 정점 v에서 v와 인접한 간선 중 v와 부모를 잇는 간선과 색이 다른 간선들을 생각합니다. 1번 정점은 인접한 모든 간선을 생각합니다.
2. 이들 중 색이 서로 다른 간선의 개수를 셉니다.
3. 모든 정점에 대해 센 값을 더합니다.

C. Split the SSHS

- dfs를 이용해 각 정점의 부모를 구합니다.
- 이제 나머지는 간단합니다. 기수 정렬을 이용해 $\mathcal{O}(QN)$ 알고리즘을 구현할 수 있으며, $\mathcal{O}(QN \log N)$ 알고리즘을 구현해도 무방합니다.

C. Split the SSHS

부분 문제 2 (13점)

- ✓ $1 \leq N \leq 200\,000, 1 \leq Q \leq 100\,000$ 입니다. $\mathcal{O}(QN)$ 은 너무 느립니다.
- ✓ 마침 트리가 선형입니다. 곧, 간선의 색을 바꿀 때마다 양옆 간선의 색만 보면 됩니다.
- ✓ 초기의 답은 $\mathcal{O}(N)$ 시간 안에 구할 수 있습니다.
- ✓ 경우를 따져 간선의 색이 바뀔 때마다 양옆에서의 답만을 $\mathcal{O}(1)$ 시간에 갱신합니다.
- ✓ 문제를 $\mathcal{O}(N+Q)$ 시간에 해결할 수 있습니다.

C. Split the SSHS

부분 문제 3 (32점)

- ✓ 이제 만점 풀이에 근접한 방법을 찾아야 합니다.
- ✓ 쿼리가 주어질 때마다 트리를 전탐색할 수는 없습니다.
- ✓ 부분 문제 2와 비슷한 방법을 이용합시다.

- ✓ 원래 간선의 색이 1이었고, 2로 바꾼다고 생각합시다.
- ✓ 색 1인 간선을 제거하고 색 2인 간선을 삽입한다고 생각합시다.

C. Split the SSHS

제거할 때에는 답에 다음과 같은 변화가 있습니다.

- ✓ 간선이 연결하는 두 정점과 인접한 색 1의 간선이 모두 존재한다면 조각의 수가 하나 늘어납니다.
- ✓ 간선이 연결하는 두 정점과 인접한 색 1의 간선이 존재하지 않는다면 조각의 수가 하나 줄어듭니다.

삽입할 때에는 답에 다음과 같은 변화가 있습니다.

- ✓ 간선이 연결하는 두 정점과 인접한 색 2의 간선이 모두 존재한다면 조각의 수가 하나 줄어듭니다.
- ✓ 간선이 연결하는 두 정점과 인접한 색 2의 간선이 존재하지 않는다면 조각의 수가 하나 늘어납니다.

C. Split the SSHS

- 간선의 색이 반대라면 비슷한 방법으로 답을 갱신합니다.
- 각 정점마다 인접한 간선 중 색 1의 간선의 수, 색 2의 간선의 수를 저장한다면 $\mathcal{O}(1)$ 시간 안에 답을 갱신할 수 있습니다.
- $\mathcal{O}(N+Q)$ 시간 안에 문제를 해결할 수 있습니다.

C. Split the SSHS

만점 풀이 (100점)

- 부분 문제 3과 같습니다.
- 다만, 이제 인접한 간선 중 색 c의 간선이 존재하는지를 빠르게 판별해야 합니다.
- 적절한 자료 구조를 이용합니다.
- std::multiset을 이용해도 좋고, splay tree 등의 이진 탐색 트리를 구현해도 좋습니다.
- 시간 복잡도는 $\mathcal{O}\left((N+Q)\log N\right)$ 입니다.

D. 센터가 돋보여야 해

`segment_tree`
출제 의도 – **Medium**

- 제출 77번, 정답 2명 (정답률 3%)
- 처음 푼 사람: 이성호, 74분
- 출제자: 이동현

D. 센터가 돋보여야 해

부분 문제 1 (3점)

- ✓ 각 쿼리마다 학생을 고를 수 있는 방법은 한 가지밖에 없습니다.
- ✓ 두 번째 종류의 동작마다 $A_2 - A_1 - A_3$ 을 출력합니다.
- ✓ 시간 복잡도는 $\mathcal{O}(N)$ 입니다.

D. 센터가 돋보여야 해

부분 문제 2 (15점)

- 2 l r 의 쿼리가 들어오면 $l < i < r$ 인 모든 i 에 대해 $\min_{l \le j < i} A_j$ 의 값을 B_i 라 합시다.
- 마찬가지로 $\min_{i < j \le r} A_j$ 의 값을 구해 C_i 라 합시다.
- 답은 $\max_{l < i < r} A_i - B_i - C_i$ 가 됩니다.
- 시간 복잡도는 $\mathcal{O}(NQ)$ 입니다.

D. 센터가 돋보여야 해

부분 문제 3 (26점)

- 2 l r 의 쿼리가 들어오면 $a = l, b + 1 = c$ 이어야 합니다.
- $A_b - A_c$ 의 최댓값은 세그먼트 트리를 이용하여 $\mathcal{O}(\log N)$ 에 구할 수 있습니다.
- 시간 복잡도는 $\mathcal{O}(N + Q \log N)$ 입니다.

D. 센터가 돋보여야 해

만점 풀이 (100점)

- 세그먼트 트리에서 어떤 노드가 구간 $[l, r]$ 을 나타낼 때 다음 다섯 개의 값을 관리합니다.
- $$\begin{cases} \min\limits_{l \le i \le r} A_i & \max\limits_{l \le i \le r} A_i \\ \max\limits_{l \le i < j \le r} (A_i - A_j) & \max\limits_{l \le i < j \le r} (A_j - A_i) \\ \max\limits_{l \le i < j < k \le r} (A_j - A_i - A_k) & \end{cases}$$
- 세그먼트 트리의 각 노드에 대해 두 자식 노드의 정보를 이용하여 각각의 값을 $\mathcal{O}(1)$ 에 구할 수 있습니다.
- 시간 복잡도는 $\mathcal{O}(N + Q \log N)$ 입니다.

E. 용암 점프

`dp, greedy`
출제 의도 – **Hard**

- 제출 51번, 정답 1명 (정답률 2%)
- 처음 푼 사람: 김기범, 146분
- 출제자: 문정후

E. 용암 점프

- 죄송합니다.

부분 문제 1 (4점)

- 첫째 발판과 둘째 발판 사이의 거리, 그리고 둘째 발판과 셋째 발판 사이의 거리만이 중요합니다.
- 이환이의 경로는 여섯 가지입니다.
- 경우의 수가 많지 않습니다. 거리의 대소에 따라 경우를 나눕니다.

E. 용암 점프

부분 문제 2 (13점)

- ✓ 이환이의 경로는 $N!$ 가지입니다.
- ✓ 각 경로를 시뮬레이션하며 경로가 조건을 만족하는지를 확인합니다.

시간 복잡도는 $\mathcal{O}\left(N \times N!\right)$ 입니다.

E. 용암 점프

부분 문제 3 (15점)

- ✓ 그리디 풀이를 생각합니다.
- ✓ 이환이는 두 배 이상씩 멀리 뛰어야 하므로, 매 순간마다 가장 가까운 발판으로 가야 합니다.
- ✓ 부분 문제 조건에 따라 가장 가까운 발판이 유일합니다.
- ✓ 각 발판에서 시뮬레이션합니다.

$2^{21} > 2\,000\,000$ 이므로 21회 이상 뛰면 좌표 범위를 벗어납니다. 따라서 $N > 21$ 이면 모든 답이 "NO"입니다. 앞으로 $N \leq 21$ 로 가정하겠습니다.

시간 복잡도는 $\mathcal{O}\left(TN^2\right)$ 입니다.

E. 용암 점프

부분 문제 4 (25점)

- 다항 시간 풀이가 필요합니다.
- 부분 문제 3의 그리디 풀이를 확장합니다.
- 매 순간마다 최대 두 개의 발판 중 하나를 선택해야 합니다.
- 두 발판을 선택하는 경우를 전탐색하면 지수 시간이 걸리므로, 동적 계획법을 이용합니다.
- $f_{x,y,z,w}$ 에서 x 는 현재 위치에서 $[y, z]$ 를 방문했음을 의미하며, w 는 직전에 움직인 거리입니다.
- w 는 임의의 두 발판 사이의 거리이므로 w 의 종류는 $\mathcal{O}\left(N^2\right)$ 개입니다.

시간 복잡도는 $\mathcal{O}\left(TN^5\right)$ 입니다.

E. 용암 점프

부분 문제 5 (38점)

- ✓ 부분 문제 4의 풀이에서 상태를 줄입니다.
- ✓ $[y, z]$를 방문했다면 x는 y 또는 z입니다.
- ✓ $y' = x, z' = y + z - y'$로 두면 상태를 줄일 수 있습니다.

시간 복잡도는 $\mathcal{O}\left(TN^4\right)$입니다.

E. 용암 점프

부분 문제 6 (52점)

- ✓ 더 줄일 수 있습니다.
- ✓ w를 직전에 방문한 발판의 위치로 두면 상태가 줄어듭니다.

시간 복잡도는 $\mathcal{O}\left(TN^3\right)$ 입니다.

E. 용암 점프

만점 풀이 (100점)

- ✓ 더 줄일 수 있습니다.
- ✓ w로 가능한 발판은 둘뿐입니다.
- ✓ $w = y + 1$ 또는 $w = z - 1$입니다.

시간 복잡도는 $\mathcal{O}\left(TN^2\right)$ 입니다.

F. 지역 순회

`binary_search`, `smaller_to_larger`, `tree_dp`
출제 의도 – **Hard**

- 제출 1번, 정답 0명 (정답률 0%)
- 처음 푼 사람: 없음
- 출제자: 박종경

F. 지역 순회

문제 요약

- 지역은 트리를 구성합니다.
- 정후의 지역 순회 경로가 정해지면 시작한 지역과 끝나는 지역에서는 반드시 홍보를 해야 하고, 순회 경로 상에서 연속한 M 개의 지역 중 적어도 하나에서는 홍보를 해야 합니다.
- 순회 경로 중 홍보하는 지역에 대해 a_i 의 정치적 지지를 얻고 이에 t_i 의 시간이 소요될 때, 영우는 $\sum a_i$ 의 최댓값, 종경이는 $\frac{\sum a_i}{\sum t_i}$ 의 최댓값을 구하고자 합니다.

F. 지역 순회

관찰

- 종경이의 문제를 영우의 문제로 환원시켜봅시다.
- '$\frac{\sum a_i}{\sum t_i} \geq k$ 인가?'라는 명제가 참이 되는 k의 최댓값이 종경이의 문제의 답입니다.
- 위 명제는 $\sum a_i \geq k \times \sum t_i$와 같고, $\sum (a_i - k \times t_i) \geq 0$과 동치입니다.
- $b_i = a_i - k \times t_i$라 하면, 위 명제는 $\sum b_i \geq 0$으로 쓸 수 있습니다.

F. 지역 순회

관찰

- ✓ 어떤 k_{max} 가 있어서, k_{max} 이하인 k 에 대해서만 위 명제가 성립합니다.
- ✓ 이 값은 k 에 대한 이분탐색으로 구할 수 있습니다.
- ✓ $\sum b_i$ 의 최댓값을 찾는 방법은 영우의 문제와 동일합니다.
- ✓ 따라서 가능한 답이 될 수 있는 수의 종류를 X 가지라 한다면, 종경이의 문제는 영우의 문제의 $\mathcal{O}(\log X)$ 배의 시간 복잡도로 풀립니다.

종경이의 문제는 영우의 문제에 이분탐색을 추가해 풀 수 있으므로, 영우의 문제를 해결하는 방법만 설명합니다. 위 관찰을 하지 못하고 영우의 문제만 풀면 절반의 점수를 얻습니다.

F. 지역 순회

부분 문제 1 (4점)

- 1번 정점을 루트로 하고, dfs로 각 정점의 깊이 dep_v 를 구합니다.
- dp_v 를 v 에서 시작하고, 자손 중 하나에서 끝낼 때 얻을 수 있는 최대 정치적 지지로 정의합니다.
- dp_v 의 초깃값은 a_v 이고, 값은 아래와 같이 갱신합니다.
- v 의 자손 정점 중 v 와 거리가 M 이하인 모든 c 에 대해, $dp_v := max(dp_v, a_v + dp_v)$
- 1번 정점 외의 다른 모든 정점도 루트로 잡고, DP를 총 N 번 합니다.

답은 N 번의 트리 DP 과정 중 최대의 dp_v 값입니다.
$\mathcal{O}\left(N^2\right)$ 의 시간 복잡도의 DP를 N 번 실행하므로 총 시간 복잡도는 $\mathcal{O}\left(N^3\right)$ 입니다.
N 번의 트리 DP로 가능한 모든 경로를 고려합니다.

F. 지역 순회

부분 문제 2 (8점)

$M = N$ 이므로 M 조건을 고려하지 않아도 됩니다.

- ✓ dp_v 를 부분 문제 1과 같이 정의하고, 깊이 우선 탐색 순서대로 구합니다.
- ✓ v의 모든 자식 c에 대해서 $dp_v = \max(dpc) + a_v$ 가 됩니다.

dp 를 모두 구했으니, 깊이 우선 탐색하며 답을 구합시다.

F. 지역 순회

- ✓ 정점 v에 도달했을 때, v를 지나면서 v를 루트로 하는 서브트리에 포함된 경로를 고려합시다.

1. v를 끝점으로 하는 경로
2. v를 경유하는 경로

- ✓ 첫 번째 경우 최댓값은 dp_v 입니다.
- ✓ 두 번째 경우, 경로상에서 v와 인접하게 나타나는 정점은 v의 자식입니다.
- ✓ 두 자식을 c_1, c_2라 하면 v에서 홍보하면 $dp_{c_1} + dp_{c_2} + a_v$, 안 하면 dp_{c_1}, dp_{c_2} 만큼의 정치적 지지를 얻습니다. 따라서 c_1, c_2는 dp값이 가장 큰 두 개의 자식을 고르는 것이 이득입니다.

  위 두 과정은 트리 DP 한 번에 가능하므로 총 시간 복잡도는 $\mathcal{O}(N)$ 입니다.

F. 지역 순회

부분 문제 3 (8점)

- ✓ $M = 1$이므로, 순회 경로상의 모든 정점에서 홍보해야 합니다.
- ✓ 이는 부분 문제 2의 풀이와 비슷하게 해결할 수 있고, 시간 복잡도 또한 동일합니다.

F. 지역 순회

부분 문제 4 (18점)

지역이 선형으로 배열되어 있습니다.

- ✓ dp_v를 정점 1부터 정점 v까지 고려하고 v를 선택할 때 얻을 수 있는 가중치 합의 최댓값으로 정의합니다.
- ✓ $dp_v = max_{v-M \le i < v}(dp_i + a_v)$라는 점화식을 얻을 수 있습니다.

위 알고리즘은 시간 복잡도가 $\mathcal{O}(NM)$인데, deque를 이용하는 $\mathcal{O}(N)$ 방법으로 최적화 할 수 있습니다.

F. 지역 순회

- deque D에 (i, dp_i)의 값을 저장하고, $i < v - M$인 것은 D에서 삭제합니다.
- $i < j$이고 $dp_i \leq dp_j$라면 i쪽을 고려하는 것보다 j쪽을 고려하는 것이 더 효율적이므로, i쪽을 D에서 삭제합니다.

N개의 값이 deque에 삽입, 삭제되는 횟수는 최대 한 번이므로 시간 복잡도는 $\mathcal{O}(N)$입니다.

F. 지역 순회

부분 문제 5 (총 22점)

$dp_{v,d}$를 v에서 d 이하 떨어진 곳에서 시작해서 루트에서 멀어지는 방향으로만 이동할 때 얻을 수 있는 $\sum a_i$의 최댓값으로 정의합니다. 깊이 우선 탐색 과정에서 한 정점 v에 도착하면 dp_v 값들을 구하고, v 관련된 경로를 처리합니다.

- ✓ 정점 v를 방문했을 때 $dp_{v,i}$ 값들을 구합니다. v의 자식들을 c_i로 씁시다.
- ✓ c_i들과 v의 거리는 1이므로 $1 \le k \le M$인 k에 대해 $dp_{v,k} = \max_{1 \le i \le k} dp_{c_i,k-1}$
- ✓ v를 시작점으로 하는 경우 $d = 0$이고,
  $dp_{v,0} = a_v + \max_{1 \le i \le k, 0 \le d \le M} (dp_{i,j}) + a_v = \max_{1 \le i \le M} (dp_{v,i}) + a_v$.
- ✓ 위 $dp_{v,d}$ 값들은 v에서 **정확히** d 떨어진 곳에서 시작하는 경우의 값입니다. 정의를 충족하기 위해 작은 d부터 보면서 $dp_{v,i} := max(dp_{v,i}, dp_{v,i-1})$로 상태 변이합니다.

F. 지역 순회

v의 $dp_{v,i}$들을 빠르게 구하는 방법은 나중에 알아봅니다. v와 관련된 경로들을 봅시다.

1. v에 도달했을 때, v에서 시작하는 경로의 최대 정치적 지지는 $dp_{v,0}$입니다.
2. v와 그의 두 자식 c_1, c_2를 지나고, v에서 홍보하지 않는 경로의 최대 정치적 지지는 $\max_{0 \le i \le M-2} (dp_{c_1,i} + dp_{c_2,M-2-i})$입니다.
3. v와 그의 두 자식 c_1, c_2를 지나고, v에서 홍보하는 경로의 최대 정치적 지지는 $\max_{0 \le i,j \le M-1} (dp_{c_1,i} + dp_{c_2,j}) + v_a$입니다.

✓ 첫 번째 경우는 dp 배열만 활용해서 구할 수 있습니다.

F. 지역 순회

두 번째 경우와 세 번째 경우는 dp를 구하면서 함께 구할 수 있습니다. $dp_{v,i}$의 초깃값은 충분히 작은 수로 둡시다.

✓ i를 1부터 k까지 늘리며 다음을 반복합니다. 두 번째, 세 번째 경우의 답을 각각 X, Y라 둡시다.

1. $0 \le j \le M-2$인 j에 대하여, $X := max(X, dp_{v,j} + dp_{c_i,M-j-2})$
2. $Y := \max(Y, dp_{c_i,M-1} + dp_{v,M})$
3. $dp_{v,0} := \max(dp_{v,0}, dp_{c_i,M-1} + a_v)$
4. $1 \le i \le M$인 i에 대해 $dp_{v,i} = max(dp_{v,i}, dp_{c_i,i-1})$

1에서 4 과정은 $\mathcal{O}(M)$의 시간 복잡도가 소요되고, 각 정점에서는 그 정점의 차수만큼 반복하므로 전체 시간 복잡도는 $\mathcal{O}(NM)$입니다.

F. 지역 순회

만점 풀이 (100점)

- ✓ 부분 문제 5의 풀이를 std::set 자료 구조를 이용하여 최적화합니다.
- ✓ 정점마다 set s_v를 만들고, $(i, dp_{v,i})$ 형태의 순서쌍들을 저장합니다.
- ✓ 단, 두 순서쌍 (i, d_i), (j, d_j)가 있어서 $i \leq j$이고 $d_i = d_j$이면 (j, d_j)는 저장하지 않습니다.
- ✓ 즉, $dp_{v,i}$ 값이 일정하지 않고 순증가하는 i에서만 set에 저장합니다.

이제 부분 문제 5의 풀이에 적용합시다.

F. 지역 순회

- ✓ 경로를 확인하는 방법은 부분 문제 5와 비슷합니다.
- ✓ 다음과 같은 알고리즘으로 모든 경로를 확인하는 동시에 s_v 까지 만들 수 있습니다.

  v 의 자식을 $c_1, ..., c_k$ 로 정의하고, i 를 1부터 k 까지 증가시키며 아래 알고리즘을 반복합니다.

1. s_{c_i} 의 모든 원소 d_1, val_1 에 대해서, s_v 의 원소 d_2, val_2 들 중 $d_1 + d_2 - 2 \times dep_v < M$ 이고 d_2 가 최대인 원소를 찾고, 부분 문제 5의 풀이와 비슷하게 답을 갱신합니다.
2. s_{c_i} 의 각 원소를 s_v 에 추가하고, 앞 슬라이드에서 설명한 조건에 맞게 하기 위해 몇 개의 원소를 제거합니다.

   1 과정은 $\mathcal{O}(\log N)$ 에 할 수 있고, 2 과정 역시 $\mathcal{O}\left((s_{c_i}\text{의 원소수}) \times \log N\right)$ 에 할 수 있습니다.

F. 지역 순회

- 위 알고리즘은 $\mathcal{O}\left(N \log(N)\right)$에 동작할 것으로 보입니다.
- 그러나, $|s_v|$는 최대 $N - dep_v$까지 될 수 있고, 트리에서 $\sum_{v=1}^{N} N - dep_v$는 $\mathcal{O}\left(N^2\right)$입니다.

  따라서 전체 시간 복잡도는 $\mathcal{O}\left(N^2 \log N\right)$입니다.
- 이를 최적화하기 위해 smaller to larger로 불리는 테크닉을 사용합니다.
- 전 슬라이드의 2번 과정에서는 두 집합을 합치면서 필요가 없는 원소들을 제거합니다. 따라서, $|s_v| < |s_{c_i}|$라면 두 집합을 맞교환 한 뒤 s_{c_i}의 원소를 s_v의 원소로 넣어도 결과가 같습니다.
- $|s_v|$는 최대 $N - dep_v$이므로, 위 방법에서 원소의 이동 횟수의 합은 $\mathcal{O}\left(N\right)$입니다.

G. 균형 발전

`dfs`, `dnc`, `segtree`

출제 의도 – **Challenging**

- 제출 3번, 정답 0명 (정답률 0%)
- 처음 푼 사람: 없음
- 참가자 평균 점수: 0점
- 출제자: 박영우

G. 균형 발전

문제 요약

- 트리 그래프를 이루는 도시들이 있을 때 시각 i에는 도시 T_i가 활성화됩니다.
- 도시 v가 활성화되면 그 도시와의 거리가 R_v 이하인 모든 자손 도시에 X_v의 인구 유입이 일어납니다.
- 총 인구 유입이 C_v 이상인 도시는 활성화됩니다.
- 각 도시가 활성화되는 시각을 구해야 합니다.

G. 균형 발전

용어 정의

- 한 도시 T_i가 시각 i에 활성화되었다면 T_i를 직접 활성화된 도시라고 합시다.
- 두 도시 u, v에 대해 u가 활성화되어서 v에 인구 유입이 일어나고, 그로 인해 v가 활성화되면 u가 v를 활성화한다고 합시다. 또한, v를 간접적으로 활성화된 도시라고 합시다.
- $RT_{T_i} = i$로 정의합시다. 즉, RT_i는 i번 도시가 직접 활성화될 경우 그 도시가 활성화되는 시간입니다.
- 도시 N개가 루트가 있는 트리를 구성하므로, 일반적인 트리와 마찬가지로 조상, 자손 정점과 같이 조상 도시, 자손 도시를 정의합니다.

G. 균형 발전

부분 문제 1 (6점)

- ✓ 모든 도시를 $T_1, T_2, \ldots$ 순서대로 보면서, 활성화되지 않았다면 활성화하고, 그 도시가 활성화하는 도시들을 다음과 같이 찾아 줍니다.

  활성화된 도시를 v라 합시다. 전처리로 dfs를 해 각 도시의 깊이 dep_v를 구해줍니다.

1. 큐를 하나 만들고 v를 추가합니다.
2. 큐에서 한 원소 v를 뽑습니다.
3. v에서 자식 방향으로만 깊이 우선 탐색하며 v의 모든 자손 도시 x에서 $dep_x - dep_v \leq R_v$, $C_x > 0$이라면 C_x에서 X_v를 빼고, 이후 $C_x \leq 0$이면 큐에 x를 추가합니다.
4. 큐가 비어 있지 않으면 2로 되돌아갑니다.

- ✓ 한 도시에서 dfs에 요구되는 시간 복잡도는 $\mathcal{O}(N)$이므로 전체 시간 복잡도는 $\mathcal{O}(N^2)$ 입니다.

G. 균형 발전

부분 문제 2 (7점)

- 부분 문제 2에서는 도시 v가 직접 활성화되었을 때, v로 인해서 인구 유입이 일어나는 도시는 $v+1$이 유일합니다.
- 따라서, 한 도시 v가 활성화되었을 때 v가 활성화하는 도시의 집합은 v를 포함하는 구간이 됩니다.
- 이제 부분 문제 1의 풀이와 비슷하게, 모든 도시를 활성화되는 시간 순서대로 보면서 그 도시가 활성화되지 않았다면 활성화하고, 그 도시부터 몇 번 도시까지 활성화되는지를 일일이 확인해서 구합니다.
- 위 과정에서, 한 도시 v가 다른 도시에게 확인되는 횟수는 최대 한 번입니다. 따라서 전체 시간 복잡도는 $\mathcal{O}(N)$입니다.

G. 균형 발전

부분 문제 3 (18점)

- 부분 문제 3에서는 트리가 선형입니다.
- 따라서 한 도시가 직접 활성화되었을 때 그 도시로 인해 인구 유입이 일어나는 도시들은 구간을 이루게 됩니다. 이 사실을 이용해 부분 문제 1의 풀이를 최적화합니다.
- 부분 문제 1의 탐색 과정은 하나의 도시가 활성화되었을 때 그 도시로 인해 인구 유입이 일어나는 모든 도시에 대해 활성화 여부를 일일이 확인합니다.
- 그러나 부분 문제 3에서는 그러한 도시들이 연속한 구간을 이루게 되므로, 세그먼트 트리를 사용해 해결할 수 있습니다.

G. 균형 발전

부분 문제 1의 활성화된 도시 처리 과정을 아래와 같이 바꿔 쓸 수 있습니다.

1. 큐에서 한 원소 v를 뽑습니다.
2. v로 인해 인구가 유입되는 구간을 찾고, 구간에 속한 원소 중 C 값이 최소인 원소 x를 찾습니다.
3. $C_x \leq X_v$ 라면 x를 큐에 넣고 활성화합니다. 이를 $C_x \leq X_v$ 인 동안 계속 반복합니다.
4. 세그먼트 트리의 lazy propagation으로 2에서 구한 구간의 모든 원소에서 X_v를 뺍니다.

✓ 2번 과정에서의 구간은 도시들의 깊이로 이분 탐색을 함으로써 찾을 수 있습니다.

✓ 세그먼트 트리의 연산은 모두 $\mathcal{O}(\log N)$ 시간에 가능하고 각 도시는 한 번만 활성화되므로 전체 시간 복잡도는 $\mathcal{O}(N \log N)$ 입니다.

G. 균형 발전

부분 문제 4 (11점)

- 부분 문제 4에서는 각 도시가 활성화되었을 때 그 도시로 인해 인구 유입이 일어나는 도시들은 활성화된 도시를 루트로 하는 서브트리를 이룹니다.
- 부분 문제 3의 풀이를 부분 문제 4에도 적용하기 위해 Euler Tour Trick(ETT)를 사용합니다.
- 루트에서부터 깊이 우선 탐색하며 l_v를 탐색에서 v가 방문된 순서라고 합시다.($l_1 = 1$)
- r_v를 v를 v의 모든 자식 x에 대해 l_x의 최댓값이라 합시다.
- 이렇게 하면 v가 활성화되었을 때 $l_v \leq l_x \leq r_v$인 모든 도시 x에 인구 유입이 일어납니다.
- 이제 부분 문제 3과 완전히 같은 풀이로 풀 수 있습니다.

부분 문제 5 (11점)

- 부분 문제 5에서는 R_i의 값이 N이 아니므로, 부분 문제 4의 방법은 사용할 수 없습니다.
- 각 도시에 대해 R_i의 값이 동일하므로 $R = R_i$로 간략히 표기합시다.
- 또한 부분 문제 4번 풀이의 구간 $[l_v, r_v]$를 $range_v$로 표기합시다.

G. 균형 발전

- 부분 문제 5에서는 한 도시로 인해 인구 유입이 일어나는 도시가 하나의 구간을 이루지 않고 여러 개의 구간을 이루게 됩니다.
- 도시 v와 거리가 정확히 R만큼 떨어진 v의 자손의 집합을 D_v라 합시다.
- 그러면, v가 활성화되었을 때 v로 인해 인구 유입이 일어나는 도시의 집합은 $range_v - \bigcup_{x=D_v} range_x$입니다.
- 이제 $range_v$를 구하고, 각 도시에 대해 D_v를 모두 구하면 부분 문제 4와 비슷하게 풀 수 있을 것으로 보입니다.

G. 균형 발전

- ✓ 다음과 같은 관찰이 필요합니다.
- ✓ $\sum_{v=1}^{N} |D_v| \leq N$

1. 각 도시 v에서, $v \in D_x$인 x의 개수를 찾읍시다.
2. 만약 두 x_1, x_2에서 $v \in D_{x_1}, v \in D_{x_2}$이면 x_1, x_2는 모두 v의 조상 도시가 되고, 트리의 성질에 따라 v에서 x_1, x_2까지의 거리는 서로 달라야 합니다.
3. 그러나 정의에 따라 x_1, x_2에서 v까지의 거리는 서로 같고, 따라서 v는 최대 하나의 x에서 $v \in D_x$입니다. 따라서 위 관찰이 성립합니다.

G. 균형 발전

- ✓ 이제 D_v 를 구하는 방법을 알아봅시다. 깊이 우선 탐색으로 구할 수 있습니다.
- ✓ 도시 v 의 모든 자손의 D 를 전부 구한 상황에서 D_v 를 구해 봅시다.

1. v 의 모든 인접한 자식 도시의 D 의 합집합을 S 로 지정합니다.
2. S 의 모든 원소에 대해 그 원소의 부모 도시를 D_v 에 넣습니다.
3. 또한 중복된 원소는 없앱니다.

   위 알고리즘은 전 슬라이드의 증명과 같은 방식으로 시간 복잡도가 $\mathcal{O}(N)$ 이라는 것을 증명할 수 있습니다.

G. 균형 발전

- 이제 부분 문제 4의 풀이를 적용할 수 있습니다.
- 우선 전 슬라이드의 알고리즘으로 D_i를 모두 구합니다.
- 한 도시 v가 활성화되면 v로 인해 직접 인구가 유입되는 $|D_v| + 1$ 개의 구간에 인구를 더합니다. 이는 부분 문제 4의 자료 구조를 그대로 이용할 수 있습니다.
- 부분 문제 4와 똑같이, v로 인해 활성화되는 도시들을 모두 찾아서 큐에 넣습니다.
- 알고리즘의 시간 복잡도는 D_i들을 찾는 데 $\mathcal{O}(N)$, 각 도시를 업데이트하는 데 $\mathcal{O}((|D_v| + 1) \times \log N)$이며 D_v의 합을 찾는 데 $\mathcal{O}(N)$이므로 전체 시간 복잡도는 $\mathcal{O}(N \log N)$입니다.

G. 균형 발전

만점 풀이 (총 100점)

- 만점 풀이는 부분 문제와 큰 관련이 없으며, 새로운 접근 방법을 요구합니다.
- 기본적인 아이디어는 각 도시가 활성화되는 시각을 이분 탐색으로 찾는 것입니다.
- $f(V, P, s, e)$를 도시 배열 V가 있어 V의 각 원소의 부모 배열이 P라 하고, 각 도시가 $[s, e]$ 시각에 활성화됨이 보장되어 있을 때 각 도시가 활성화되는 시각을 구하는 함수라고 합시다.
- 또한 각 도시와 1번 도시까지의 거리(깊이)인 $depth_v$를 미리 계산해 둡니다.
- f를 재귀적으로 설계합시다.

G. 균형 발전

- ✓ 이분 탐색을 적용하기 위해 $m = \lfloor (s+e)/2 \rfloor$ 시각에 활성화되는 도시를 구합시다.
- ✓ 이 경우, 간단한 풀이는 깊이 우선 탐색을 응용합니다.

1. 탐색 방문 순서대로 활성화 여부를 결정합니다.
2. 도시 v의 활성화 여부에 영향을 줄 수 있는 도시는 v의 조상 도시들뿐입니다.
3. 각각의 부모 도시들이 활성화되어 있는지의 여부를 확인해 v 도시가 활성화되어 있는지 확인하면 됩니다. 이때, $RT_v \leq m$인 경우에도 v가 활성화됨에 유의해야 합니다.

- ✓ 각 도시의 활성화 여부를 결정하는 데에 $\mathcal{O}(N)$ 시간이 듭니다. 이는 너무 느립니다.

G. 균형 발전

- ✓ 깊이 우선 탐색 방문 순서대로 활성화 여부를 결정하므로, 세그먼트 트리를 사용해 아래와 같이 최적화할 수 있습니다. 편의상 세그먼트 트리를 arr_i 의 배열로 표기합시다.

1. v 에 방문했을 때 $RT_v \leq m$ 또는 $arr_{dep_v} \geq C_v$ 이면 v 를 활성화합니다.
2. v 가 활성화되었다면, $i \in [dep_v, dep_v + R_v]$ 인 i 에서 $arr_i := arr_i + X_v$.

- ✓ arr_i 의 원소가 너무 많으므로, 동적 세그먼트 트리를 사용할 수 있습니다.
- ✓ 구간 업데이트 쿼리로 들어오는 영역의 상한과 하한이 $\mathcal{O}(N)$ 개이므로 좌표 압축을 이용한 세그먼트 트리와 펜윅 트리를 사용할 수 있습니다. 개선된 시간 복잡도는 $\mathcal{O}(\log N)$.

G. 균형 발전

- 시각 m 이전에 활성화되는지의 여부로 도시를 두 집합으로 나누었으니 분할 정복을 사용합니다.
- 우선 시각 m에 활성화되는지의 여부에 따라 활성화된 도시의 배열을 X, 나머지의 배열을 Y 라 합니다.
- 배열 PX_i를 도시 X_i의 조상 도시 중 X에 속한 가장 가까운 도시로 정의합니다.(그런 것이 없다면 $PX_i = 0$.) 마찬가지로 PY_i도 정의합니다.
- 위 방법을 생각해 보면, 포레스트를 이루는 도시 집합 V를 두 개의 포레스트로 분할했음에도 조상 도시와 자손 도시의 거리 관계는 변함없습니다.
- 위 방법은 깊이 우선 탐색을 사용해 한번에 할 수 있고, 동시에 각 도시의 활성화 여부를 판단할 수도 있습니다.

G. 균형 발전

- 이제 X 와 Y 에서 활성화되는 시각을 구합니다.
- X 가 활성화되는 시각은 $[s, m]$ 에 속합니다. 이 시각에 Y 의 도시는 전혀 활성화되지 않으므로, Y 의 도시가 X 의 도시에 주는 영향은 없습니다. 따라서 $f(X, PX, s, m)$ 을 실행하면 됩니다.
- Y 에 속한 도시가 활성화되는 시각은 $(m, e]$ 에 속합니다. 이 시간에는 모든 X 에 있는 도시가 활성화되고, 이들 도시로부터 영향을 받습니다.
- 따라서 각 도시가 시각 m 에 활성화되는지의 여부를 판단할 때 각 활성화되지 않은 도시에 대해 얼마나의 인구 유입이 있었는지를 확인하고, 배열 C 에서 그만큼의 값을 빼서 X 에 속한 도시들의 영향을 고려해야 합니다. 이후에 $f(Y, PY, m+1, e)$ 를 실행하면 됩니다.

G. 균형 발전

- 시간 복잡도를 분석합시다.
- $f(V, P, s, e)$ 에서는 두 번의 깊이 우선 탐색(m 시간에 가능한지 판별하는 것과 새 부모 배열을 구하는 것)이 실행되고, 이때의 시간 복잡도는 $\mathcal{O}(|V| \log N)$ 입니다.
- 각각의 도시 v 에 대해 $v \in V$ 인 함수 $f(V, P, s, e)$ 의 실행 횟수는 일반적인 이분 탐색과 동일하게 $\mathcal{O}(\log N)$ 입니다.
- 따라서 전체 시간 복잡도는 $\mathcal{O}\left(\log^2 N\right)$ 입니다.

G. 균형 발전

구현 디테일

- 풀이의 특성상 배열을 인자로 사용하는 함수가 있고, 따라서 풀이의 상수도 큽니다.
- 사용할 수 있는 최적화로는 앞서 설명한 두 깊이 우선 탐색을 하나로 합치는 것, PX, PY 배열을 인자로 전달하지 말고 전역 변수로 사용하는 것 등이 있습니다.
- 또한 동적 세그먼트 트리로는 주어진 시간 제한 내에 동작하는 것은 힘들므로, 좌표 압축을 통한 세그먼트 트리나 누적 합을 이용해 그보다 빠르게 동작하는 펜윅 트리를 사용하는 것이 좋습니다.
- 상술한 풀이와는 별개로 이차원 세그먼트 트리를 이용한 풀이도 존재합니다.