목차

문제		의도한 난이도	출제자
A	당신은 운명을 믿나요?	Beginner	김태윤 ystaeyoon113
B	Parity Constraint Closest Pair (Easy)	Easy	이국렬 lky7674
C	어깨동무	Easy	노윤헌 edwardblue
D	대회 이름 정하기	Medium	김건우 coconut99
E	마계안암	Medium	김태윤 ystaeyoon113
F	고연전/연고전 기차놀이	Medium	노윤헌 edwardblue
G	연고전/고연전	Hard	이국렬 lky7674
H	간단한 쿼리 문제	Challenging	강인구 Lawali
I	트리 위의 세 사람	Challenging	김동현 kdh9949

A. 당신은 운명을 믿나요?

`greedy`
출제진 의도 – **Beginner**

- ✓ 제출 131번, 정답 74명 (정답률 57.252%)
- ✓ 처음 푼 사람: **국렬귀여워(연세대학교)**, 1분
- ✓ 출제자: **김태윤(ystaeyoon113)**

A. 당신은 운명을 믿나요?

- ✓ 문자열의 일부를 제거하고 얻을 수 있는 문자열은, 거꾸로 생각하면 순서를 유지한 채로 일부 글자들을 뽑은 문자열이 됩니다.
- ✓ 서브태스크 1은 점괘에 K가 존재하지 않습니다. 그래서 굉장히 쉽게 풀 수 있습니다.
- ✓ 점괘에서 항상 KOREA와 YONSEI 둘 중 하나를 찾을 수 있는데, 점괘에 K가 없으므로 KOREA를 찾을 수는 없습니다.
- ✓ 따라서 정답은 무조건 YONSEI입니다.

A. 당신은 운명을 믿나요?

- ✓ 서브태스크 2는 일반적인 경우입니다.
- ✓ 앞에서부터 읽으면서 일부 문자열들을 뽑아 KOREA나 YONSEI를 만들 수 있다면, 해당 문자열을 출력하면 됩니다.
- ✓ 이때, 만약 i번째 글자가 현재 상황에서 문자열을 만드는 데 필요한 글자라면 뽑는 것이 유리합니다.
- ✓ 가령 KOREA를 찾는 상황에서 현재 KO까지 찾았고 i번째 점괘에서 R을 발견했다면, R을 뽑지 않는 것보다 뽑는 경우가 유리합니다.
- ✓ 그러므로 KOREA와 YONSEI에 대해 각각의 탐색 상황을 관리하면서 그리디하게 찾아나갈 수 있습니다.

B. Parity Constraint Closest Pair (Easy)

`sort`, `greedy`
출제진 의도 – **Easy**

- ✓ 제출 261번, 정답 53명 (정답률 20.307%)
- ✓ 처음 푼 사람: **종이비행기 (연세대학교)**, 9분
- ✓ 출제자: **이국렬 (lky7674)**

B. Parity Constraint Closest Pair (Easy)

- ✓ 서브태스크 1은 그냥 모든 두 점의 조합에 대해서 계산해 보면 됩니다.
- ✓ $O(n^2)$ 으로 쉽게 해결 할 수 있습니다.

B. Parity Constraint Closest Pair (Easy)

- ✓ 서브태스크 2는 관찰이 조금 더 필요합니다.
- ✓ 먼저 모든 좌표들을 정렬합니다.
- ✓ 먼저 두 점의 거리가 짝수이려면, 두 점의 parity가 같아야 합니다.
- ✓ 따라서 먼저 짝수인 점들만 고려했을 때 인접한 두 점의 거리가 최솟값의 후보가 됩니다. 이는 홀수인 점들의 경우도 똑같습니다.

B. Parity Constraint Closest Pair (Easy)

- ✓ 두 점의 거리가 홀수인 경우에는, 두 점의 parity가 달라야 합니다.
- ✓ 이때는 모든 인접한 숫자들을 관찰하여, 거리가 홀수인 경우에 후보가 됩니다.
- ✓ 인접하지 않은 숫자끼리의 거리는 절대로 후보가 될 수 없습니다. 이를 보이겠습니다.
- ✓ 만약 인접하지 않은 숫자 x와 z가 답이라고 하겠습니다. 서로 인접하지 않으므로 $x < y < z$인 어떤 y가 존재합니다.
- ✓ 이때 x와 z의 parity가 다르므로, 반드시 x와 y의 parity가 다르거나 y의 z의 parity가 다르고, $\max(y - x, z - y) < z - x$이므로 모순입니다.

C. 어깨동무

`binary search`
출제진 의도 – **Easy**

- ✓ 제출 206번, 정답 43명 (정답률 20.874%)
- ✓ 처음 푼 사람: **hellosworld16(연세대학교)**, 12분
- ✓ 출제자: **노윤헌(edwardblue)**

C. 어깨동무

- ✓ 답을 정해 놓고 이분탐색하는 파라메트릭 서치 문제입니다.
- ✓ H 값이 증가하면 지치는 사람의 수는 단조 감소합니다. 즉 이분 탐색을 사용할 수 있게 됩니다.
- ✓ H 값은 0 이상 10^9 이하이고, 단조성을 만족하므로, H 값이 정해졌을 때 지친 고려대생이 몇 명인지 구하는 문제를 해결할 수 있다면 이분탐색으로 답을 구할 수 있습니다.
- ✓ 이는 모든 학생들을 보면서 이웃한 사람과 키 차이를 보면서 지친 사람들을 세 주면 되므로, $\mathcal{O}(n)$ 의 시간이 걸립니다.
- ✓ 이분 탐색에는 $\mathcal{O}(\log \max(h))$ 의 시간이 걸리므로, 전체 시간복잡도는 $\mathcal{O}(n \log \max(h))$ 입니다.

D. 대회 이름 정하기

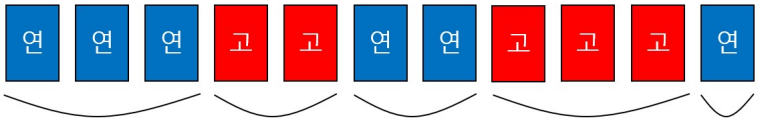
`binary search`, `segment tree`
출제진 의도 – **Medium**

- ✓ 제출 58번, 정답 18명 (정답률 31.034%)
- ✓ 처음 푼 사람: **ClockWorker(고려대학교)**, 29분
- ✓ 출제자: **김건우(coconut99)**

D. 대회 이름 정하기

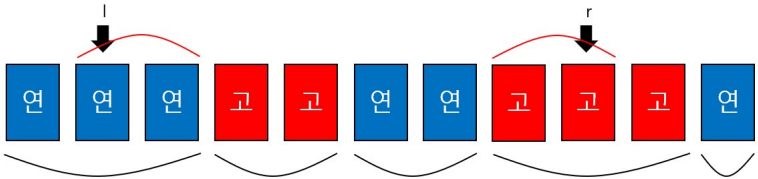
- ✓ 같은 글자를 연속해서 고를 수 없습니다.
- ✓ 그러므로 같은 글자끼리 묶어서 구간을 만들 때, 구간 하나에 최대 하나의 카드만 고를 수 있습니다.
- ✓ 그리고 그 구간에서 가장 큰 수를 고르는 것이 최선입니다.

D. 대회 이름 정하기



- 카드를 고를 수 있는 구간을 건너뛴다면, 다음 카드를 고르기 위해 구간을 하나 더 건너뛰어야 합니다.
- 두 구간을 건너뛴 다음에도 카드를 고를 수 있다면, 건너뛴 두 구간에서도 각각 카드를 하나씩 고를 수 있습니다.
- 그러므로 각 구간에서 가장 큰 수가 써진 카드를 하나씩만 고르는 것이 최선입니다.
- 서브태스크 1은 l 부터 r 까지 각 구간의 최댓값을 직접 구해서 풀 수 있습니다.

D. 대회 이름 정하기



- 서브태스크 2를 풀기 위해, 각 구간의 시작점과 끝점, 그리고 최댓값을 저장합니다.
- 최댓값을 누적합으로 저장하고, l과 r이 각각 어느 구간에 속하는지 이분탐색으로 찾습니다.
- 빨간 선으로 표시한 구간을 제외하면 Y와 K가 공통으로 고르게 될 구간입니다. 이 구간에서 고를 수 있는 수의 합을 누적합으로 $O(1)$에 구해줍니다. 이를 S라고 하겠습니다.

D. 대회 이름 정하기

- 이제 빨간 구간에 속하는 수의 최댓값을 구한 뒤, l 번째와 r 번째 카드의 글자에 따라 경우를 나눠줍니다. l, r 이 속한 구간의 최댓값을 각각 L, R 이라고 하겠습니다.
- l, r 이 같은 구간에 속하는 경우, Y와 K는 모두 카드를 고르지 못하므로 0점입니다.
- l 번째와 r 번째 카드가 '연고', '고연'일 때, 각각 Y와 K가 이깁니다. 점수는 $S + L + R$ 입니다.
- l 번째와 r 번째 카드가 '고고'일 때, Y와 K의 점수는 각각 $S + R, S + L$ 입니다.
- l 번째와 r 번째 카드가 '연연'일 때, Y와 K의 점수는 각각 $S + L, S + R$ 입니다.

D. 대회 이름 정하기

- ✓ 구간 최댓값 L ,R을 브루트포스로 찾는 방법으로 서브태스크 2를 풀 수 있습니다.
- ✓ 세그먼트 트리나 희소 배열로 구간 최댓값을 찾는다면 만점을 받을 수 있습니다.
- ✓ 시간 복잡도는 $O(N + QlogN)$ 또는 $O((N + Q)logN)$ 입니다.

E. 마계안암

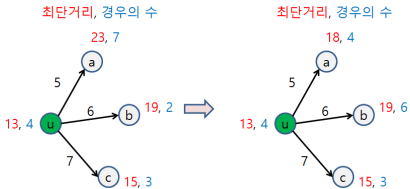
`dijkstra`, SCC
출제진 의도 – **Medium**

- ✓ 제출 52번, 정답 2명 (정답률 3.846%)
- ✓ 처음 푼 사람: **국렬귀여워**, 135분
- ✓ 출제자: **김태윤(ystaeyoon113)**

E. 마계안암

- ✓ 서브태스크 1의 경우 잘 알려진 문제입니다.
- ✓ 다익스트라 알고리즘을 이용해 안암역으로부터 최단거리를 구하는 과정과 동시에 경우의 수를 구해주면 됩니다.

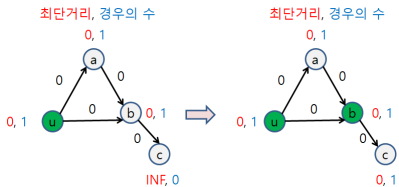
E. 마계안암



- ✓ 정점 u에서 다른 정점 v로 가는 것을 고려할 때 최단거리가 줄어들었다면 v의 최단거리를 갱신하고 v까지 최단 경로로 도달하는 경우의 수는 u까지의 경우의 수와 같습니다.
- ✓ 만약 u에서 v로 가는 거리가 현재까지 갈 수 있는 v의 최단거리와 같다면 v까지 도달하는 경우의 수에 u까지 도달하는 경우의 수를 더합니다.
- ✓ 위 알고리즘이 정당한 이유는 다익스트라 알고리즘의 동작은 각 정점을 최단거리가 짧은 순서대로 방문하는데, 간선의 가중치가 모두 양수이기 때문입니다.

E. 마계안암

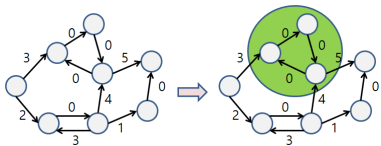
- ✓ 위 알고리즘은 서브태스크 2를 풀기에는 부족합니다. 그 이유는 다음의 반례에서 볼 수 있습니다.



- ✓ 위와 같이 가중치 0인 간선이 존재하는 경우에 최단거리가 같은 정점 간에도 가중치 0인 간선을 통해 연결될 수 있는데, 다익스트라 알고리즘에서 어떤 정점이 먼저 고려될지는 미지수입니다.
- ✓ 따라서 최단경로는 올바르게 구할 수 있으나, 경우의 수는 올바르게 구할 수 없습니다.

E. 마계안암

- 그래프에서 가중치가 0인 간선들만 이용해 SCC를 만들고, 같은 SCC에 속하는 정점은 하나로 합칩니다. 그리고 원래 간선들 중 다른 SCC끼리 연결된 간선들만 연결합니다.
- 이렇게 만들어진 그래프를 G라고 하겠습니다. 이때 G에는 0으로 이루어진 사이클이 없습니다. 따라서 G는 가중치가 0인 간선들만 고려했을 때 위상정렬이 가능한 그래프입니다.



- 또 관찰할 수 있는 것은, 하나의 SCC 안에 2개 이상의 정점이 들어있다면 해당 SCC에 속한 정점들은 모두 도달하는 경우의 수가 무수히 많습니다.

E. 마계안암

- G에서 다익스트라 알고리즘을 적용해 봅시다.
- G에 존재하는 정점 u에서 정점v로 가중치 0인 간선만 이용해서 도달할 수 있다면, 0인 간선들을 이용해 위상 정렬했을 때 u가 반드시 v보다 먼저 오게 됩니다.
- 다익스트라 알고리즘을 적용할 때 반드시 u는 v이전에 고려되어야 합니다.
- 따라서 다익스트라 알고리즘을 적용해 먼저 고려해야 하는 정점을 고를 때 거리만 고려하는 것이 아니고, 위상 정렬의 순서까지 고려해야 합니다.
- G를 만들 때 타잔 알고리즘을 사용하면, 동시에 위상정렬된 순서도 쉽게 알 수 있게 됩니다.
- 마지막으로 G의 정점들에 대한 경우의 수를 통해 원래 그래프의 경우의 수를 복원할 수 있습니다.

F. 고연전/연고전 기차놀이

`dynamic programming, deque`
출제진 의도 – **Medium**

- ✓ 제출 14번, 정답 5명 (정답률 35.714%)
- ✓ 처음 푼 사람: **노세윤**, 96분
- ✓ 출제자: **노윤헌(edwardblue)**

- ✓ K를 1로, Y를 -1로 간주하고 누적합(prefix sum)을 계산하면, 각 기차의 부분합의 절댓값은 1 이하가 되어야 합니다. i번째 원소까지의 누적합을 $S[i]$라고 합시다.
- ✓ 동적 계획법을 사용해서 문제를 해결해 봅시다.
- ✓ $dp[i]$를 i번째 사람까지 봤을 때 최소 기차 개수라고 정의합시다.
- ✓ 마지막 사람을 포함하는 기차의 누적합은 $1, 0, -1$이므로, 마지막 기차를 제외한 앞의 기차들의 누적합은 $S[i]+1, S[i], S[i]-1$ 중 하나입니다. 따라서 마지막 기차의 첫 사람은 $i-L+1$번째 사람부터 i번째까지 L명의 사람 중에서 누적합이 위의 셋 중 하나인 사람입니다.

F. 고연전/연고전 기차놀이

- ✓ 누적합이 같은 서로 다른 두 사람 중에서는 반드시 뒷 사람의 dp 값이 앞 사람보다 크거나 같습니다.
- ✓ 예를 들어 i 번째 사람을 포함하는 마지막 기차가 누적합이 $S[i]$ 인 사람 j 부터 시작되었다고 하면, j 는 $i-L+1, \ldots, i$ 번째 중에서 누적합이 $S[i]$ 인 첫 번째 사람인 경우가 최적입니다.
- ✓ 마지막 기차의 첫 사람의 누적합은 $S[i]+1, S[i], S[i]-1$ 의 셋 중 하나이므로, 이 누적합이 이 세 가지인 경우에 대해 i 번째 사람 앞의 L 명 중에서 가장 앞에 있는 사람을 구할 수 있으면 됩니다.
- ✓ 이는 각 누적합 별로 deque와 같은 선형 자료구조를 만들어 i 번째 사람 앞의 L 명의 누적합을 누적합별로 순서대로 관리하면 됩니다.
- ✓ 모든 누적합의 deque 안에는 L 개의 원소가 존재하고, 각 사람은 정확히 1 번 deque에 들어갔다 나가므로, 총 시간 복잡도는 $\mathcal{O}(n)$ 이 됩니다.

G. 연고전/고연전

`min-cut`
출제진 의도 – **Hard**

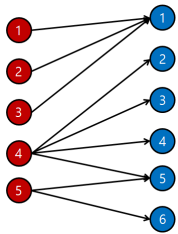
- ✓ 제출 9번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: **-(-)**, -분
- ✓ 출제자: **이국렬 (lky7674)**

G. 연고전/고연전

- ✓ 서브태스크 1은 $M \leq 18$ 이므로 고려대 팀원들을 선발하는 모든 경우의 수를 고려할 수 있습니다.
- ✓ 어떤 고려대 팀원들 $u_1, u_2, .., u_k$ 이 선발된 경우를 따져보겠습니다.
- ✓ 위 팀원들이 탈락시킬 수 있는 연세대 팀원들을 모두 구할 수 있습니다.
- ✓ 이중에서 (남은 연세대 팀원) - (남은 고려대 팀원)의 값이 최대가 되는 경우를 구해주면 됩니다.
- ✓ 모든 경우의 수를 고려하고, 각각의 경우에 모든 관계를 고려해야 하므로 $O(2^M \times (K+N))$ 에 해결할 수 있습니다.

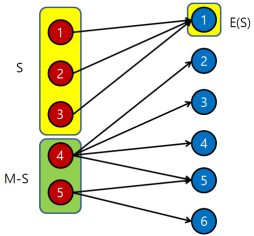
G. 연고전/고연전

- 서브태스크 2는 문제에서 원하는 요구사항을 살짝 바꿔서 생각해야 합니다.
- 즉, (남은 고려대 팀원) - (남은 연세대 팀원)의 값이 최소가 되는 경우를 구할 생각입니다.
- 각 인원을 정점으로 하고, 탈락시킬 수 있는 관계를 간선으로 이은 그래프를 생각해 봅시다.
- 그래프는 왼쪽에 고려대 팀원들이, 오른쪽에 연세대 팀원들이 위치한 이분 그래프로 생각할 수 있습니다.

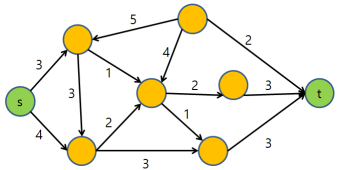


G. 연고전/고연전

- 국렬이가 선발한 고려대 팀원의 집합을 S라고 하고, 그리고 그 집합이 탈락시키는 연세대 팀원의 집합을 $E(S)$라고 하겠습니다.
- 이때 (남은 고려대 팀원) - (남은 연세대 팀원)의 값을 수식으로 표현하면 $(M-S)-(N-E(S))=(M-S+E(S))-N$입니다.



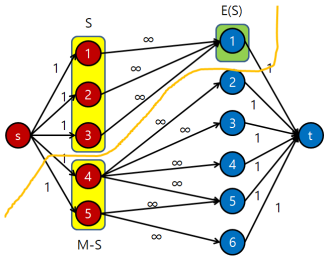
G. 연고전/고연전



- 한편, 다음과 같이 가중치가 있는 그래프에서 간선 몇개를 지워서 s에서 t로의 경로가 없도록 만드는 문제를 생각해 봅시다. 간선을 지울때는 그 간선의 가중치 만큼의 비용이 발생합니다.
- 간선을 지워서 s에서 t로의 경로가 없도록 하면 이를 s-t cut이라고 하고, 그때의 비용의 합을 cut cost라고 합니다. s-t cut 중 가장 cost가 작은 cut을 min-cut이라고 합니다.
- 각 간선의 가중치를 flow graph의 capacity로 바꾸어 모델링한 그래프의 maximum flow와, 원래 그래프의 min-cut cost가 같다는 정리가 maxflow-mincut theorem입니다.

G. 연고전/고연전

- ✓ 원래 문제로 돌아가서, 원래 그래프에 s와 t를 추가하고 s로부터 고려대 팀원들에는 가중치 1인 간선을 만들고, 연세대 팀원들로부터 t로 가중치 1인 간선을 만듭니다. 원래 그래프에 존재하던 관계의 가중치는 매우 큰 수의 가중치를 부여합니다.



- ✓ 이렇게 만든 그래프의 min-cut cost가 $M - S + E(S)$ 입니다.

G. 연고전/고연전

- ✓ 따라서 국렬이는 maximum flow가 구해진 상태에서 s로부터 도달할수 없는 고려대 팀원들을 S에 포함시키면 됩니다.
- ✓ 이때, 고려대 팀원을 한 명도 구하지 않는 경우가 생길 수 있습니다. 이는 문제의 조건을 위배합니다.
- ✓ 이를 위해서 국렬이가 어떤 고려대 팀원을 반드시 포함시키도록 그래프를 변형해 min-cut을 구해야 합니다.
- ✓ 그래서 총 M번의 그래프를 만들어 min-cut을 구하는 작업을 반복합니다. i번째 그래프를 만들 때, s에서 i번 고려대 팀원으로 향하는 간선의 cost가 매우 크도록 그래프를 만듭니다.
- ✓ 이렇게 하면 국렬이가 해당 그래프에서는 반드시 i번째 고려대 팀원을 선발한다는 것이 보장됩니다.

H. 간단한 쿼리 문제

`mo's algorithm`
출제진 의도 – **Challenging**

- 제출 17번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: **-(-)**, -분
- 출제자: **강인구(Lawali)**

H. 간단한 쿼리 문제

- ✓ 서브테스크 1은 쿼리로 주어진 구간을 정렬하는 것으로 나이브하게 풀 수 있습니다.
- ✓ 쿼리당 $O(n \log n)$에 해결할 수 있습니다.

H. 간단한 쿼리 문제

- ✓ 서브테스크 2는 모스 알고리즘을 통해 풀 수 있습니다.
- ✓ 구간에 대한 답을 구하는 문제는 어렵지만, 구간을 확장하거나 축소하는 것은 쉽습니다.
- ✓ 현재 $[l, r]$의 구간에 대한 답을 알고 있을 때, $[l, r+1]$의 구간에 대한 답을 구할 수 있습니다.
- ✓ $[l, r]$의 원소 중 a_{r+1}보다 작은 원소의 합과 그 개수, 큰 원소의 합과 그 개수를 구하는 것을 통해 가능합니다.
- ✓ 위 연산은 세그먼트 트리 또는 펜윅 트리를 통해 $O(\log N)$에 가능합니다.
- ✓ 같은 방식으로 $[l, r] \to [l-1, r]$, $[l, r] \to [l+1, r]$, $[l, r] \to [l, r-1]$ 또한 구할 수 있습니다.

H. 간단한 쿼리 문제

- ✓ 이제 어떤 정수 B에 대하여 쿼리를 $(\lfloor l/B \rfloor, r)$ 순으로 정렬합니다.
- ✓ 이 때 인접한 두 쿼리의 l의 차이는 $O(B)$ 입니다.
- ✓ r은 1부터 N까지 증가하는 것이 $O(N/B)$번 일어날 수 있습니다.
- ✓ l이 움직이는 횟수는 $O(QB)$, r이 움직이는 횟수는 $O(N^2/B)$ 입니다.
- ✓ $B = \sqrt{N}$으로 잡는다면 움직이는 횟수가 $O((N+Q)\sqrt{N})$번이고, 한번 움직일때 마다 $O(\log N)$인 연산을 하므로 이 문제는 $O((N+Q)\sqrt{N}\log N)$에 해결될 수 있습니다.

H. 간단한 트리 문제

- ✓ 서브테스크 3은 더 빠르게 이 문제를 해결해야 합니다.
- ✓ 서브테스크 2의 풀이에서 $O(\log N)$을 제거할 수 있으면 이 문제를 해결할 수 있습니다.
- ✓ $[l, r]$의 구간에서 어떤 값 v을 추가할 때 쿼리의 답의 변화를 $f(l, r, v)$라고 합시다.
- ✓ $f(l, r, v) = f(1, r, v) - f(1, l-1, v)$가 성립하고, 이 점을 활용할 수 있습니다.

H. 간단한 쿼리 문제

- ✓ $g(r, v) = f(1, r, v)$ 라고 하면 모스 알고리즘을 통해 구해야 하는 서로 다른 (r, v) 의 쌍의 갯수는 $O((N + Q)\sqrt{N})$ 개입니다.
- ✓ r 을 1 부터 N 까지 증가시키면서 필요한 v 에 대한 쿼리를 전처리로 구할 수 있습니다.
- ✓ 이는 sqrt decomposition 을 통해 $O(\sqrt{N})$ 에 업데이트, $O(1)$ 에 g 에 대한 쿼리를 구할 수 있습니다.
- ✓ 이 알고리즘을 sweepline mo's algorithm 이라고 하고, 전체 문제를 $O((N + Q)\sqrt{N})$ 에 해결할 수 있습니다.

I. 트리 위의 세 사람

`tree`, `small-to-large`, `fft/ntt`
출제진 의도 – **Challenging**

- ✓ 제출 2번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: -, -분
- ✓ 출제자: **김동현(kdh9949)**

- ✓ A, B, C 의 공통 LCA가 되는 D 를 고정하고, D 가 루트인 서브트리를 봅시다.
- ✓ D 의 각 자식에 대해, 밑에 달려 있는 트리를 일자로 만들어 줄 수 있습니다.
  같은 깊이에 해당하는 후손 노드들의 가중치를 모두 하나로 합쳐 주면 됩니다.
- ✓ 노드 X 에 대해, 높이 h_X 를 (X 에서 가장 멀리 떨어진 후손까지 거리 + 1)로 정의합시다.
- ✓ 결국 D 의 각 자식 U 에 대해 필요한 정보는 길이 h_U 의 가중치 배열입니다.

- ✓ 풀이에 쓰이는 핵심적인 Lemma가 하나 있습니다.
  Lemma. 정점이 N 개인 트리의 모든 노드에 대해, (두 번째로 높은 자식의 높이) + (세 번째로 높은 자식의 높이) + ...를 모두 더한 값은 N 이하이다.
  - 조금 더 강한 명제인 "$N-$(루트의 높이) 이하이다"를 간단한 귀납법으로 증명할 수 있습니다.
- ✓ 이 사실을 이용하면, 트리에서 DFS를 수행하면서 모든 노드에 대해 가중치의 배열을 $O(N)$ 시간복잡도로 계산하면서 문제를 해결할 수 있습니다.
- ✓ Small-to-large의 요령으로, 가장 높이가 높은 자식의 배열에 나머지 자식들의 배열 값을 더해준 뒤 마지막에 루트에 해당하는 원소를 추가해 주면 됩니다.

- ✓ 가중치 배열을 계산하는 건 좋은데, 그래서 문제의 답은 어떻게 빨리 구할 수 있을까요?
- ✓ 자식들을 높이가 높은 것부터 순서대로 놓았다고 했을 때, 자식 중 세 개를 골라서 깊이 합이 K 가 되도록 후손을 각각 고르면 됩니다.
- ✓ 고른 세 자식 중 가운데 자식 M 을 고정해 봅시다. 배열 세 개를 다음과 같이 정의합니다.
  - $pf[i]$: M 앞에 있는 자식들 밑에 달린 깊이 i 인 후손 가중치 합
  - $cur[i]$: M 밑에 달린 깊이 i 인 후손 가중치 합
  - $sf[i]$: M 뒤에 있는 자식들 밑에 달린 깊이 i 인 후손 가중치 합

- ✓ pf, sf 배열을 각 M에 대해 계산해야 하는데, 다음과 같이 하면 됩니다.
  - 자식들을 뒤쪽부터 보면서 sf 배열을 미리 전처리를 해 둡니다.
  - 이제 M을 앞쪽부터 순서대로 처리합시다. pf 배열은 첫 번째로 높은 자식의 가중치 배열을 갱신하면서 실시간으로 만들 수 있습니다. sf 배열은 방금 전처리한 것을 가져다 쓰면 됩니다.
- ✓ 이렇게 해도 되는 이유는 M은 아무리 높아야 "두 번째로 높은 자식"이기 때문입니다.
  - sf 배열을 구할 때 첫 번째로 높은 자식은 안 봐도 됩니다.
  - pf 배열을 구축하는 과정 역시 두 번째로 높은 자식부터 시작하게 됩니다.
  - 모든 과정을 $O((\text{두 번째로 높은 자식의 높이}) + (\text{세 번째로 높은 자식의 높이}) + \ldots)$에 수행할 수 있습니다. Lemma에 의해, 처리 시간이 총 $O(N)$입니다.

- ✓ 이제 답을 진짜 구해 봅시다.
- ✓ 가운데 자식이 M 일 때 답에 더해지는 값은 $\sum_{i=1}^{h_M}(cur[i] \cdot \sum_{j=1}^{h_M}(sf[j] \cdot pf[K-i-j]))$ 입니다.
  (정의되는 인덱스를 벗어난 값은 0 이라고 합시다)
- ✓ Lemma에 따르면 제일 바깥쪽 합은 일일이 하나씩 계산해도 그 횟수가 총 $O(N)$ 입니다.
- ✓ $t[x] = \sum sf[j] \cdot pf[x-j]$ 를 계산해 두면, 그 뒤는 총 $O(N)$ 만에 된다는 뜻입니다.
- ✓ 배열에서 각각 필요한 인덱스의 범위가 $2h_M$ 이하이므로, 역시 Lemma에 의해 convolution을 수행하는 총 배열 길이 합이 $O(N)$ 입니다.
- ✓ mod가 $998\,244\,353$ 이므로, 적절한 NTT 구현체를 사용하면 $O(NlogN)$ 에 문제를 해결할 수 있습니다.