20672 대세는 바이러스야

연속된 일부분이란?

트리의 루트를 1로 잡습니다. 풀이에서는 몬스터의 유전자 $g[i]$를 대신 $dna[i]$라고 표기하겠습니다.

몬스터가 연속된 일부분을 차지해야 한다는 점이 막막해 보입니다. 이 "연속된 일부분"의 성질을 몇 개 생각해 봅시다.

- 깊이가 가장 얕은 정점은 유일합니다. 이를 연속된 일부분의 "루트"라고 부릅시다.
- 따라서, 연속된 일부분은 특정 루트에서 시작해서 아래로 뻗어나가는 식으로 만들 수 있습니다.
- 만약 연속된 일부분이 밑에 있는 정점 v로 뻗어내렸다면, v의 자손으로도 뻗어내릴 수 있습니다. 뻗어내리지 않았다면, v의 자손으로도 뻗어내릴 수 없습니다.

DP 정의

루트가 아닌 정점 v와 치트키 g에 대해 $dpY[v][g]$와 $dpN[v][g]$를 다음과 같이 정의합니다.

- v의 서브트리를 v 빼고 모두 제거했다고 가정합시다.
- 출발지가 v라고 합시다.
- $dpY[v][g]$는, v에 몬스터가 있으면서, 치트키가 g인 경우의 수입니다.
- $dpN[v][g]$는, 몬스터가 (보스 제외) 한 마리 이상 있지만 v에는 없으면서, 치트키가 g인 경우의 수입니다. 이 경우 v의 서브트리에는 더 이상 연속된 일부분이 뻗어내릴 수 없습니다.

출발지 v에 대해, 모든 g에 대한 $g(dpY[v][g] + dpN[v][g])$의 합을 구하면 우리가 원하는 치트키 합을 얻습니다.

공간 복잡도가 너무 크지 않아요?

언뜻 보면 공간 복잡도가 $O(N^2)$인 것 같지만, 사실 가능한 치트키는 그 만큼 많지 않습니다. 치트키는 $dna[1]$의 약수이기 때문입니다. 10^9 이하의 자연수 중 약수가 가장 많은 수는 735,134,400으로, 약수가 1,344개입니다. (이런 수를 highly composite number라고 부릅니다.) 따라서 dpY, dpN을 map으로 바꿔 공간 복잡도를 줄일 수 있습니다. 다만 아직 map으로 바꾸지는 맙시다.

이제부터 D를 약수의 최대 개수인 1,344로 둡시다.

DP 식

이제 DP 상태 전이를 생각해 봅시다. 어떤 정점 v에 대해, $dpY[v][g]$와 $dpN[v][g]$를 모두 계산했고, v의 자식이 $a, b_1, \cdots, b_k$라고 합시다.

- a는 연속된 일부분의 루트가 될 수 있습니다. 이때 $dpY[a][gcd(dna[1], dna[a])]$가 1만큼 증가합니다.
- $b_1, \cdots, b_k$ 중 하나가 연속된 일부분의 루트가 될 수 있습니다. 이때 $dpN[a][dna[1]]$이 증가하긴 하는데... 일단 $X[a]$만큼 증가한다고 합시다.
- v가 연속된 일부분에 속하고, 이것이 a로 뻗어 내려갔다고 합시다. 이때 $dpY[a][gcd(g, dna[a])]$가 증가하긴 하는데... 일단 $Y[a]$만큼 증가한다고 합시다.
- v가 연속된 일부분에 속하지만, a로는 뻗어 내려가지 않았다고 합시다. 이때 $dpN[a][g]$가 증가합니다. $Z[a]$만큼 증가한다고 합시다.
- v가 연속된 일부분에 속하지 않는다고 합시다. 이때 $dpN[v][g]$가 $dpN[a][g]$만큼 증가합니다.

XYZ

X, Y, Z를 구해 봅시다.

- $RC[v]$를, "v를 루트로 하는 연속된 일부분의 개수"라고 합시다. 리프 노드로부터 거슬러 올라가는 트리 DP로 구할 수 있습니다.
- $AC[v]$를, "v의 서브트리에 완전히 포함되는 연속된 일부분의 개수"라고 합시다. 단, 연속된 일부분이 비어있어도 됩니다. 이것도 트리 DP와 위의 $RC[v]$를 가지고 구할 수 있습니다.
- $X[a] = AC[b_1] + \ldots + AC[b_k]$입니다. 사실 $X[b_1], \ldots, X[b_k]$도 다 구해야 되니까, $AC[a] + AC[b_1] + \ldots + AC[b_k]$를 구해놓고 하나씩 빼면 됩니다.
- $Y[a] = Z[a] = CC[b_1] * \ldots * CC[b_k]$입니다. X와는 달리 모든 CC의 곱을 구해놓고 하나씩 나누면 안 됩니다. 0으로 나누는 경우가 생기기 때문입니다. 그래서, 배열 LCC = $[CC[a], CC[b_1], \ldots, CC[b_k]]$라고 하고, $LLCC[i]$를 "처음부터 i번째까지 LCC값의 곱", $RLCC[i]$를 "i번째부터 끝까지 LCC값의 곱"으로 둡니다. 이걸 사용하면 Y와 Z를 구할 수 있습니다.

최적화

거의 다 왔습니다! 이제 자잘한 최적화를 거치면 끝납니다.

- dpY를 map이나 unordered_map으로 바꾸면 공간 복잡도가 $O(N \log(dna[1]))$로 줄어듭니다. 배열에서 첫 i개의 수의 gcd로 가능한 값은 $O(\log N)$개이기 때문입니다.
- 사실 dpN에서는 g가 전혀 중요하지 않습니다. $dpN[v]$를 "모든 g에 대해 $g \cdots dpN[v][g]$의 합"으로 두면 공간 복잡도가 $O(N)$으로 줄어듭니다.

- gcd를 $O(ND)$번이나 구하고 싶지는 않습니다. 상수 시간으로 줄여봅시다. 우선, 각각의 $dna[i]$를 $gcd(dna[1], dna[i])$로 바꿔도 답이 변하지 않습니다. 그리고 그렇게 하면 서로 다른 dna값이 최대 D개이므로, 서로 다른 gcd 연산은 $O(D^2)$개입니다. 이걸 모두 전처리합니다. 마지막으로, 모든 dna값을 좌표압축하듯이 1 이상 D 이하로 바꾸면, gcd 계산은 배열 값 받아오기로 대체됩니다.

이 모든 과정을 거치면 $O((N + D^2) \log(dna))$에 문제가 풀립니다.