"중등부 1번. 꿀 따기" 문제 풀이

작성자: 이종영

부분문제 1

각 장소마다 꿀을 딸 수 있는 양을 저장한 길이 N의 배열을 A라 하자. $1 \le i < j < k \le N$을 만족하는 모든 (i, j, k) 쌍에 대해 i, j, k중 하나의 장소가 벌통이고 나머지 두 장소가 벌이 있는 장소인 경우를 고려한다. 각 벌의 행동을 $O(N)$에 구현할 수 있으므로, 시간복잡도는 $O(N^4)$이다.

부분문제 2

부분문제 1의 풀이에서, 부분합 배열 $S_i = A_1 + \cdots + A_i$를 이용하면 각 경우에 대해 딴 꿀의 양을 $O(1)$에 구할 수 있다. 시간복잡도는 $O(N^3)$이다.

부분문제 3

$1 \le i < j \le N$을 만족하는 모든 (i, j) 쌍에 대해 i와 j가 벌이 있는 장소인 경우를 고려한다.

$i + 1 < j$인 경우 i와 j 사이에 벌통이 위치할 수 있다. 이 경우 벌통만이 두 벌이 모두 꿀을 딸 수 있는 장소이므로, 벌통에서 꿀을 딸 수 있는 양이 최대인 것이 최적이다. 즉, $i < k < j$인 k들 중 A_k의 최댓값을 구해야 한다. 이는 고정된 i에 대해 j를 증가시키며 함께 관리할 수 있다.

$1 < i$인 경우 i 왼쪽에 벌통이 위치할 수 있다. i 및 i의 오른쪽에서 따는 꿀의 양은 i와 j가 고정됨에 따라 고정되고, i의 왼쪽에서는 벌통의 위치까지의 각 장소에서 두 벌이 모두 꿀을 따게 된다. i 왼쪽의 모든 장소에서 꿀을 따는 것이 최적이므로 $p = 1$임을 알 수 있다. 즉, 벌통은 항상 1에 위치한다.

$j < N$인 경우 j 오른쪽에 벌통이 위치할 수 있다. 앞의 경우와 비슷하게 벌통은 항상 N에 위치함을 알 수 있다.

시간복잡도는 $O(N^2)$이다.

부분문제 4

$1 \le i < j \le N$인 i, j에 대해 벌이 i와 j에 위치하고, $1 \le k \le N$인 k에 대해 벌통이 k에 위치한다고 하자. $(i \ne k, j \ne k)$

$i < k < j$인 경우 i와 j 사이의 모든 장소에서 한 마리의 벌이 꿀을 따게 된다. 단, 예외적으로 벌통의 위치 k에서는 두 벌이 모두 꿀을 따게 된다. 따라서 고정된 k에 대해 $i = 1, j = N$인 경우가 최적이므로 각 k에 대해 $O(1)$에 구할 수 있다.

$i < j < k$인 경우 부분문제 2의 풀이와 같은 이유로 최적의 경우 $k = N$이다. j를 고정한다면 j 및 j의 오른쪽에서 따는 꿀의 양은 고정되고, j의 왼쪽에서는 i와 j 사이의 각 장소에서 한 마리의 벌이 꿀을 따게 된다. 따라서 $i = 1$인 경우가 최적이므로 각 j에 대해 $O(1)$에 구할 수 있다.

$k < i < j$인 경우 앞의 경우와 비슷하게 고정된 i에 대해 $k = 1, j = N$인 경우가 최적이므로 각 i에 대해 $O(1)$에 구할 수 있다.

각 경우를 $O(N)$에 해결할 수 있으므로 시간복잡도는 $O(N)$이다.

"중등부 2번. 두 개의 팀" 문제 풀이

작성자: 김준원

문제를 정확히 이해해야 한다.

- 팀장을 고르면, 그 팀장이 고른 팀원들은 모두 팀장 밑에 있고, 연결되어 있어야 한다는 것
- 팀원들의 실력의 합(팀의 점수)이 가장 크도록 골라야만 한다는 것 (팀원들 중 일부를 나머지 한 팀에게 양보하는 선택을 할 수 없다)
- 팀장은 자기 자신을 무조건 선택해야 한다는 것
- 무조건 두 개의 팀을 골라야 한다는 것 (팀 하나를 고르고 나서, 남은 사원들의 실력이 모두 음수라고 하더라도, 나머지 팀 하나를 구성해야만 한다)

에 유의하라.

부분문제 1

위 모든 조건을 정확히 이해했다면, 부분문제 1을 깊이 우선 탐색으로 해결할 수 있다. 부분문제 1에서는 모든 사원의 실력이 양수이다. 따라서 팀장을 골랐다면, 팀장은 팀장 밑에 있는 모든 사원(즉, 팀장을 루트로 하는 부트리에 있는 모든 정점)을 팀원으로 고를 수밖에 없다. 무조건 더 많이 고르면 고를수록 팀의 점수가 높아지기 때문이다.

따라서, 두 명의 팀장은 서로 트리에서 조상/자손 관계가 될 수 없다. 한 팀장이 다른 팀장을 무조건 팀원으로 받아들여야만 하는 일이 생기기 때문이다. 대신, 두 팀장은 한 *부모 정점* 아래에 있는 두 '자매' 정점이 되는 것이 최적이다.

먼저 깊이 우선 탐색을 통해, 각 정점 x에 대해 x를 루트로 하는 부트리에 있는 정점들의 실력의 합 $sum[x]$을 계산해준다. 그리고, 다시금 각 정점 x을 순서대로 보면서, x가 위에서 언급한 *부모 정점*이 되는 경우를 고려한다. x의 자녀 정점이 둘 이상 있는 경우에, **x의 자녀 정점들 중 가장 sum 값이 큰 두 개의 정점**을 선택하는 것이 최적이다.

모든 정점에 대해 위 최적의 경우를 계산하면, 계산한 값들의 최댓값이 전체 문제의 답이 된다.

부분문제 2, 3

부분문제 2, 3에서 주어지는 트리는 직선이다. 편의상 루트 정점을 가장 왼쪽에 두고, 리프 정점을 가장 오른쪽에 두자. 이제, 하나의 팀은 직선 위에서 연속된 구간('사원 x부터 사원 y까지'와 같은 형태)으로 나타나고, 팀장은 구간에서 가장 왼쪽에 있는 정점이 된다. 문제를 풀기 위해서, 부분합 기법을 이용해 다음을 계산한다.

$sum[x]$: 사원 1부터 사원 x까지 x명의 사원의 실력의 합

(부분문제 1에서 정의한 sum과는 무관함에 유의하라.)

이 때, 사원 x부터 사원 y까지로 이루어진 팀의 점수는 $sum[y] - sum[x-1]$로 나타나게 된다. 그리고, 팀장이 사원 x일 때 팀의 점수가 가능한 한 가장 커야 하므로, 사원 y는 $x < y$이면서 $sum[y]$가 최대인 사원이 되어야 한다. 각 사원 x에 대해 사원 x가 팀장이 되는 경우에 점수가 최대가 되게 하는 사원 y를 찾을 수 있다.

이는 단순히 계산하면 $O(N^2)$, 슬라이딩 윈도우 기법을 사용하여 계산하면 $O(N)$에 계산할 수 있다. 위 값을 계산할 때, 각 사원 x가 팀장이 되었을 때 어떤 정점 y를 고르는 게 최적인지도 함께 기록을 해 두자. 최적인 정점 y가 여러 개 있다면, 그러한 정점들 중 가장 정점 번호가 작은 정점을 골라야 팀의 크기가 최소화되고, 두 팀의 팀원이 겹치지 않게 하기 좋을 것이다.

첫번째 팀의 팀장 x는 사원 1부터 사원 N까지 어떤 사원이든 될 수 있다. 이를 반복문으로 순회하면서, 사원 x가 팀장일 경우에 두 번째 팀의 팀장이 될 수 있는 사원들을 차례로 순회한다. 이를 단순히 매번 순회하면 $O(N^2)$, x가 증가할 때 y 또한 증가한다는 사실을 이용해 슬라이딩 윈도우 기법을 적용하면 $O(N)$에 해결할 수 있다.

부분문제 4, 5

부분문제 2, 3의 풀이의 확장이다. 정점 x를 팀장으로 골랐을 때, '어디까지 골라야 할까?'를 계산해야 한다. 흥미로운 사실은, '어디까지 골라야 할까?'를 재귀적으로 계산할 수 있다는 사실이다.

정점 x가 팀장이라고 하자. 정점 x는 반드시 이 팀의 팀원에 포함되어야 한다. 그리고, 정점 x의 자녀인 정점 y를 잠깐만 생각해보자. 만약 정점 y가 팀장일 때에 팀이 얻을 수 있는 최대 점수, 그리고 이 점수를 얻을 수 있게 되는 팀원이 누구인지 알고 있다고 하자.

- 최대 점수가 양수라면, 정점 x가 팀장일 때에도 팀원 목록에 **정점 y와 정점 y를 팀장으로 했을 때의 팀원들**을 모두 포함시키는 것이 이득이다.
- 최대 점수가 음수라면, 그냥 정점 y와 그 아래에 있는 정점들은 팀원이 되지 않는 것이 이득이다.
- 최대 점수가 0이라면, 이들을 포함시키든 포함시키지 않든 점수에 변화가 없다. 팀의 크기는 작을수록 좋으므로 제외하는 것이 이득이다.

이와 같은 규칙으로, 정점 x에서 시작하는 DFS를 수행하면 x가 팀장일 때에 얻을 수 있는 최대 점수와 팀원 목록을 알 수 있다. 이를 이용하면 $O(N^3)$ 또는 $O(N^2)$의 시간 복잡도에 문제를 풀 수 있다.

부분문제 6

위 풀이의 가장 큰 문제점은 '매번 팀원들의 목록을 대조해서 겹치는 팀원이 있는지 확인해야 한다'는 것이다. 그 대신 DFS를 한 번 똑똑하게 수행하면 **모든 것을 알 수 있게 된다.**

다음과 같은 값들을 DFS 내부에서 재귀적으로 계산해 보자.

1. $a[x]$: 정점 x를 팀장으로 했을 때 팀이 얻을 수 있는 최대 점수
2. $b[x]$: 정점 x 또는 그 아래의 어딘가에 있는 정점을 팀장으로 했을 때 팀이 얻을 수 있는 최대 점수
3. $c[x]$: 정점 x 또는 그 아래의 어딘가에 있는 정점들 중 [정점 x를 팀장으로 했을 때 최적의 팀에서 고른 정점들을 제외한 정점] 중 하나를 팀장으로 했을 때, 팀이 얻을 수 있는 최대 점수

이 때 다음과 같은 점화식을 얻게 된다. 정점 x의 자녀 정점들의 집합을 C_x라고 하자.

1. $a[x]$: 사원 x의 실력 + [$y \in C_x$에 대해 $\max(0, a[y])$의 합]

2. $b[x]$: $y \in C_x$에 대해 $b[y]$의 최댓값, 또는 사원 x의 실력이 더 크다면 사원 x의 실력

3. $c[x]$: $y \in C_x$에 대해 [$a[y]$가 양수라면 $b[y]$, 아니라면 $c[y]$] 의 합

이를 이용해 우리가 원하는 최댓값을 구할 수 있다. 두 팀의 팀장은 1. 서로 조상-자손 관계일 수도 있고, 2. 서로 조상-자손 관계가 아닐 수도 있다.

1. 이 때 조상 역할인 팀장의 번호를 x라고 하자. 문제의 답은 $a[x] + c[x]$이다.

2. 부분문제 1의 발상과 같다. 이 때 두 팀장의 최소 공통 조상(LCA)이 x라고 하자. 문제의 답은, x의 자녀들 y에 대해 $b[y]$들을 모두 모았을 때, 가장 큰 두 값의 합이다.

DFS를 통해서 각 정점 x에 대해서 $a[x]$, $b[x]$, $c[x]$를 계산한 뒤, 각 정점 x를 순서대로 돌면서 위 1.과 2.에 해당하는 두 개의 값을 차례로 구할 수 있다. 답은 구한 모든 값의 최댓값이다.