

“맛집 추천” 문제 풀이

작성자: 윤창기

부분문제 1

부분문제 1은 트리가 직선인 특수한 경우이다. 이 경우, R_i 들을 모두 구간으로 생각할 수 있다.

$1 \leq i \leq N$ 에 대해 D_i 를 “선택한 모든 구간들이 i 원쪽으로 넘어가지 않으면서 서로소일 때 (서로 겹치지 않을 때) 가능한 최대 만족도”라고 정의하자. 문제의 답은 D_1 이 됨을 쉽게 알 수 있다. 모든 D_i 의 값을 동적 계획법으로 구해보자.

언급한 조건에서 i 번 정점은 구간으로 덮지 않거나, 어떤 $[i, j]$ 꼴의 구간을 사용하여 덮어야 한다.

- i 번 정점을 구간으로 덮지 않는 경우 얻을 수 있는 최대 만족도는 D_{i+1} 이다.
- 어떤 구간 $[i, j]$ 를 사용했을 때 g 의 만족도를 얻을 수 있다면, 이 구간을 사용했을 때 가능한 최대의 만족도는 $D_{j+1} + g$ 이다.

이 경우를 모두 고려하면 $O(M + N)$ 시간에 문제를 해결할 수 있다.

부분문제 2

부분문제 2는 맛집의 개수 M 이 20 이하로 매우 작은 경우이다. 따라서 가능한 모든 맛집의 집합을 탐색하면서 문제의 조건을 만족하는지 판별하는 방법으로 문제를 해결하기에 충분하다.

선택된 맛집들에 대해, 너비 우선 탐색 등을 이용하여 거리가 d_i 인 점들을 모두 구할 수 있다. 이 중 겹치는 점이 있는지 판단하는 방법으로 $O((N + M)2^M)$ 시간에 문제를 해결할 수 있다.

혹은, 두 맛집 $i \neq j$ 가 동시에 선택되지 못할 조건을 $d(i, j) \leq d_i + d_j$ 로 나타낼 수 있으므로, 모든 맛집에 대해 $d(i, j)$ 를 $O(MN)$ 또는 $O((M^2 + N) \log N)$ 시간에 구하는 방법으로 $O(N \log N + M^2 2^M)$ 시간에 문제를 해결할 수도 있다.

부분문제 3

부분문제 3은 $M, N \leq 2000$ 인 경우이다. 따라서 $O(MN)$ 시간에 문제를 해결하면 충분하다.

임의로 1번 정점을 루트로 잡고, 부분문제 1의 관찰을 일반화하여 D_v 를 “ v 를 루트로 하는 서브트리에서, 선택한 모든 R_i 들이 v 의 서브트리에 포함되며 서로소일 때 가능한 최대 만족도”로 정의해보자. 문제의 답이 D_1 임을 확인할 수 있다. 마찬가지로 경우를 분할하여 생각해보면

- v 가 어떤 R_i 들로도 덮이지 않는 경우, 가능한 최대 만족도는 v 의 모든 자식 u 에 대해 D_u 의 합이다.
- v 가 R_i 로 덮이는 경우, R_i 는 v 를 포함하면서 v 의 서브트리 밖을 나가면 안 된다. v 의 서브트리에서 R_i 를 지우면, 이 서브트리가 여러 개로 분할되며, 각 서브트리 조각의 루트는 $d(c_i, x) = d_i + 1$ 을 만족하는 x 이다. 가능한 최대 만족도는 v 의 서브트리에 속한 점들 중 $d(c_i, x) = d_i + 1$ 을 만족하는 x 에 대해 D_x 를 합한 값이다.

이 때 v 가 R_i 로 덮이면서, v 의 서브트리 밖이 R_i 로 덮이지 않으려면, v 는 R_i 가 덮는 점들 중 가장 루트에 가까워야 한다. 따라서 각 R_i 를 고려하는 시점은 v 가 c_i 의 d_i 번째 조상일 때로 유일하며, 이는 DFS나 자료구조를 활용하여 빠른 시간 안에 구할 수 있다.

$d(c_i, x) = d_i + 1$ 을 만족하는 정점의 개수가 $O(N)$ 개이므로, 전체 시간복잡도 $O(MN)$ 에 문제를 해결할 수 있다.

부분문제 4

부분문제 4는 $N \leq 2000$ 인 경우이다. $(c_i, d_i) = (c_j, d_j)$ 인 두 맛집 i, j 가 있다면, 둘 중 만족도가 큰 것만 고려해도 관계가 없다.

$\mathcal{R}(c, t)$ 를 $d(c, x) = t$ 를 만족하는 정점 x 의 개수라고 하자. 부분문제 3의 동적 계획법을 채우기 위해 필요한 시간복잡도는 $O(N + M + \sum_i \mathcal{R}(c_i, d_i + 1))$ 임을 알 수 있다.

이 때, $\sum_{d=1}^{N-1} \mathcal{R}(c, d) = N - 1$ 이고 $(c_i, d_i + 1)$ 이 모두 다르므로,

$$\sum_i \mathcal{R}(c_i, d_i + 1) \leq \sum_{c=1}^N \sum_{d=1}^{N-1} \mathcal{R}(c, d) = \sum_{c=1}^N (N - 1) = N(N - 1)$$

의 부등식이 성립한다. 따라서 전체 시간 $O(N^2 + M)$ 에 문제를 해결할 수 있다.

부분문제 5

부분문제 5는 도시가 완전 이진 트리 모양인 경우이다. 역시 부분문제 3의 동적 계획법을 개선하여 문제를 해결할 수 있다.

어떤 정점 z 의 부모를 $p(z)$ 로 쓰자. $d(c, x) = d + 1$ 을 만족하는 점 x 들은 다음의 겹치지 않는 조건들 중 하나를 만족한다. v 는 $R(c, d)$ 가 덮는 가장 루트에 가까운 점이므로, $d(c, v) = d$ 임에 주목하자.

- c 의 서브트리에 속하고, 깊이가 $(\text{dep}_c: c\text{의 깊이}) + d + 1$ 이다.
- $p(c)$ 의 (c 와는 다른) 서브트리에 속하고, 깊이가 $\text{dep}_{p(c)} + d$ 이다.
⋮
- $p^j(c)$ 의 ($p^{j-1}(c)$ 와는 다른) 서브트리에 속하고, 깊이가 $\text{dep}_{p^j(c)} + d + 1 - j$ 이다.
⋮
- $v = p^d(c)$ 의 ($p^{d-1}(c)$ 와는 다른) 서브트리에 속하고, 깊이가 $\text{dep}_v + 1$ 이다.
- $p(v)$ 이다. D_v 를 구하는 데는 기여하지 않으므로 무시해도 좋다.

따라서, $F(v, d)$ 를 v 의 서브트리 중 v 와의 거리가 d 인 정점 x 에 대한 D_x 의 합으로 정의하면, 위의 각 경우에 해당하는 D_x 의 합을 $F(v, d)$ 들을 이용해 나타낼 수 있다.

- c 의 서브트리에 속하고, 깊이가 $\text{dep}_c + d + 1$ 이다.
 $\rightarrow F(c, d + 1)$
- $p^j(c)$ 의 ($p^{j-1}(c)$ 와는 다른) 서브트리에 속하고, 깊이가 $\text{dep}_c + d + 1 - j$ 이다. ($1 \leq j \leq k$)
 $\rightarrow F(p^j(c), d + 1 - j) - F(p^{j-1}(c), d + 1 - j)$

따라서 $F(v, d)$ 를 빠른 시간 안에 관리할 수 있으면 문제를 해결할 수 있다. 이 부분문제의 경우 $d \leq \log N$ 이므로, 매번 $O(\log N)$ 의 $F(\cdot, \cdot)$ 값만 사용하거나 수정하여 동적계획법을 수행할 수 있다. 따라서 시간복잡도 $O(M \log N)$ 에 문제를 해결할 수 있다.

부분문제 6

부분문제 6은 트리에 차수 3 이상인 정점이 하나뿐인 경우이다. 일반성을 잃지 않고 그 정점을 1번 점이라 두자. 1번 점을 루트로 하는 트리를 생각했을 때, 트리의 특성에 의해 루트에 연결된 모든 서브트리가 직선 모양이 되는 것을 알 수 있다. 직선 e 에 속한 정점들을 루트에서부터 가까운 순서로 $e(1), e(2), \dots$ 라고 쓰자.

따라서 부분문제 1의 간단한 동적 계획법을 각 직선에 적용하여 $F(1, d) = \sum_e D_{e(d)}$ 를 선형 시간에 구할 수 있다. D_1 을 구하기 위해, 1번 정점을 R_i 로 덮는 상황을 생각해 보자. $b = d(c_i, 1)$ 이라고 두면, R_i 는 c_i 가 속한 직선 e 에서는 $e(d_i + b)$ 까지, 다른 직선 e' 에서는 $e'(d_i - b)$ 까지를 덮는다. 따라서 얻을 수 있는 만족도의 최댓값은 아래와 같은 식으로 나타난다.

$$\sum_{e' \neq e} D_{e'(d_i - b + 1)} + D_{e(d_i + b + 1)} + g_i = F(1, d_i - b + 1) - D_{e(d_i - b + 1)} + D_{e(d_i + b + 1)} + g_i$$

따라서, 시간복잡도 $O(M + N)$ 에 문제를 해결할 수 있다.

부분문제 7

부분문제 5의 풀이는 일반적인 경우 d 가 $N - 1$ 까지 커질 수 있으므로 적용하기에 어려움이 있다. 이를 최적화하기 위해 트리의 센트로이드 분할 (Centroid decomposition)을 생각하자.

정점 c 가 트리 T 의 센트로이드(centroid)라는 것은, c 를 루트로 했을 때 가장 큰 서브트리의 크기가 전체 정점의 절반을 넘지 않는다는 것을 말한다. 한 트리에서 센트로이드는 1개 또는 2개 존재하며, 선형 시간에 찾을 수 있다. 반복적으로 서브트리의 센트로이드를 찾아서 트리를 분할하는 과정을 센트로이드 분할이라고 하며, 센트로이드의 정의상 총 $O(N \log N)$ 시간에 수행할 수 있다.

편의상 센트로이드 분할 과정을 트리 형태의 도식으로 나타낼 수도 있다. 즉, 전체 트리에서 센트로이드 c 를 기준으로 트리를 분할하고, 한 서브트리의 센트로이드 d 를 기준으로 서브트리를 분할하고, 다시 e 를 … 와 같은 방식으로 센트로이드 분할이 이루어졌다면 c 가 d 의 부모, d 가 e 의 부모, … 가 되도록 트리를 구성하는 방식이다. 이렇게 만든 도식을 ‘센트로이드 트리’ T^* 라고 부르자. T^* 는 항상 깊이가 $O(\log N)$ 인 좋은 성질을 갖는다. 이후, 부분문제 5의 관찰과 유사한 흐름으로 전체 문제를 해결할 수 있다.

T^* 에서 어떤 정점 z 의 부모를 $\pi(z)$ 로 쓰자. $d(c, x) = d + 1$ 을 만족하는 점 x 는 다음의 겹치지 않는 조건들 중 하나를 만족한다.

- c 의 서브트리에 속하고, $d(c, x) = d + 1$ 이다.
- T^* 에서 $\pi(c)$ 의 (c 와는 다른) 서브트리에 속하고, $d(\pi(c), x) + d(\pi(c), c) = d + 1$ 이다.
⋮
- $\pi^j(c)$ 의 ($\pi^{j-1}(c)$ 와는 다른) 서브트리에 속하고, $d(\pi^j(c), x) + d(\pi^j(c), c) = d + 1$ 이다.
⋮
- T^* 의 루트 (센트로이드) $r = \pi^k(c)$ 의 ($\pi^{k-1}(c)$ 와는 다른) 서브트리에 속하고, $d(r, x) + d(r, c) = d + 1$ 이다.

항상 $k \leq \log N$ 임에 주목하자. 이번에도 $F(v, d)$ 와 비슷한 값을 정의하여 각 경우를 닫힌 꼴로 나타낼 것이다. $F^*(v, d)$ 를 T^* 에서 v 의 서브트리 중 v 와의 거리가 d 인 정점 x 에 대한 D_x 의 합, $G^*(v, d)$ 를 T^* 에서 $\pi(v)$ 의 v 가 속한 서브트리 중 $\pi(v)$ 와의 거리가 d 인 정점 x 에 대한 D_x 의 합이라고 두면,

- c 의 서브트리에 속하고, $d(c, x) = d + 1$ 이다.

$$\rightarrow F^*(c, d + 1)$$

- $\pi^j(c)$ 의 ($\pi^{j-1}(c)$ 와는 다른) 서브트리에 속하고, $d(\pi^j(c), x) + d(\pi^j(c), c) = d + 1$ 이다. ($1 \leq j \leq k$)

$$\rightarrow F^*(\pi^j(c), d + 1 - d(\pi^j(c), c)) - G^*(\pi^{j-1}(c), d + 1 - d(\pi^j(c), c))$$

따라서 $F^*(v, d)$, $G^*(v, d)$ 의 값만 빠른 시간 안에 관리할 수 있다면 문제를 해결할 수 있다. 이 때, 센트로이드 분할 과정을 생각해보면 $F^*(v, d) > 0$ 혹은 $G^*(v, d) > 0$ 인 (v, d) 의 개수가 많아야 $O(N \log N)$ 개이다. 따라서 $F^*(v, \cdot)$, $G^*(v, \cdot)$ 를 배열로 관리하면 전체 문제를 $O((N + M) \log N)$ 에 해결할 수 있다.