"바이러스 (virus)" 문제 풀이

작성자: 구재현

부분문제 1

문제에서 바이러스가 **전파** 되는 과정을 최단 경로 문제로 해석하면 이후 풀이 이해와 구현에 도움이 된다. j 번 사람의 **영역** 을 $P[j]$ 에서 거리가 $D[j]$ 이하인 지점의 집합이라고 하자. i 번 사람과 j 번 사람 간의 전파의 매개체는, i 번 사람과 j 번 사람의 영역의 교집합이다. $cost(i, j)$ 를, 이 교집합에 속하는 점 v 중 $C[v]$ 의 최솟값이라고 하자 (없을 경우 ∞).

i 번 사람이 시간 t 에 감염되면, j 번 사람은 시간 $t + cost(i, j)$ 에 바이러스에 감염되어 있다. 각 사람을 정점으로 하고, i 번 사람에서 j 번 사람으로 가는 가중치 $cost(i, j)$ 의 간선을 가지는 그래프를 G 라고 하자. 바이러스가 전파되는 방식은, 0 번 정점에서 시작하는 최단 경로를 따른다.

고로 이 문제는 위에서 설명한 방식으로 그래프를 만든 후 Dijkstra's algorithm 이나 Floyd-Warshall algorithm 등을 사용하여 0 번 정점에서 각 정점에 도달하는 최단 경로를 계산하는 문제라고 생각할 수 있다. 그래프를 만들기 위해서는 $cost(i, j)$ 를 계산해야 하는데, 지문의 정의를 따라서 계산해 주면 된다. 시간 복잡도는 $O(N^2(N + M))$ 정도이다.

부분문제 2

부분문제 2에서는 그래프 크기가 크기 때문에 Floyd-Warshall algorithm은 사용할 수 없으며, $cost(i, j)$ 역시 단순하게 계산할 수 없다. $cost(i, j)$ 가 배열의 구간 최솟값 형태로 표현되기 때문에, 적절한 전처리를 사용하여 빠르게 해결할 수 있다. 위 방법을 사용해도 부분문제 2를 맞을 수 있지만, 이후 부분 점수로 확장이 편리한 다른 풀이를 소개한다.

바이러스가 전파되는 과정을 모델링할때, 지점 역시 감염될 수 있다고 생각하고, 사람에 대한 정점 뿐만이 아니라 지점에 대한 정점을 추가로 만들어주자. 만약 i 번 사람이 시간 t 에 감염되었다면, i 번 사람의 영역에 속하는 모든 지점 v 가 시간 $t + C[v]$ 에 감염된다고 하자. 어떠한 지점이 감염된다면, 이 지점을 영역으로 가지는 모든 사람이 감염된다는 것을 관찰할 수 있다.

이렇게 그래프를 구성할 경우 정점의 개수는 $O(N + M)$, 간선의 개수는 $O(NM)$ 개로 약간 증가하지만, 각 간선의 가중치를 계산하기 위해 추가적인 작업이 필요하지 않다. 이 그래프에서 Dijkstra's algorithm을 적용하면 된다.

부분문제 3

부분문제 3에서는 KOI 도시의 구조가 직선의 형태고, 각 사람의 영역은 이 직선의 구간을 이룬다. 이 점을 활용하여, 같은 최단 경로를 조금 더 효율적인 그래프 구성으로 계산한다.

부분문제 2에서 만든 그래프는 다음과 같은 두 종류의 정점과, 두 종류의 간선이 있다.

- 정점 p_i 는 i 번 사람에 대응
- 정점 l_j 는 j 번 지점에 대응
- p_i 에서, i 번 사람의 지점 구간 l_j ($j \in [L_i, R_i]$) 로 가는 가중치 $C[j]$ 의 간선
- i 번 사람의 지점 구간 l_j ($j \in [L_i, R_i]$) 에서, p_i 로 가는 가중치 0 의 간선

세 번째 종류의 간선은 끝점에 따라 가중치가 다르기 때문에 그래프 구성을 효율화하기 어렵다. 고로 다음과 같이 그래프 구성을 약간 바꾼다.

- 정점 p_i 는 i 번 사람에 대응
- 정점 in_j, out_j 는 j 번 지점에 대응
- p_i 에서, i 번 사람의 지점 구간 in_j ($j \in [L_i, R_i]$) 로 가는 가중치 0 의 간선
- i 번 사람의 지점 구간 out_j ($j \in [L_i, R_i]$) 에서, p_i 로 가는 가중치 0 의 간선
- in_j 에서 out_j 로 가는 가중치 $C[j]$ 의 간선

구간에 간선을 더하는 것과 구간 쿼리가 유사하다는 것에서 착안해, 세그먼트 트리 형태로 그래프의 크기를 줄이자. n 개의 지점에 대해 세그먼트 트리를 만들고, 각 트리의 노드에 대응되는 두 개의 정점을 만든다. 모든 간선의 가중치는 0 이다.

- 세그먼트 트리의 각 노드 n 에 대해서 $\textsc{InTree}_n, \textsc{OutTree}_n$ 이라는 정점을 만든다.
- 리프가 아닌 노드 n 에 대해, $\textsc{InTree}_n \to \textsc{InTree}_{2n}, \textsc{InTree}_n \to \textsc{InTree}_{2n+1}$ 의 형태로 간선을 만든다. $\textsc{OutTree}_n$ 에 대해서는 거꾸로 한다.
- 리프 노드 n 에 대해, 해당 노드에 대응되는 사람이 j 라고 하면, $\textsc{InTree}_n \to in_j, out_j \to \textsc{OutTree}_n$ 형태의 간선을 만든다.

이 세그먼트 트리는 $O(n)$ 개의 정점과 $O(n)$ 개의 간선을 가진다. 이 세그먼트 트리 구조를 사용하여, 그래프의 간선 수를 줄일 수 있다: p_i 에서, i 번 사람의 지점 구간 in_j ($j \in [L_i, R_i]$) 로 가는 가중치 0 의 간선을 만드는 대신, 세그먼트 트리에서 해당 구간에 대응되는 $O(\log n)$ 개의 노드에 대해, $\textsc{InTree}_n$ 으로 가는 가중치 0 의 간선을 만들면 된다. $\textsc{OutTree}_n$ 에 대해서도 동일하게 진행하면, 총 $O(n+m)$ 개의 정점과 $O(n + m \log n)$ 개의 간선을 가지는 그래프를 얻게 된다. 이 그래프에서 Dijkstra's algorithm을 사용하면 전체 문제를 $O(n \log n + m \log^2 n)$ 에 해결할 수 있다.

추가적으로, 가중치가 0 이 아닌 간선이 최대 n 개라는 점에 착안하여, Dijkstra's algorithm을 처리하는 과정에서 0 인 간선들이 존재할 때는 BFS로 처리하는 방법도 존재한다. 0-1 BFS와 비슷한 원리이며, 이 방법을 사용하면 전체 문제를 $O((n+m) \log n)$ 에 해결할 수도 있다. 실제 수행 시간은 대략 2배 정도 빨라지며, 이 최적화를 적용하지 않아도 만점을 받을 수 있게 시간 제한이 설정되어 있다.

부분문제 4

부분문제 1의 풀이를 트리에서 구현하면 된다. 트리에서 모든 점간 최단 경로를 계산해 두면 가능하다.

부분문제 5

부분문제 2의 풀이를 트리에서 구현하면 된다. $cost(i, j)$ 를 $O(1)$에 구하는 방식은 일반화가 (가능하지만) 다소 복잡하고, 이후 설명한 다른 풀이를 적용하는 것이 좋다.

부분문제 6

부분문제 3의 풀이를 트리에서 구현하기 위해서는, 트리에서 특정 정점과 거리가 D 이하인 정점들의 집합을 효율적으로 표현할 수 있어야 한다. 부분문제 3에서 사용한 세그먼트 트리로는 이것이 어렵지만, 대신 Centroid decomposition tree을 사용하면 이러한 표현이 가능하다.

트리의 Centroid decomposition을 할 때, 우리는 주어진 트리의 Centroid c 를 찾고, c 를 루트로 한 트리에 대해 적절한 전처리를 한 후, c 를 제거한 각 서브트리들에 대해 재귀적으로 문제를 해결한다. 트리의 각 정점 v 에 대해서, v 가 Centroid였을 시점에 v 를 포함한 트리를 v 의 **클러스터** 라고 하자. 또한, 각 정점 v 에 대해서 Centroid decomposition tree는, 트리에서 Centroid decomposition을 했을 때 특정 정점 v 와 거리가 D 이하인 정점들은 다음 집합의 합집합이다:

- v의 클러스터 상에서 v 와의 거리가 D 이하
- $p(v)$ 의 클러스터 상에서 $p(v)$ 와의 거리가 $D - dist(p(v), v)$ 이하
- $p^2(v)$ 의 클러스터 상에서 $p^2(v)$ 와의 거리가 $D - dist(p^2(v), v)$ 이하
- ...

각 클러스터 c에 대해, $\textsc{InDecomp}_{c,d}$ 는 c 의 클러스터 상에서 c 와의 거리가 d 이하인 점들의 in_v 로 가는 정점으로 정의된다. $\textsc{OutDecomp}_{c,d}$ 역시 유사하게 정의한다 (방법이 동일하니 여기서부터는 설명하지 않는다.) 이 때 다음과 같이 간선을 이어주면 된다:

- $\textsc{InDecomp}_{c,d} \to \textsc{InDecomp}_{c,d-1}$
- c 의 클러스터 상에서 거리가 정확히 d 인 모든 정점 v 에 대해 $\textsc{InDecomp}_{c,d} \to in_v$

이를 통해 총 $O(n \log n + m)$ 개의 정점과 $O((n+m)\log n)$ 개의 간선을 가지는 그래프를 얻게 된다. 이 그래프에서 Dijkstra's algorithm을 사용하면 전체 문제를 $O((n+m)\log^2 n)$ 에 해결할 수 있다. 부분문제 3과 동일한 최적화를 적용하면 이를 $O((n+m)\log n)$ 에 해결할 수도 있다.