C3. Two Trains

[21500050 김기훈, 21701065홍원표]

1. 문제 분석

N개의 도시와 그 도시간의 M개의 관계가 주어진다(Vertex=N, Edge=M).
Train alpha, Train Beta가 존재하며, 두기차의 두 도시간 이동에 소요되는 연료량이 주어진다. 두 기차는 연결되어서 갈 수 있는데, 이때 연결된 기차는 임의의 중간지점 V_x에서 V_N으로 이동하게 된다. 이 연결된 기차는 Weight가 다르기 때문에 기존의 두 기차와 별개의 기차(Train U)라고 생각해볼 수 있다. 다시 말해, Train U는 V_N에서 V_x로 이동한다고 반대로 생각해 볼 수 있다. 이렇게 보면 p, q, r은 각 열차의 단위거리당 연료소모량이라고 볼 수 있다. 즉, Vertex 1에서 출발하는 Train Alpha, Vertex 2에서 출발하는 Train Beta 과 Vertex N에서 출발하는 Train U 이 세 Train이 어느 Vertex에서 만나야 가장 짧은 Weight이 걸리는가라는 문제로 **재해석**이 가능하다.

1. 문제 풀이

문제에서 도시간의 관계가 주어져 있으므로 Adjacency List 형태로 구현하였다. 각 도시간 방향성에 대한 조건이 없기 때문에 Undirected Graph 형태로 입력을 받아 저장하였다. 문제의 이해는 어렵지 않았으나, 두 기차가 어디서 만나서 가는 것이 비용을 최소화 하는지를 결정하는 변수(기차 Alpha, Beta 각각의 연료소모량, 그래프의 연결관계)가 너무 많다는 것을 금방 알 수 있었다. 따라서 모든 가능한 경우를 탐색하는 Exhaustive Search를 사용해야겠다는 생각을 하였고, 각 점에서 떨어진 거리를 알 수 있는 탐색 방법인 BFS를 사용하였다. BFS의 결과를 각각의 Vector에 (City Index, Number of Steps)의 pair로 저장하였다. 두 기차는 어느 점에서든 만나서 갈 수 있으므로 **도착지점에서 각 도시로의 거리**도 생각해 주어야 했다. 출발점이 다른 같은 의미의 3개의 Vector를 사용해 $\mathbf{cost_{alpha} * d^{alpha}_i + cost_{beta} * d^{beta}_i + cost_{dest} * d^{end}_i}$ 를 모든 점에 대해서 계산한 후, 최소값을 구했다.

1. 문제 풀이 분석 (Vertex의 개수를 N이라고 가정)

BFS를 3번 이용하였으므로, O(N+M).
따라서 Time Complexity는 O(N+M).

1. Application

N+1 명의 친구가 M개의 연결로 이루어진 지하철 노선 상에서 교통비의 합을 가장 적게하여 만나고 싶을 때, 그들이 만날 역을 찾을 수 있다.
(한 역에서 다른 역까지의 거리는 동일, 그리고 지하철 요금은 지나는 역의 개수에 비례하여 증가한다고 가정 ← 미국 지하철의 Case)

1. Limitation(Optional) + 최적화 + 추가내용