Parity Constraint Shortest Path 풀이

> 2019년에 연세대학교 최적화 연구실에서는 갑자기 특정 문제에 Parity Constraint(홀짝 제약)을 넣은 상태로 문제를 푸는 것이 유행한 적이 있었고, 졸업하신 김강산 선배도 Parity Constraint 관련 논문을 작성하셨다. 이를 기념하기 위해서 국렬이는 연세대학교 프로그래밍 경진대회에 Parity Constraint 문제를 내기로 하였다. 정점이 N개, 비용이 있는 무방향 간선이 M개 있는 그래프가 주어진다. 모든 정점에는 1부터 N까지 번호가 매겨져 있다. 임의의 정점을 선택했을 때 다른 정점으로 가는 경로는 항상 존재하며, 서로 다른 두 정점 사이에는 최대 한 개의 간선이 존재한다. 정점 1에서 출발해서 다른 정점으로 이동하려고 한다. 이미 지나갔던 정점이나 간선도 다시 지나갈 수 있으며, 지나갈 때마다 비용이 추가된다. 경로를 구성하는 간선의 비용의 합을 편의상 '경로의 비용'이라 정의하자. 경로의 비용이 홀수면 홀수 경로, 짝수면 짝수 경로라고 부르자. 각 정점으로 이동할 때 비용이 최소가 되는 홀수 경로의 비용과 짝수 경로의 비용을 구하려고 한다.

1번 정점에서 시작해서 나머지 모든 정점으로의 홀수 최단 경로와 짝수 최단 경로를 구하는 문제입니다.

그래프의 정점에 추가적인 정보를 넣어서 정의합니다. 같은 정점이더라도 홀수 경로냐 짝수 경로냐에 따라 최단 거리가 달라질 수 있으므로 d_{u0}을 정점 u까지의 짝수 경로의 최단 거리, d_{u1}을 정점 u까지의 홀수 경로의 최단 거리라고 정의합니다. 인접 정점을 살펴볼 때, 최단 경로의 홀짝성에 따라서 잘 갱신해주면 됩니다.

소스 코드

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

const long long INF = 1e18;

int main() {
    FAST();

    int n, m;
    cin >> n >> m;

    vector<vector<pair<int, int>>> adj(n + 1);
    for (int i = 0; i < m; i++) {
        int u, v, w;
        cin >> u >> v >> w;

        adj[u].emplace_back(v, w);
        adj[v].emplace_back(u, w);
    }

    priority_queue<tuple<long long, int, int>, vector<tuple<long long, int, int>>, greater<tuple<long long,
int, int>>> pq;
    vector<vector<long long>> dist(2, vector<long long>(n + 1, INF));

    pq.emplace(0, 0, 1);
    dist[0][1] = 0;
```

```
    while (!pq.empty()) {
        auto [cost, parity, here] = pq.top();
        pq.pop();

        if (dist[parity][here] < cost) {
            continue;
        }

        for (auto [there, weight] : adj[here]) {
            long long there_cost = cost + weight;

            if (dist[there_cost & 1][there] > there_cost) {
                dist[there_cost & 1][there] = there_cost;
                pq.emplace(there_cost, there_cost & 1, there);
            }
        }
    }

    for (int i = 1; i <= n; i++) {
        cout << (dist[1][i] == INF ? -1 : dist[1][i]);
        cout << ' ';
        cout << (dist[0][i] == INF ? -1 : dist[0][i]);
        cout << '\n';
    }
```

```
}
```