SUAPC 2023 Winter J번 'BOJ 27531 치즈 G1' 해설

jemin0619

2024.11.24

1 문제 분석

각 줄마다 들어오는 입력을 다른 방식으로 해석할 수 있다.
a_i b_i p_i가 주어질 때, 이를 그래프로 모델링해서 간선 p_i를 선택하면 a_i와 b_i를 살 수 있는 것으로 바꿀 수 있다. $(a_1, a_2, ..., a_N)$과 $(b_1, b_2, ..., b_N)$는 모두 1부터 N까지의 수를 한 번씩만 포함하는 순열이므로 이를 통해 만들어질 그래프에서는 모든 노드가 크고 작은 $cycle$에 속한다는 것을 알 수 있다.

그러면 각 $cycle$을 분리한 뒤, $cycle$마다 $cycle$에 속한 정점들을 모두 고르기 위해서 필요한 최소 비용을 구해주면 풀 수 있다. $cycle$을 분리하는 과정은 $Permutation\ Cycle\ Decompositon$을 통해 할 수 있고, 최소 비용은 DP를 통해 $O(N)$에 구할 수 있다.

$cycle$이므로 원형 DP를 적용해야 한다는 것에 주의한다. 시작과 끝을 잇는 간선을 $edge[0]$으로 할 때 점화식은 다음과 같다.

$dp1$: 시작과 끝을 잇고 시작한 경우
$dp2$: 시작과 끝을 잇지 않고 시작한 경우

$$dp1[0] = edge[0]$$
$$dp1[1] = edge[0] + edge[1]$$
$$dp1[i] = min(dp1[i-1],\ dp1[i-2]) + edge[i] \quad (1 < i < N)$$

$$dp2[0] = INF$$
$$dp2[1] = edge[1]$$
$$dp2[i] = min(dp2[i-1],\ dp2[i-2]) + edge[i] \quad (1 < i < N)$$

답이 될 수 있는 후보는 3가지이다.
$dp1[N-1]$: 시작과 끝을 이어놓은 상태에서 $edge[N-1]$을 선택한 경우
$dp1[N-2]$: 시작과 끝을 이어놓은 상태에서 $edge[N-2]$을 선택한 경우
$dp2[N-1]$: 시작과 끝을 잇지 않은 상태에서 $edge[N-1]$을 선택한 경우 (선택해야만 시작 정점을 고를 수 있음)
즉, $min(dp1[N-1],\ dp1[N-2],\ dp2[N-1])$이 답이 된다.

2 관련 글 및 문제

SUAPC 2023 Winter Solution - Official Solutions

BOJ 17404 RGB거리 2 G4

3 코드

```cpp
#include <bits/stdc++.h>
using namespace std;
#define fastio cin.tie(NULL)->sync_with_stdio(false)
#define ll long long
#define pii pair<ll, ll>

vector<pii> adj[200'001];
vector<bool> vis(200'001, false);

int main(){
    fastio;
    ll N; cin>>N;
    for(int i=0; i<N; i++){
        ll a,b,c; cin>>a>>b>>c;
        adj[a].push_back({b, c});
        adj[b].push_back({a, c});
    }

    ll ans = 0;
    for(int i=1; i<=N; i++){
        if(vis[i]) continue;
        vector<ll> group;
        vis[i] = true;
        int cur = adj[i][0].first;
        group.push_back(adj[i][1].second);
        group.push_back(adj[i][0].second);
        while(cur != adj[i][1].first){
            vis[cur] = true;
            for(auto[nxtN, nxtW] : adj[cur]){
                if(vis[nxtN]) continue;
                group.push_back(nxtW);
                cur = nxtN;
            }
        }
        vis[cur] = true;

        vector<ll> dp1(group.size(), 0), dp2(group.size(), 0);
        dp1[0] = group[0];
        dp1[1] = group[0] + group[1];

        dp2[0] = 0x7f7f7f7f;
        dp2[1] = group[1];

        for(int j=2; j<group.size(); j++){
            dp1[j] = min(dp1[j-1], dp1[j-2]) + group[j];
            dp2[j] = min(dp2[j-1], dp2[j-2]) + group[j];
        }

        ans += min({dp1[group.size()-1], dp1[group.size()-2], dp2[group.size()-1]});
    }
    cout<<ans;
    return 0;
}
```