BOJ: 1368번 물대기

문제 설명

문제

선주는 자신이 운영하는 N개의 논에 물을 대려고 한다. 물을 대는 방법은 두 가지가 있는데 하나는 직접 논에 우물을 파는 것이고 다른 하나는 이미 물을 대고 있는 다른 논으로부터 물을 끌어오는 법이다. 각각의 논에 대해 우물을 파는 비용과 논들 사이에 물을 끌어오는 비용들이 주어졌을 때 최소의 비용으로 모든 논에 물을 대는 것이 문제이다.

입력

첫 줄에는 논의 수 N(1 ≤ N ≤ 300)이 주어진다. 다음 N개의 줄에는 i번째 논에 우물을 팔 때 드는 비용 Wi(1 ≤ Wi ≤ 100,000)가 순서대로 들어온다. 다음 N개의 줄에 대해서는 각 줄에 N개의 수가 들어오는데 이는 i번째 논과 j번째 논을 연결하는데 드는 비용 Pi,j(1 ≤ Pi,j ≤ 100,000, Pi,j = Pj,i, Pi,i = 0)를 의미한다.

출력

첫 줄에 모든 논에 물을 대는데 필요한 최소비용을 출력한다.

접근 방법

논에 대한 연결이 Pi,j = Pj,i를 만족하는 무방향 그래프로 주어진다. 연결 비용을 최소화 하려면 그래프에서 사이클이 생기지 않는 가장 비용이 낮은 N - 1개의 간선을 선택하면 된다. 따라서 최소 신장 트리를 이용하여 문제 해결을 시도하였다.

처음 생각할 부분은 시작 점의 설정이다. 논을 연결하는 것도 중요하지만 물을 댈 수 있는 우물도 반드시 있어야 한다. 이를 위해서 우물 파는 비용이 가장 낮은 곳을 시작 점으로 설정했다. “우물 파는 비용이 가장 낮은 논의 우물을 반드시 파는게 옳은가?”에 대해서 생각하게 되었다.

1. 우물 파는 비용이 가장 낮은 논이 다른 논과 연결된 경우
   우물 파는 비용이 가장 낮은 논의 우물을 파는 것이 가장 비용이 적게 든다.
2. 우물 파는 비용이 가장 낮은 논이 다른 논과 연결되지 않은 경우
   우물 파는 비용이 가장 낮은 논의 우물은 반드시 판다.

위와 같은 결론을 스스로 내렸고, 시작 점은 우물 파는 비용이 가장 낮은 곳이다. 최소 신장 트리를 만드는 프림 알고리즘을 사용하였다. 시작 논과 연결된 논을 대상으로 하여 우물을

파는 것과 기존의 논을 연결하는 것을 비교하여 둘 중 비용이 더 낮은 것을 선택하는 방식을 택하였다.

하지만 방식에 한 가지 문제가 있었다. 바로 우물을 파는 것을 우선순위 큐에 넣지 않은 것이다. 이로 인해 우물을 정확히 파지 못하는 문제가 생겨서 문제 해결에 어려움을 겪었다. 이 문제는 우물 파는 비용을 우선순위 큐에 넣으며 해결하였다.

주요 코드

전역 변수 선언

```
int n;
vector<pair<int, int>> dig;
int adj[300][300];
priority_queue<tuple<int, int, int>, vector<tuple<int, int, int>>, greater<>> pq;
vector<int> visited(300, 0);
```

입력

```
cin >> n;
    for (int i = 0; i < n; i++) {
        int cost;
        cin >> cost;
        dig.emplace_back(cost, i);
    }
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> adj[i][j];
```

최소 우물 파기 비용 찾기

```
int start = 0;
    int ans = 1e9;

    for (int i = 0; i < n; i++) {
        if (dig[i].first < ans) {
            ans = dig[i].first;
            start = i;
```

```
        }
    }
```

우선순위 큐 초기화 및 프림 알고리즘 적용

```
int cnt = 0;
    visited[start] = 1;

    for (int i = 0; i < n; i++) {
        if (start == i) continue;
        pq.emplace(adj[start][i], start, i);
        pq.emplace(dig[i].first, -1, i);
    }

    while (cnt < n - 1) {
        int a, b, cost;
        tie(cost, a, b) = pq.top(); pq.pop();
        if (visited[b]) continue;

        visited[b] = 1;
        ans += cost;
        cnt++;

        for (int i = 0; i < n; i++) {
            if (b == i or visited[i]) continue;
            pq.emplace(adj[b][i], b, i);
        }
    }
```

느낀점

문제에서 발생할 수 있는 예외에 대한 고려와 실수를 줄이기 위한 노력을 해야겠다고 느꼈다. 처음에는 dig[] 배열을 정렬을 해서 논 - 우물 파기 비용이 잘못 연결되어 원하는 답을 얻지 못했다. 이런 기본적인 실수를 줄이고 문제에서 발생할 수 있는 예외를 잘 생각할 수 있으면 더 빠르고 정확하게 풀 수 있을 것이다.