#12-2. 플로이드 와샬 알고리즘

나정휘

https://justiceHui.github.io/

플로이드 와샬 알고리즘

Floyd-Warshall Algorithm

- APSP를 푸는 알고리즘
- 시간 복잡도 : $O(V^3)$
- 공간 복잡도 : $O(V^3) \rightarrow O(V^2)$

- DP 기반 알고리즘
  - D[k][u][v] : u에서 출발해서 1..k번 정점을 경유한 뒤 v번 정점으로 가는 최단 거리
  - D[0][u][u] : 0으로 초기화
  - u ≠ v일 때 D[0][u][v]는 간선이 있으면 간선의 가중치, 없으면 INF로 초기화
  - D[k][u][v] ← D[k-1][u][k] + D[k-1][k][v]
  - 배열 D의 크기는 V^3

플로이드 와샬 알고리즘

Floyd-Warshall Algorithm

- 배열 D의 크기를 $2V^2$로 줄일 수 있음
  - D[k][u][v]를 계산할 때 D[k-1][*][*]만 사용하므로
  - D[2][V][V] 크기로 만들고 토글링하면 됨
- 배열 D의 크기를 V^2로 줄일 수 있음
  - 최솟값을 구하는 문제인데 값이 매번 감소하기 때문에 그냥 덮어써도 됨
  - 시간 복잡도는 여전히 $O(V^3)$

플로이드 와샬 알고리즘

```cpp
int N, M, D[111][111];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> M;
    for(int i=1; i<=N; i++) for(int j=1; j<=N; j++) D[i][j] = 1e9; // inf로 초기화
    for(int i=1; i<=N; i++) D[i][i] = 0;                           // i -> i 비용은 0
    for(int i=1; i<=M; i++){
        int s, e, w; cin >> s >> e >> w;
        D[s][e] = min(D[s][e], w);
    }
    // 여기까지 오면, D[i][j]는 다른 정점을 경유하지 않고 i에서 j로 직접 가는 최단 거리
    for(int k=1; k<=N; k++){
        for(int i=1; i<=N; i++){
            for(int j=1; j<=N; j++){
                D[i][j] = min(D[i][j], D[i][k] + D[k][j]);
            }
        }
        // D[i][j]: 1..k번 정점을 경유해서 i에서 j로 가는 최단 거리
    }
    for(int i=1; i<=N; i++) for(int j=1; j<=N; j++) cout << D[i][j] << " \n"[j==N];
}
```

질문?

플로이드 와샬 알고리즘

BOJ 11780 플로이드 2

- 경로가 존재하는 모든 (i, j)에 대해, i에서 j로 가는 최단 거리와 그 경로를 구하는 문제

- P[i][j] = i에서 j로 가는 최단 경로에서 경유하는 가장 큰 정점 번호
  - 만약 i에서 바로 j로 가는 경로가 최소 비용이라면 P[i][j] = 0

- i → j 최단 경로를 구하는 것은 i → P[i][j]의 최단 경로, P[i][j] → j의 최단 경로를 구하는 것으로 분할 가능
  - Recursive(s, e) = s에서 e로 갈 때 경유하는 모든 정점을 구하는 함수

```
void Recursion(vector<int> &v, int s, int e){
    if(P[s][e] == 0) return;
    Recursion(v, s, P[s][e]);
    v.push_back(P[s][e]);
    Recursion(v, P[s][e], e);
}

vector<int> GetPath(int s, int e){
    vector<int> path;
    path.push_back(s);
    Recursion(path, s, e);
    path.push_back(e);
    return path;
}
```

플로이드 와샬 알고리즘

BOJ 23258 밤편지

- N개의 정점으로 구성된 무향 가중치 그래프가 주어짐
- 출발 정점 S와 도착 정점 T가 주어지면, 반딧불은 S에서 T까지 최단 경로로 이동함
- x번 정점에는 2x 만큼의 이슬이 있고, 반딧불은 S와 T를 제외하고 방문하는 모든 정점에서 이슬을 마심
- 각 반딧불은 상수 C를 갖고 있어서, 이슬을 2C 방울 이상 마시면 더 이상 이동할 수 없음
- S, T, C가 주어지면 S에서 T로 가는 최소 시간을 구하는 쿼리를 Q번 처리하는 문제

- 사실 경유하는 정점 번호의 최댓값만 신경 써도 됨
  - $2^1 + 2^2 + \ldots + 2^{x-1} < 2^x$ 이기 때문
  - D[k][i][j] = k번 이하의 정점만 경유해서 i에서 j로 가는 최단 거리

- 플로이드 와샬 알고리즘을 이용해서 해결할 수 있음
  - 시간 복잡도, 공간 복잡도 모두 $O(N^3)$

플로이드 와샬 알고리즘

BOJ 23258 밤편지

```cpp
#include <bits/stdc++.h>
using namespace std;
constexpr int INF = 0x3f3f3f3f;

int N, Q, D[333][333][333];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    cin >> N >> Q;
    for(int i=1; i<=N; i++){
        for(int j=1; j<=N; j++){
            cin >> D[0][i][j];
            if(D[0][i][j] == 0 && i != j) D[0][i][j] = INF;
        }
    }
    for(int k=1; k<=N; k++){
        for(int i=1; i<=N; i++){
            for(int j=1; j<=N; j++){
                D[k][i][j] = min(D[k-1][i][j], D[k-1][i][k] + D[k-1][k][j]);
            }
        }
    }
    while(Q--){
        int c, s, e; cin >> c >> s >> e;
        if(D[c-1][s][e] == INF) cout << -1 << "\n";
        else cout << D[c-1][s][e] << "\n";
    }
}
```

질문?