플로이드-와샬 알고리즘

모든 지점에서 모든 지점까지의 길을 찾아주는 알고리즘

이전 다익스트라 알고리즘의 경우 **시작 지점이 정해져 있는 알고리즘**입니다. 다라서 모든 경로에서 최소 거리를 찾기 위해서는 다익스트라를 N번만큼 돌려야 하는 번거로움이 있습니다.

반면 이번에 보는 플로이드-와샬 알고리즘의 경우 시간 복잡도 $O(N^3)$의 속도로 **모든 경로에서의 최솟값**을 찾아주며 **코드가 상당히 짧아** 느린 시간복잡도에도 많이 사용하게 됩니다.

그 외에도 위와 같이 모든 '모든 경로를 탐색해 준다'는 특징을 이용해 '모든 노드간 관계'를 구하는 변형적인 문제들도 많이 보이게 됩니다.

핵심 개념

이 알고리즘의 핵심 경로는 **경유지** 입니다.

만일 그냥 A→B 로 가는 것보다 C 를 거쳐서 가는 길이 더 빠르다고 하면 A→C→B 의 비용을 A→B 로 저장하는 것입니다.

플로이드-와샬 C++ 코드

```
#include <iostream>
using namespace std;

int n; // 노드 수
int road[100][100];

void FloydWarshall(){
    for(int via = 1; via <= n; via++){
        for(int from = 1; from <= n; from++){
            for(int to = 1; to <= n; to++){
                if(road[from][to] > road[from][via] + road[via][to])
                    road[from][to] = road[from][via] + road[via][to];
            }
        }
```

```
    }
}
```

일단 플로이드 와샬은 **'모든 지점→모든 지점'**의 경로를 뽑는게 목표입니다. 따라서 2차원 배열을 통해서 `array[시작점][도착점]` 으로 경로의 비용을 저장해줍니다.

그리고 3중 포문을 통해서 `시작점→도착점` 과 `시작점->경유지->도착점` 값을 비교하고 업데이트 해 주는 것입니다. 여기서 순서가 중요한데 **경유지, 출발지, 도착지** 순으로 보면 됩니다.

- 이 알고리즘은 '경유지'를 탐색하는게 메인 목적으로 순서가 다르면 최적화된 값이 나오지 않습니다.
    - 더 자세히 이야기 하자면 만일 경유지 탐색을 하는 포문이 안에 있으면 탐색에 있어서 **순서**가 생기게 됩니다.
    - 1 → 2 → 3 → 4 로 가야하는 경로에서 1 → 4의 경로가 없을 때 `시작지점`을 먼저 탐색해 보겠습니다.
        - 일단 첫 포문에서 from = 1 로 되어 1→2만 연결하게 되고 나머지는 연결점이 없게 됩니다
        - 이후 순차적으로 2, 3, 4를 보는데 각각 2→3, 3→4만을 연결하게 됩니다.
        - 마지막으로 4를 탐색한 뒤 1→4를 탐색하지 않고 종료를 해 1→4 경로가 없는 것으로 처리됩니다.
    - 그러면 플로이드-와샬의 순서대로 `경유지` 먼저 보도록 하겠습니다.
        - 경유지가 1을 봅니다, 없으므로 삭제하게 됩니다.
        - 경유지 2를 보면 1→2, 2→3 이 있어 이를 연결해줄 수있으니 1→3 값을 업데이트 해 줍니다.
        - 경유지 3을 본 뒤 2→3 3→4 가 있어 2→4가 업데이트, 위 과정에서 1→3 이 생겨 1→4도 생기게 됩니다.
        - 이후 탐색을 종료하고 보면 1→4의 값을 얻을 수 있게 됩니다.
- 경유지가 우선이 되면 **이후 연결되는 경우가 있어도 그 부분을 다시 탐색하지 않고도 값을 얻을 수 있습니다.** 따라서 경유지를 우선으로 탐색하게 되는 것입니다.

```
#include <iostream>
using namespace std;

int n; // 노드 수
int road[100][100];

void FloydWarshall(){
    for(int via = 1; via <= n; via++){
```

```
        for(int from = 1; from <= n; from++){
            if(road[from][via] == MAX)   continue;
            for(int to = 1; to <= n; to++){
                if(road[from][to] > road[from][via] + road[via][to])
                    road[from][to] = road[from][via] + road[via][to];
            }
        }
    }
}
```

추가로 위 코드처럼 from→via 의 경유지가 있는지를 체크해 속도 향상을 엿볼 수도 있습니다. (단 큰 차이는 없습니다)

추천 문제

11404번: 플로이드

첫째 줄에 도시의 개수 n이 주어지고 둘째 줄에는 버스의 개수 m이 주어진다. 그리고 셋째 줄부터 m+2줄까지 다음과 같은 버스의 정보가 주어진다. 먼저 처음에는 그 버스의 출발 도시의 번호가 주어진다. 버

https://www.acmicpc.net/problem/11404

15723번: n단 논법

모든 중앙대 컴퓨터공학부(소프트웨어학부) 학생들은 미인이다. 지무근은 중앙대 컴퓨터공학부 학생이다. 그러므로 지무근은 미인이다. 위 연역 논증은 대표적인 삼단논법의 예시이다. 삼단논법이란 전제 두 개

https://www.acmicpc.net/problem/15723

11562번: 백양로 브레이크

첫 줄에 Y대학교 건물의 수 n과 길의 수 m이 주어진다. (n ≤ 250, m ≤ n * (n - 1) / 2) 다음 m줄에 걸쳐, u v b (1 ≤ u ≤ n, 1 ≤ v ≤ n, u != v, b = 0 또는 1) 의 형태로 길에 대한 정보가 주어진다.

https://www.acmicpc.net/problem/11562