# Shortest Path

## 플로이드 와샬, 다익스트라, 벨만포드

Dosawas 2024-05-26

# 목차

## 차례

- 7회차 리뷰
- Weighted Graph
- 다익스트라 알고리즘
- 플로이드 워셜 알고리즘
- 벨만 포드 알고리즘

# 7회차 리뷰

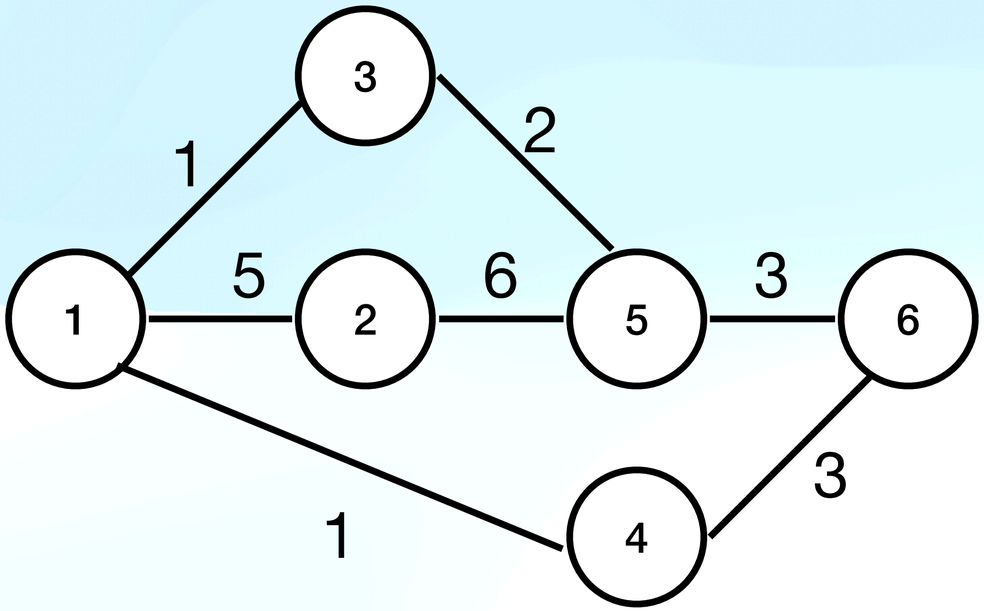
- 트리의 순회
  - Preorder
  - Inorder
  - Postorder
- 그래프
  - Dfs
  - Bfs

# 가중치 그래프의 표현

- Static 으로 선언하면 다른 클래스에서도 쓸 수 있다.
- template에 node size를 넣을 수도 있다.
  - 자기 나름대로 만들어보자
- 사용법 : myGraph<data_type> 구조체명;
- addEdge는
  - 양방향이냐 아니냐
  - 가중치가 있느냐 없느냐
  - 인접 행렬이냐 리스트냐

```
12  template <class T>
13  struct myGraph{
14      static const int V_MAX = 5e4+1;
15      int V, E;
16      vector<pair<T, T> > adj[V_MAX];
17
18      void addEdge(T u, T v, T w){
19          adj[u].push_back({v, w});
20          adj[v].push_back({u, w});
21      }
22
23      // 필요한 최단거리 알고리즘 작성
24      // 필요한 알고리즘 함수 작성(예를 들면 MST, LCA 등등)
25  };
26
27  myGraph<int> G;
28
29  int main(){
30      fast_io
31      cin >> G.V >> G.E;
32      for(int i=0;i<G.E;i++){
33          int u, v, w; cin >> u >> v >> w;
34          G.addEdge(u, v, w);
35      }
36  }
```

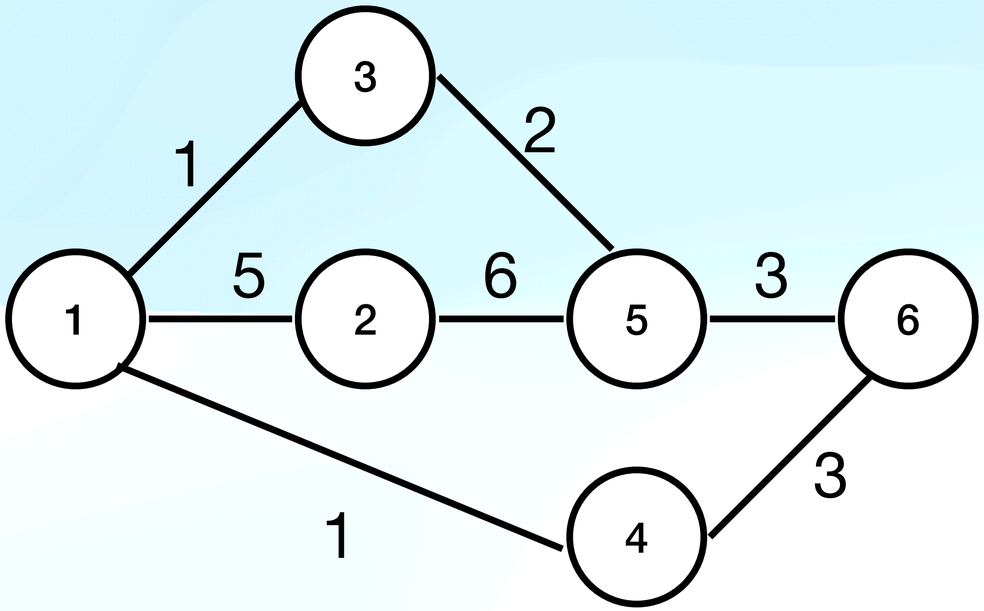
# 최단 거리 문제




엽의 그래프에 대한 정보

1. 양방향 그래프다.
2. 가중치가 있는데 일단 양수다.
3. 사이클이 있다.

# 최단 거리 문제

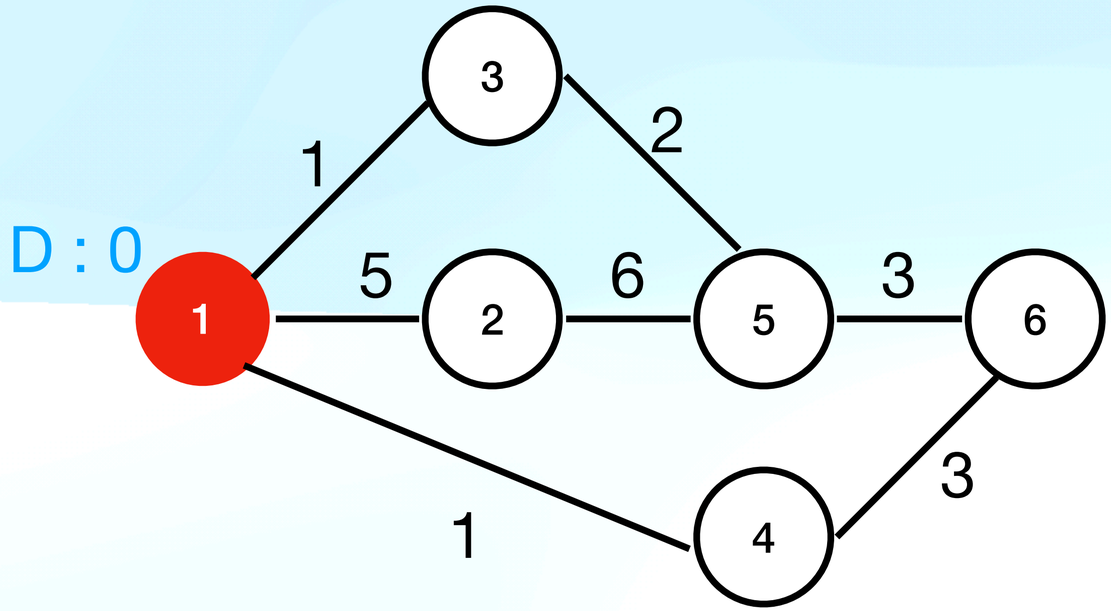



이런 문제는 어떻게 풀까?

- 1번과 5번 사이 최단 거리를 구하시오

일단 1번에서 출발 해볼까요?

# 최단 거리 문제

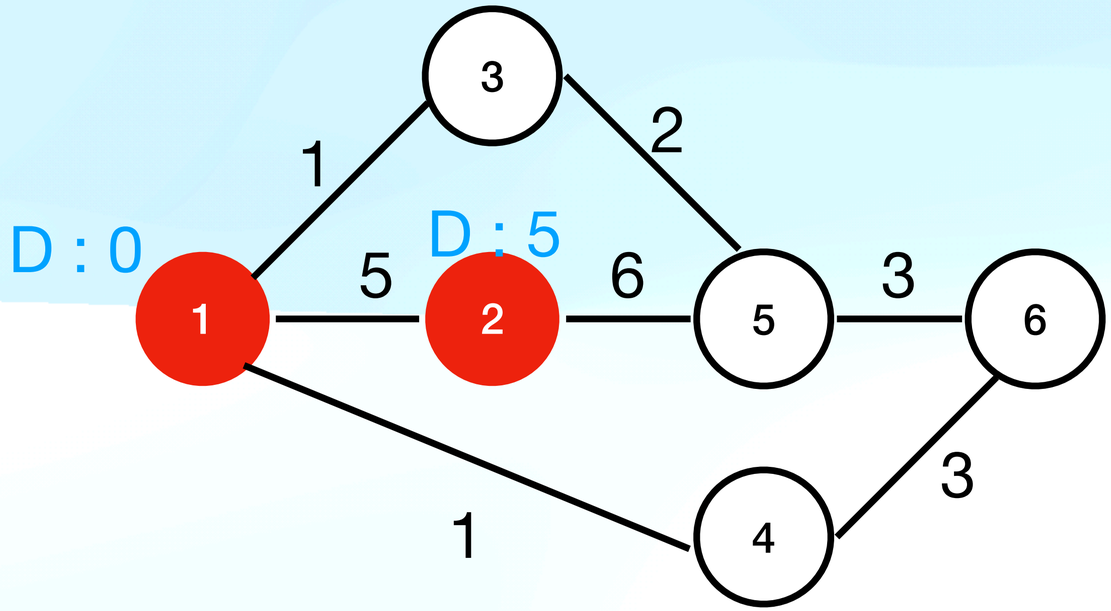



이런 문제는 어떻게 풀까?

- 1번과 5번 사이 최단 거리를 구하시오

일단 1번에서 출발 해볼까요?

# 최단 거리 문제

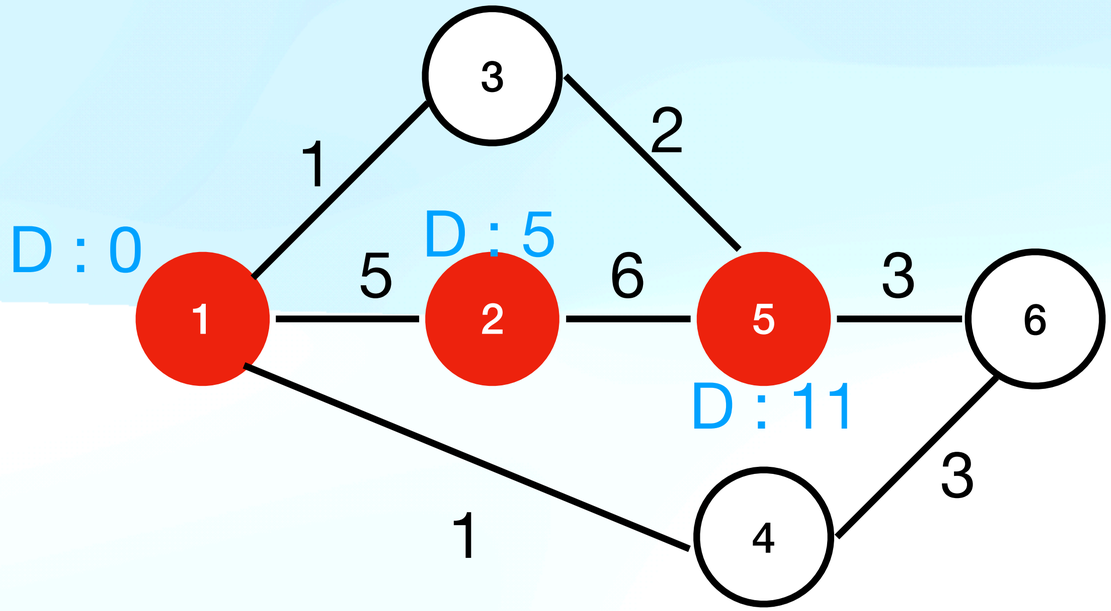



이런 문제는 어떻게 풀까?

- 1번과 5번 사이 최단 거리를 구하시오

일단 1번에서 출발 해볼까요?

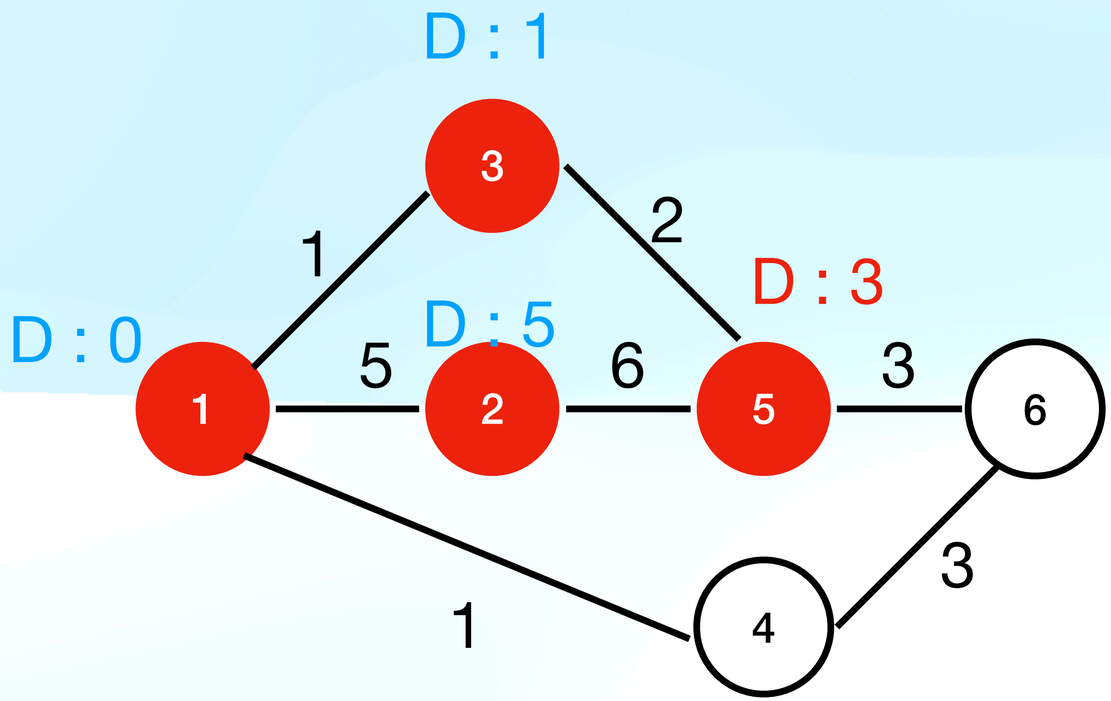
# 최단 거리 문제




- 1번과 5번 사이 최단 거리를 구하시오

1 - 2 - 5 경로로 왔을 때 최단거리는 11이네요.

# 최단 거리 문제




- 1번과 5번 사이 최단 거리를 구하시오

1 - 3 - 5 경로로 왔을 때 최단거리는

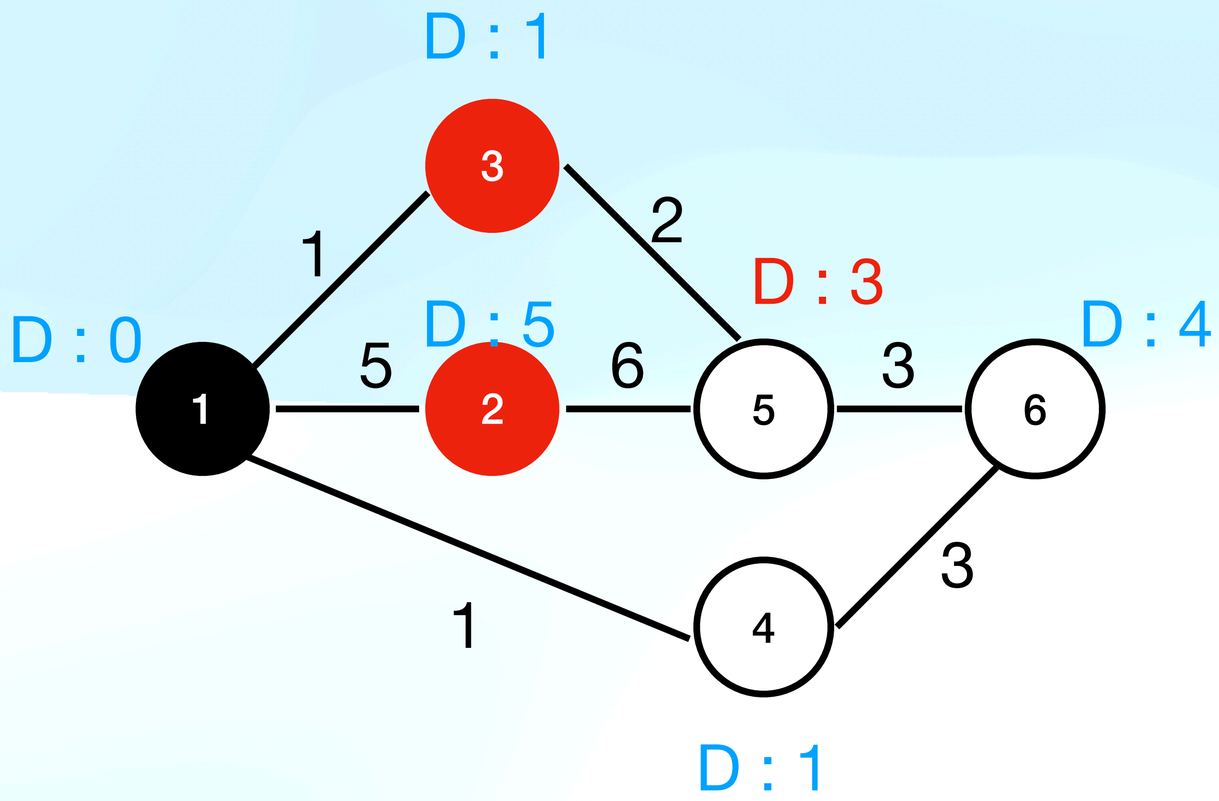
3으로 갱신되었습니다.

이런 과정을 relax라고 생각하면 됩니다.

If $D_5 > D_3 + W_{3,5}$ :

$$D_5 = D_3 + W_{3,5}$$

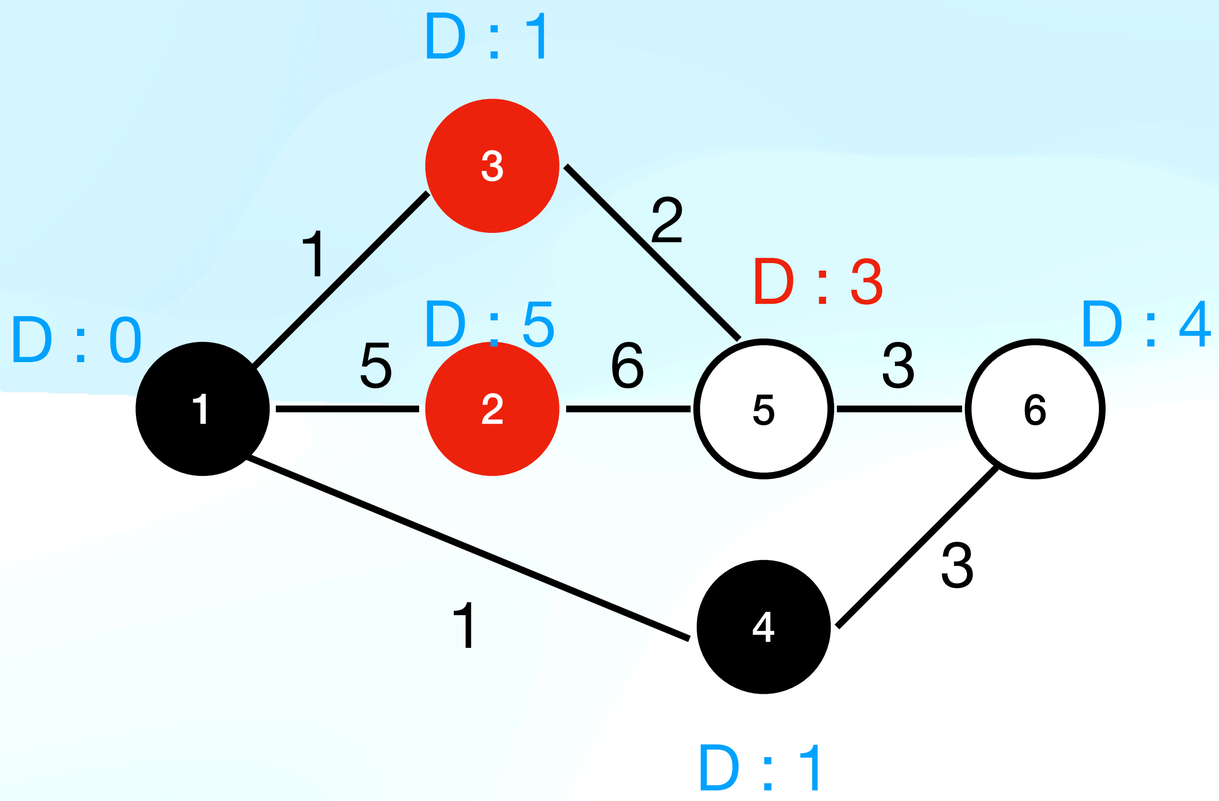
# 최단 거리 문제




- 1번과 5번 사이 최단 거리를 구하시오

다른 길로 가봅시다.

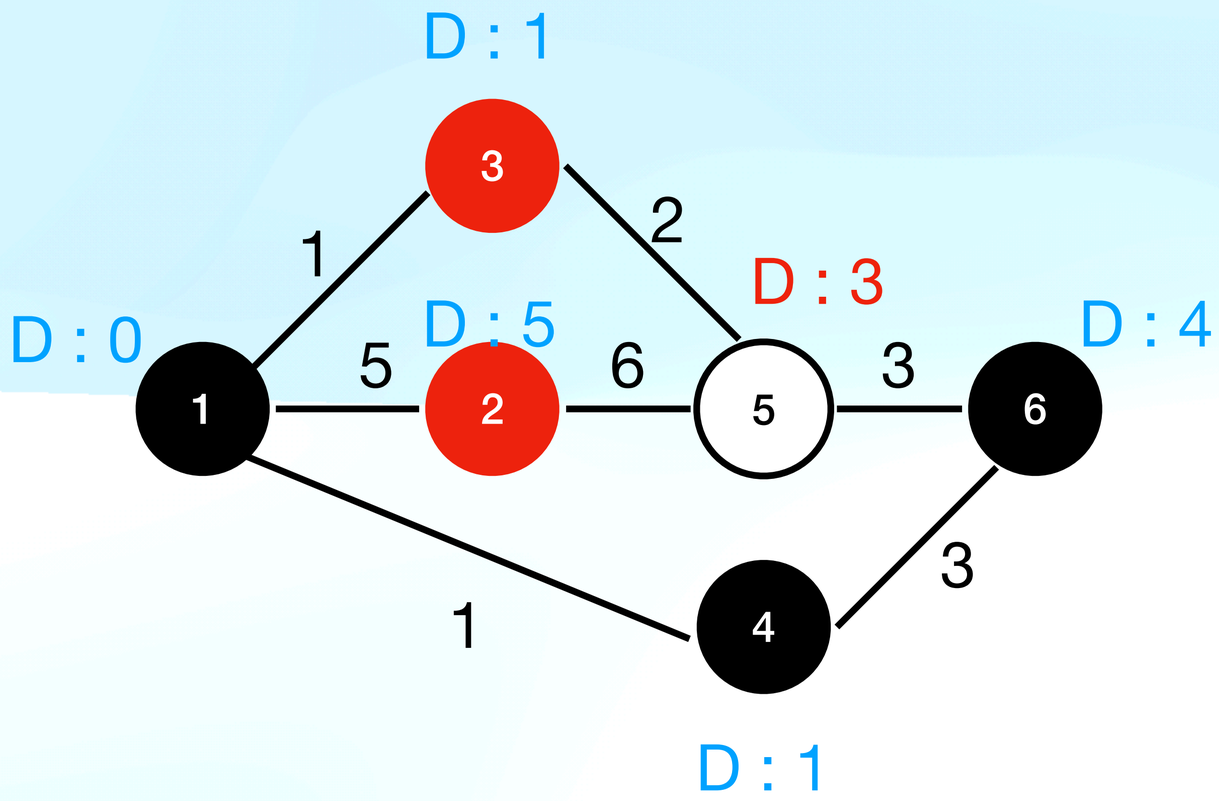
# 최단 거리 문제




- 1번과 5번 사이 최단 거리를 구하시오

다른 길로 가봅시다.

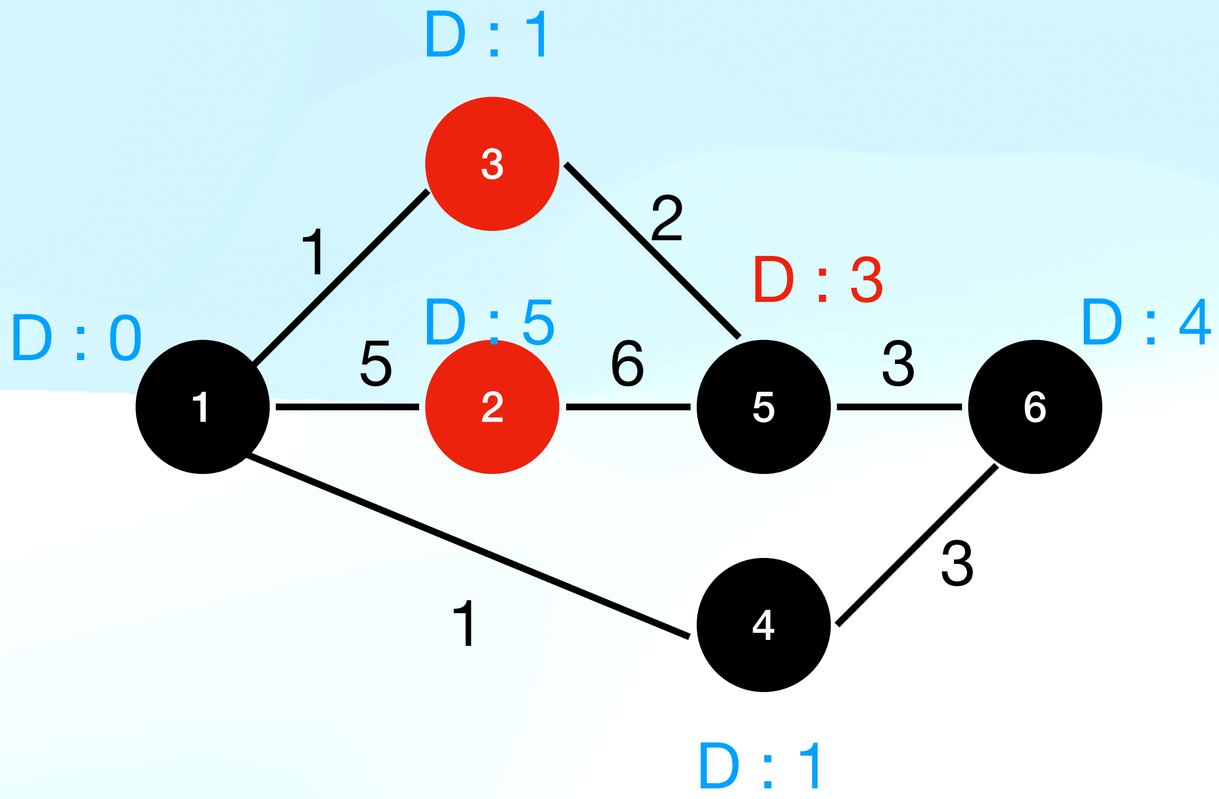
# 최단 거리 문제




- 1번과 5번 사이 최단 거리를 구하시오

다른 길로 가봅시다.

# 최단 거리 문제



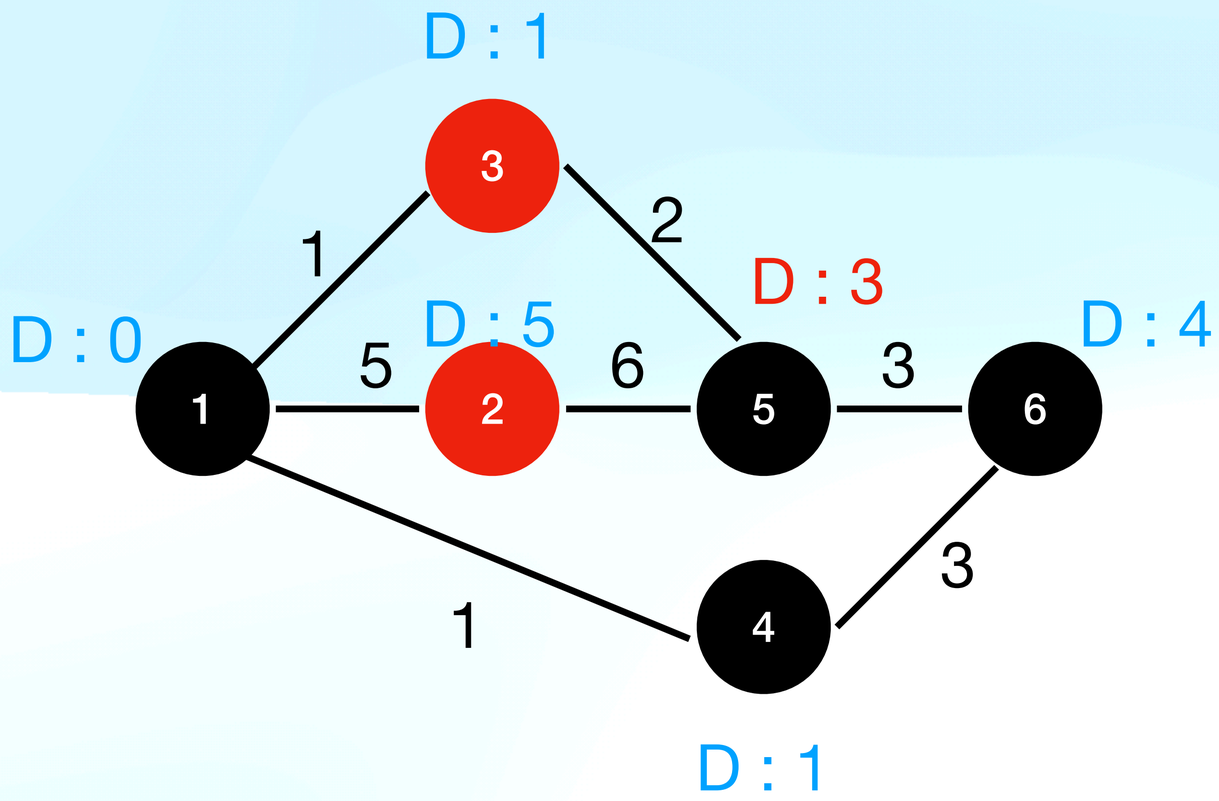

- 1번과 5번 사이 최단 거리를 구하시오

1 - 4 - 6 - 5 의 경로로 왔을 때는

1 + 3 + 3 = 7이지만

D는 이미 3이므로 relax하지않습니다.

# 최단 거리 문제



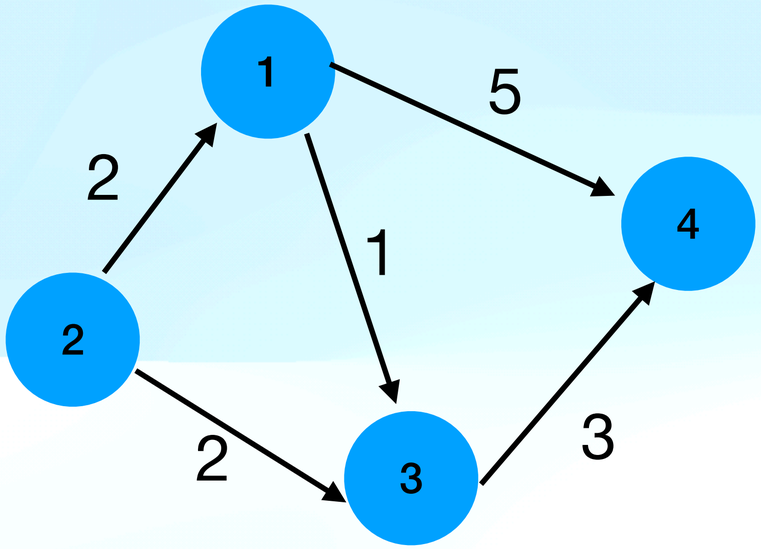

- 그럼 이런 질문들은 어떨까요?
- 1 ~ 4 최단거리
- 1 ~ 6 최단거리
- 1 ~ 2 최단거리
- 등등

모든 쌍의 최단거리를 구해주세요!!

# 플로이드 와샬 - 모든 쌍 최단거리

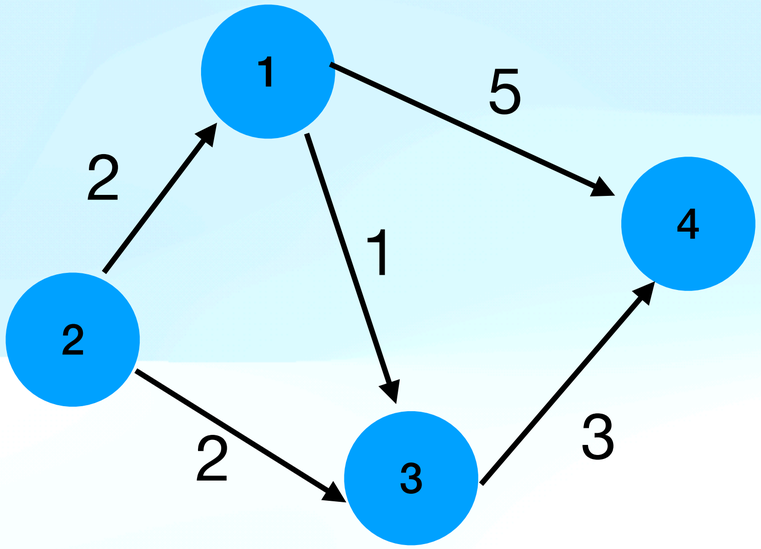
```cpp
template <class T>
struct myGraph{
    static const int V_MAX = 1e2+1;
    static const T INF = 1e9+1;
    int V, E;
    // 인접리스트
    // vector<pair<T, T> > adj[V_MAX];
    // 인접행렬
    T adj[V_MAX][V_MAX];

    void init_edge(){
        for(int i=1;i<=V;i++){
            for(int j=1;j<=V;j++){
                if(i == j) adj[i][j] = 0;
                else adj[i][j] = INF;
            }
        }
    }

    void addEdge(T u, T v, T w){
        //인접 행렬이니까
        adj[u][v] = min(adj[u][v], w);
        // adj[v][u] = min(adj[v][u], w);
    }

    // 플로이드 와샬
    void floyd(){
        for(int drop_by = 1;drop_by<=V;drop_by++){
            for(int i=1;i<=V;i++){
                for(int j=1;j<=V;j++){
                    // drop_by를 들러서 가는게 이득이면 relax
                    adj[i][j] = min(adj[i][j], adj[i][drop_by] + adj[drop_by][j]);
                }
            }
        }
    }

    void print_adj(){
        for(int i=1;i<=V;i++){
            for(int j=1;j<=V;j++){
                if(adj[i][j] == INF) cout << "0 ";
                else cout << adj[i][j] << ' ';
            }
            cout << '\n';
        }
    }
};

myGraph<int> G;

int main(){
    fast_io
    cin >> G.V >> G.E;
    G.init_edge();
    for(int i=0;i<G.E;i++){
        int u, v, w; cin >> u >> v >> w;
        G.addEdge(u, v, w);
    }
    G.floyd();
    G.print_adj();
}
```

# 플로이드 와샬 - 모든 쌍 최단거리



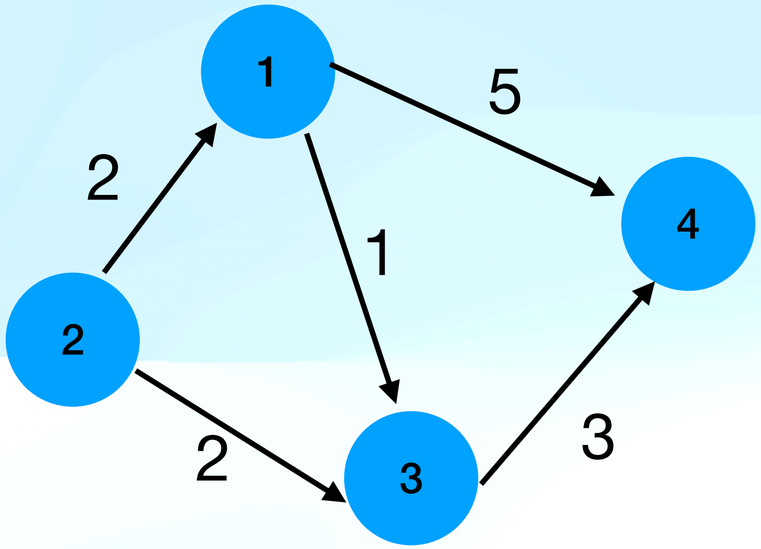

| - | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | INF | INF | INF |
| 2 | INF | 0 | INF | INF |
| 3 | INF | INF | 0 | INF |
| 4 | INF | INF | INF | 0 |

# 플로이드 와샬 - 모든 쌍 최단거리



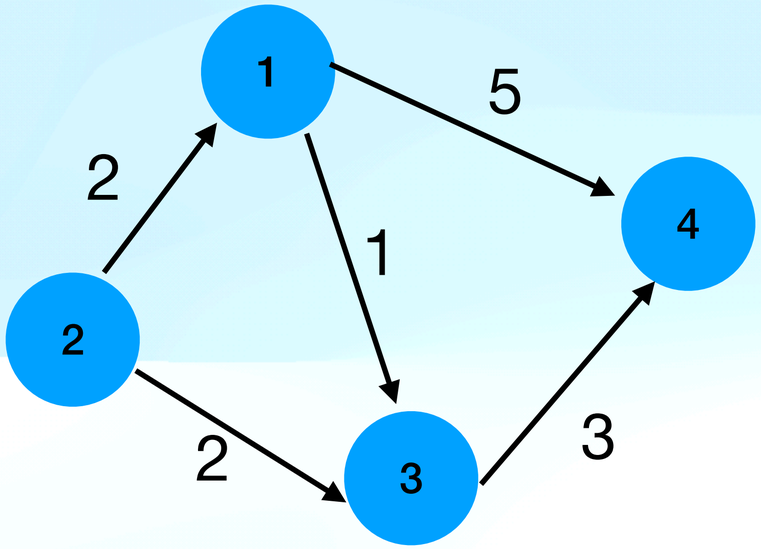

| - | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | INF | 1 | 5 → 4 |
| 2 | 2 | 0 | 2 | 5 |
| 3 | INF | INF | 0 | INF |
| 4 | INF | INF | INF | 0 |

# 플로이드 와샬 - 모든 쌍 최단거리




| - | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | INF | 1 | 5 → 4 |
| 2 | 2 | 0 | 2 | 5 |
| 3 | INF | INF | 0 | 3 |
| 4 | INF | INF | INF | 0 |

# 플로이드 와샬 - 모든 쌍 최단거리




Bottom up 방식

| - | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | INF | 1 | 5 → 4 |
| 2 | 2 | 0 | 2 | 5 |
| 3 | INF | INF | 0 | 3 |
| 4 | INF | INF | INF | 0 |

# 다익스트라 - 한 정점에서 다른 모든 정점까지 최단거리

- 플로이드 워셜은 모든 쌍의 최단거리를 dp를 통해 구해두고 이를 $O(V^3)$에 풀었습니다.
- 근데 굳이 모든 정점 쌍을 구할 필요가 없다면??
  - 예를 들면, 친구들이 홍대에서 출발해서 각자 자기 집까지의 최단거리가 궁금하다.
  - == 각자 집에서 홍대까지의 최단거리가 궁금하다.

- 다익스트라 알고리즘을 통해, 하나의 기준점으로부터 다른 정점까지 최단거리를 구해보자.

# 다익스트라 - 한 정점에서 다른 모든 정점까지 최단거리

- 논리
  - 1. **한 정점**에서 시작, 다른 정점까지의 거리는 **무한대**
  - 2. **가중치가 작은 간선**을 이용해서 다른 정점으로 이동한다.
  - 3. 이동했을 때,
    - 이미 그 정점에 더 빨리가는 길이 있으면 굳이 돌아가는 길은 버려도 된다.
    - 만약 이번에 도착한 경우가 더 빠르다면 그 정점까지 거리를 갱신해준다.
  - 4. 모든 정점을 다 갱신했다면 끝낸다. $O(E \log E)$

# 14221 - 편의점

## 문제

영선이는 이사할 일이 생겨 집을 알아보고 있다. 영선이는 혼자 살기 때문에, 편의점에서 대충 때울 때가 많아, 집을 고르는 기준을 편의점과의 거리가 가장 가까운 곳으로 하려한다.

영선이가 이사할 도시는 정점과 간선으로 표현할 수 있는데, 이사가려 하는 집의 후보들과 편의점은 정점들 위에 있다.

영선이는 캠프 강사 준비로 바쁘므로, 대신하여 집을 골라주자. 만약 거리가 같은 지점이 여러 개라면 정점 번호가 가장 낮은 곳으로 골라주자.

## 입력

처음 줄에는 정점의 개수 n, 간선의 개수 m이 주어진다.(2 ≤ n ≤ 5,000, 1 ≤ m ≤ 100,000) 다음 m줄에는 a,b,c가 주어지는데 이는 a, b를 잇는 간선의 거리가 c라는 것이다.(1 ≤ a, b ≤ n, 1 ≤ c ≤ 10,000)

다음 줄에는 집의 후보지의 개수 p와 편의점의 개수 q가 주어진다.(2 ≤ p+q ≤ n, 1 ≤ p, 1 ≤ q) 다음 줄에는 집의 후보지들의 정점번호, 그 다음줄에는 편의점의 정점번호가 주어진다. 집의 후보지와 편의점은 서로 겹치지 않는다.

## 출력

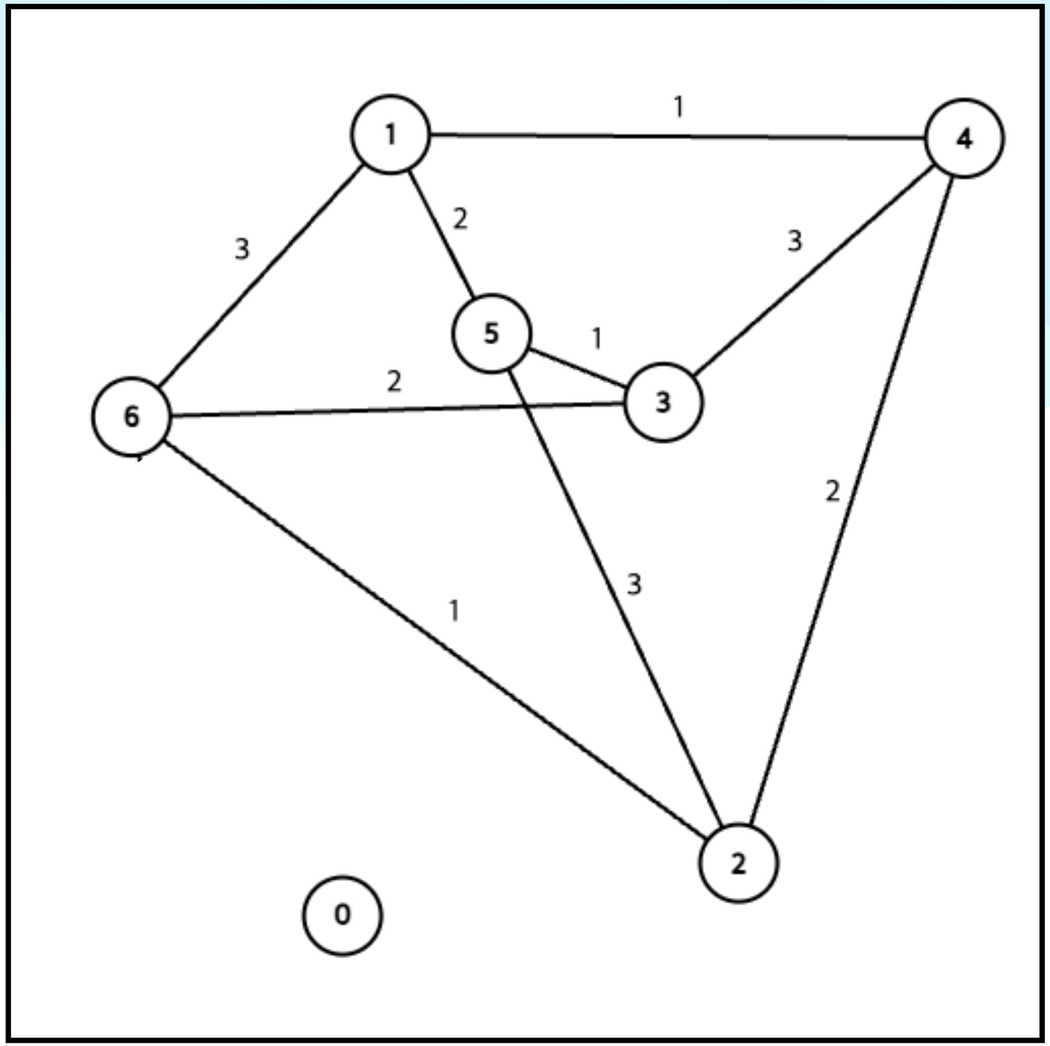
편의점으로부터 가장 가까운 지점에 있는 집 후보의 정점 번호를 출력하시오. 만약 거리가 같은 곳이 여러 군데라면 정점 번호가 낮은 곳을 출력하시오.

# 다익스트라 - 한 정점에서 다른 모든 정점까지 최단거리

- 제일 먼저 생각나는 방법

각 집에서 출발하는 경우를 전처리해두고

편의점에서 각 집까지 거리를
$O(1)$에 구할 수 있도록 한다

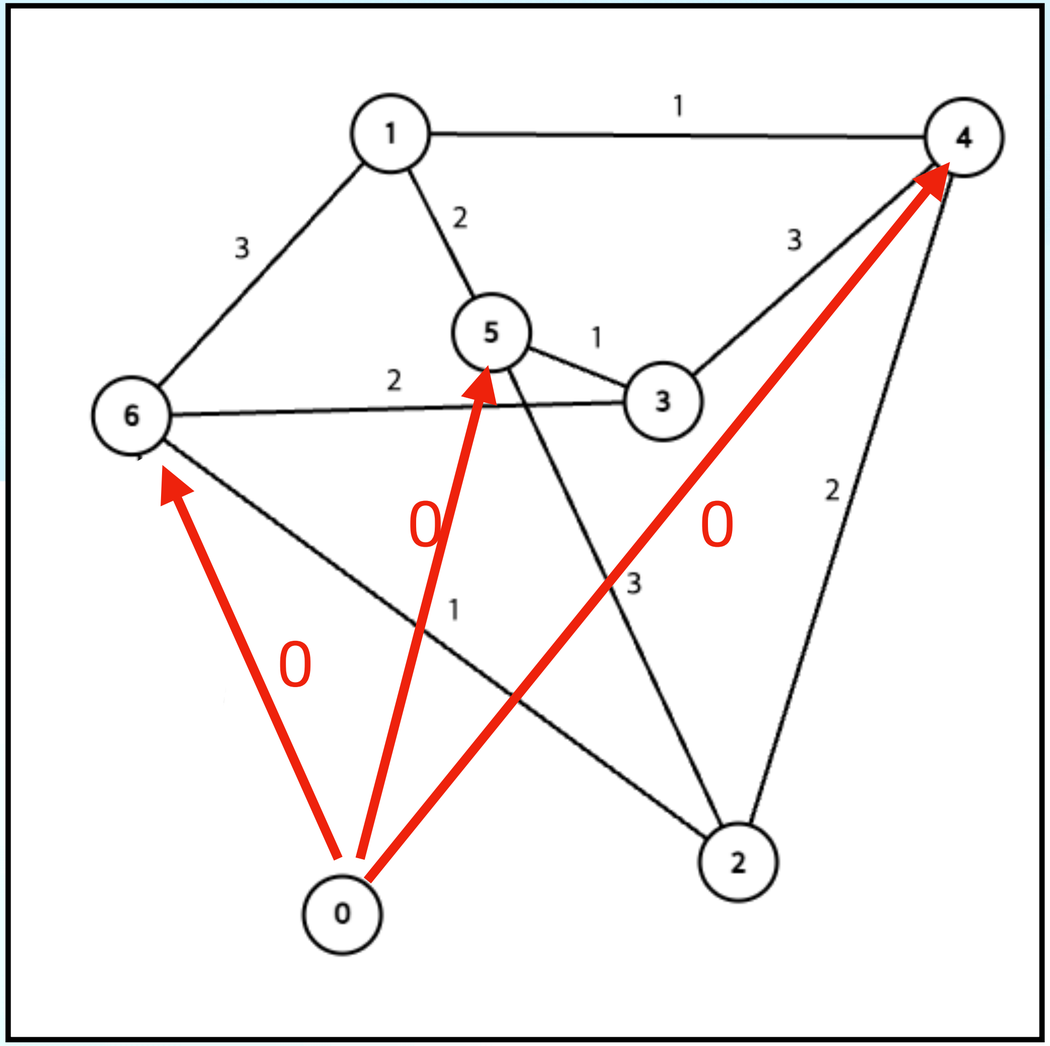



시간 복잡도 : $O(pE \log E + pq)$

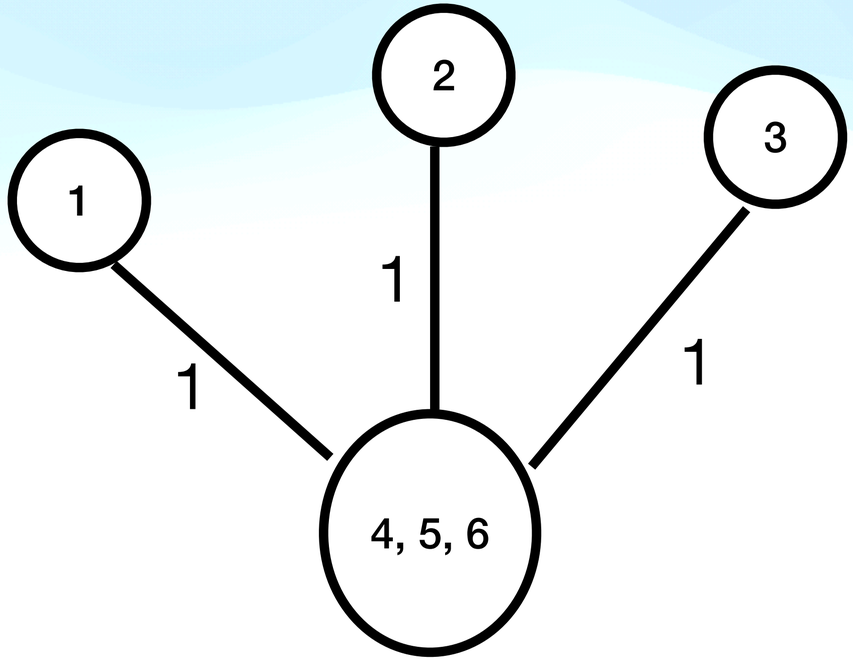
문제는 앞의 항이 시간이 오래 걸린다는 것이다.

E < 1e5

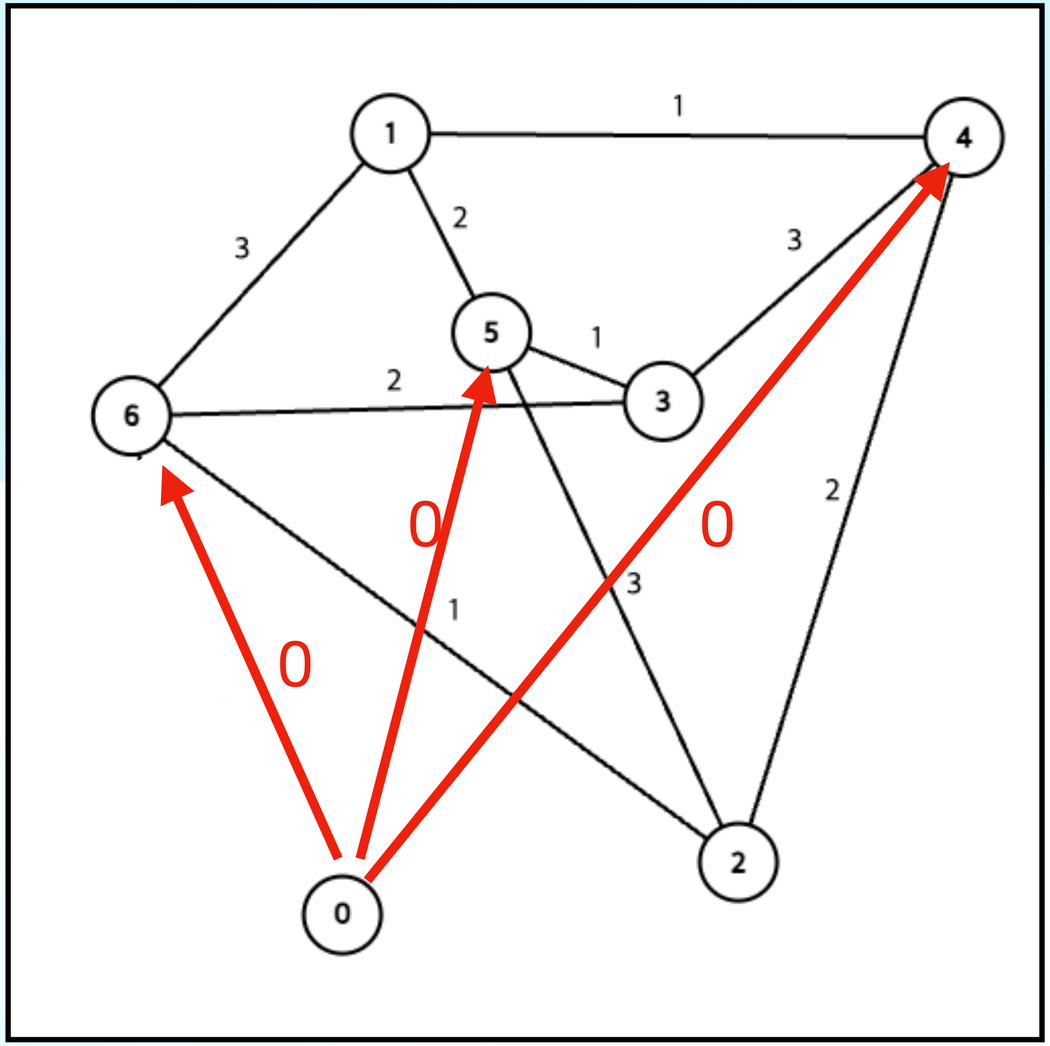
# 다익스트라 - 한 정점에서 다른 모든 정점까지 최단거리




새로운 정점 0을 추가해서
각 편의점으로 길을 만들어 주면



# 다익스트라 - 한 정점에서 다른 모든 정점까지 최단거리




조금 다르게 풀기!

0이라는 정점을 추가하지 않고

애초에 시작점을 pq에 넣을 때

여러개를 넣는 방법도 가능하다!

# 14221-편의점 코드

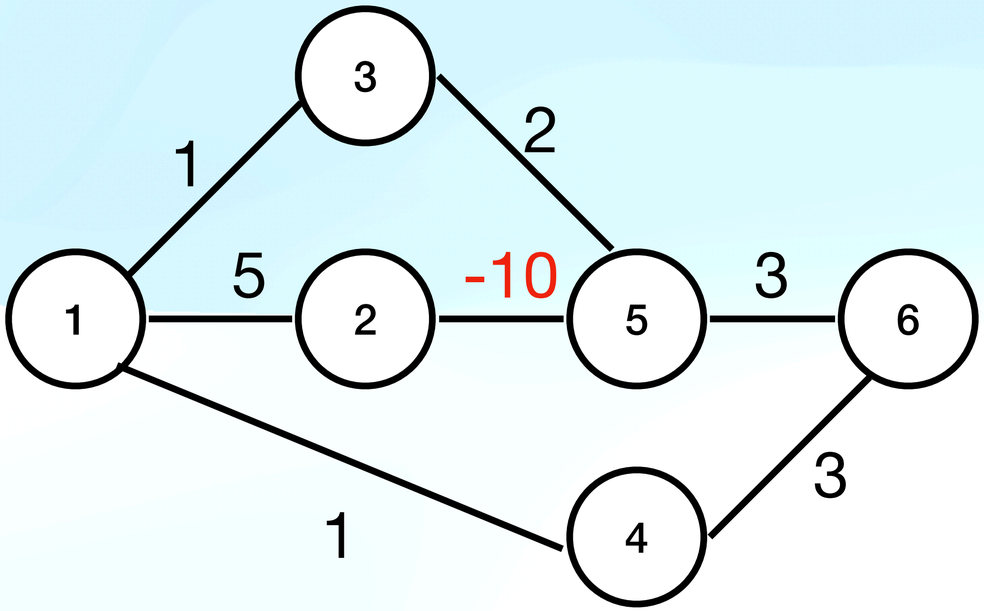
정렬 해두기→

시작점 편의점 집단→

```cpp
myGraph<int> G;

int main(){
    fast_io
    cin >> G.V >> G.E;
    for(int i=0;i<G.E;i++){
        int u, v, w; cin >> u >> v >> w;
        G.addEdge(u, v, w);
    }

    int p, q; cin >> p >> q;
    vector<int> P;
    for(int i=0;i<p;i++){
        int x; cin >> x;
        P.push_back(x);
    }
    sort(P.begin(), P.end());
    for(int i=0;i<q;i++){
        int x; cin >> x;
        // q개의 편의점에서 동시에 시작
        G.adj[0].push_back({x, 0});
    }

    vector<int> D = G.dijkstra(0);
    int mn = 1e9+1;
    int ans = 0;
    for(int pp : P){
        if(mn > D[pp]){
            mn = D[pp];
            ans = pp;
        }
    }
    cout << ans ;
}
```

```cpp
template <class T>
struct myGraph{
    static const int V_MAX = 5e3+1;
    const T INF = numeric_limits<T>().max();

    int V, E;
    // 인접리스트
    vector<pair<T, T> > adj[V_MAX];

    inline void addEdge(T u, T v, T w){
        adj[u].push_back({v, w});
        adj[v].push_back({u, w});
    }

    // 다익스트라
    vector<T> dijkstra(int src){
        vector<T> dist(V+1, INF);
        dist[src] = 0;

        priority_queue<pair<T, T>, vector<pair<T, T>>, greater<pair<T, T>>> pq;
        pq.push({dist[src], src});

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

            if(dist[here] < cost) continue;
            for(auto [nxt, w] : adj[here]){
                if(dist[nxt] > cost + w){
                    dist[nxt] = cost + w;
                    pq.push({dist[nxt], nxt});
                }
            }
        }
        return dist;
    }
};
```

# 다익스트라의 심각한 결함



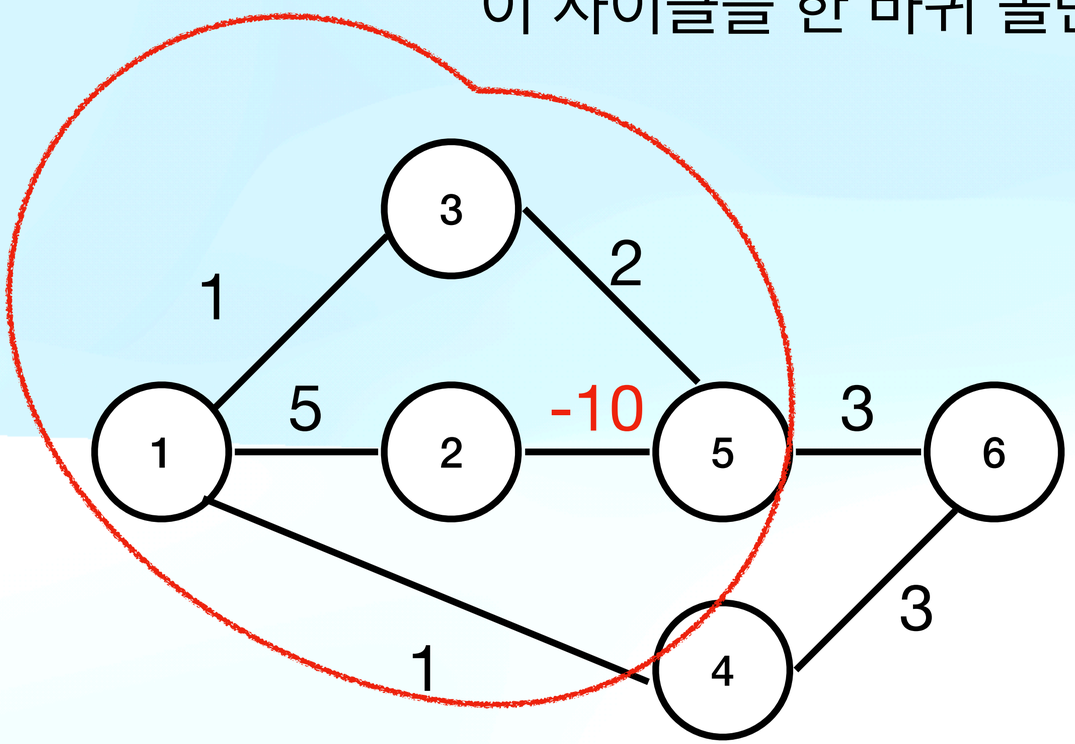

맨 처음 봤던 그래프입니다.

여기서 딱하나의 간선의 가중치만 바뀌었습니다.

최단거리는 어떻게 될까요?

# 다익스트라의 심각한 결함

이 사이클을 한 바퀴 돌면 전체적으로 -2가 줄어든다.



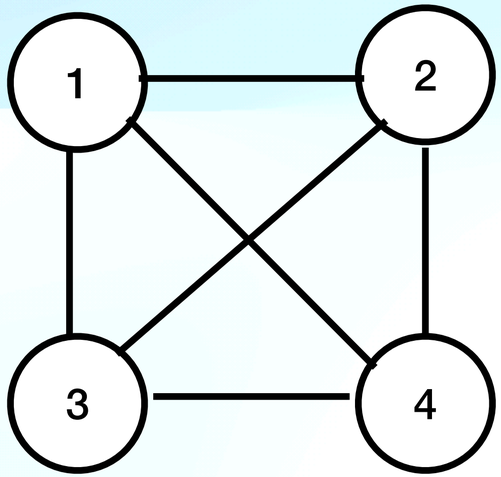

따라서 계속 돌면서 최단거리를 줄일 수 있다.

즉, 다익스트라 알고리즘은 간선이 음수일 때는 사용하기 어렵다.

# 벨만-포드 알고리즘

가장 복잡한 형태의 그래프를 Edge가 가장 많은 그래프라고 생각해보면

정점이 V개인 위와 같은 그래프의 Edge의 개수는 V(V-1)/2개라고 생각할 수 있습니다.

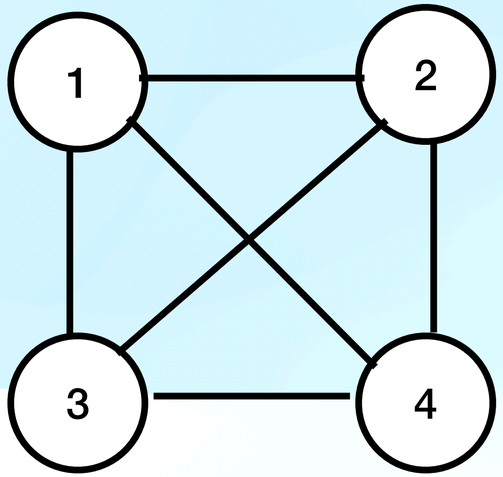



이처럼 6개의 간선이 있는 상태가 가장 복잡할 것입니다.

하지만 6개의 간선을 모두 쓰지 않아도
사실 3개의 간선만 써도
모든 정점에 도착할 수 있습니다.

# 벨만-포드 알고리즘

내가 1번에서 출발한다고 생각하고 4번까지 가는 길을 모두 적어보자.



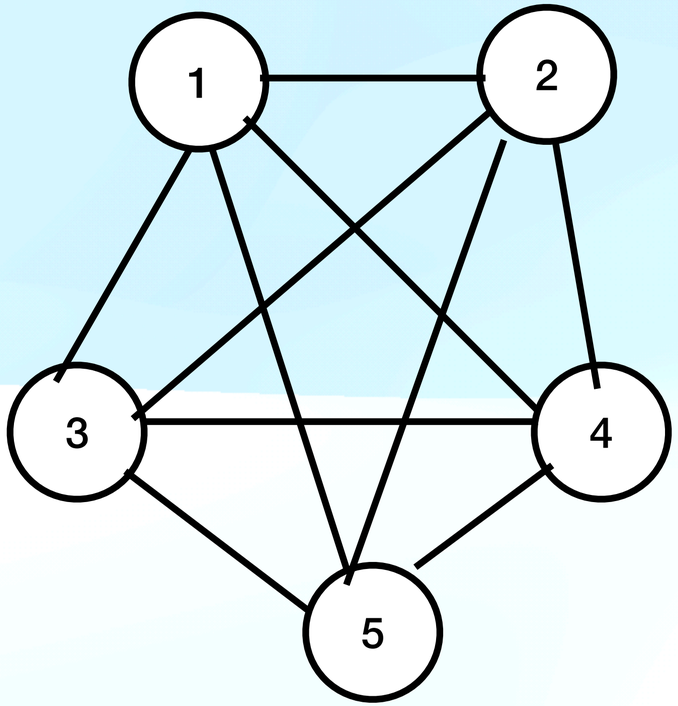

1. 1 → 4
2. 1 → 2 → 4
3. 1 → 3 → 4
4. 1 → 2 → 3 → 4
5. 1 → 3 → 2 → 4

총 5가지의 경우가 있다.

이 중 turn의 수를 세어보자. 그니까 화살표의 개수로 생각하자는 뜻이다.

화살표의 개수는 1, 2, 3 총 3가지가 있다.

# 벨만-포드 알고리즘




이번에도 1번에서 4번으로 가는길을 적어보자

1. 1 → 4
2. 1 → 2 → 4
3. 1 → 3 → 4
4. 1 → 5 → 4
5. 1 → 2 → 3 → 4
6. 1 → 2 → 5 → 4
7. 1 → 3 → 2 → 4

....

10. 1 → 2 → 3 → 5 → 4

..

16. 1 → 5 → 3 → 2 → 4

화살표의 개수는 1, 2, 3, 4개의 경우가 있습니다.

# 벨만-포드 알고리즘

- 예시의 상황을 통해 알 수 있는 사실
  - 음수간선이 없다면 relax되는 일은 V-1(정점의 개수-1)만큼 일어난다.
    - 즉, 그 이후에는 어짜피 반복해도 relax되는 일이 없다.
  - 음수간선이 있다면 , 직접해봅시다. (특히, 음수 사이클이 있는 경우)

- 음수 사이클이 있다면 V번만큼 모든 간선에 의한 relax를 진행해도 더 relax된다.
- 즉, V만큼의 relax과정을 거친 후에도 relax가 되면 그건 음수사이클이 있는 경우다.

# 벨만-포드 알고리즘

1. Upper : 최단거리의 상한

   V만큼의 relax후에 바뀌지 않을 거라면
   이것이 상한이 된다.

2. 음수 사이클로 인한 INF를 V번만큼 계속 더해 나오는 오버플로우를 조심하자!!

3. 마지막에 음수사이클이 있다는 것을 그냥 빈 배열을 리턴해서 알려줬다.

```cpp
// bellmanFord
vector<T> BellmanFord(int src){
    vector<T> upper(V+1, INF);
    upper[src] = 0;
    bool is_relax= false; // V번의 반복 이후 relax 되는가?
    // V번의 반복
    for(int iter = 1; iter <= V ; iter++){
        is_relax = false;
        for(int u = 1; u <= V ; u++){
            for(auto [v , w] : adj[u]){
                if(upper[u] == INF) continue;

                if(upper[v] > upper[u] + w){ // relax됨
                    upper[v] = upper[u] + w;
                    is_relax = true;
                }
            }
        }
        // relax안되었으면 음수 사이클 없는 거임
        if(!is_relax) break;
    }
    // V+1반복 후에 relax되었으면 음수사이클 있는 거임
    if(is_relax) upper.clear();
    return upper;
}
```

# 11657 타임머신 - 벨만포드 기본 문제

## 타임머신 성공

골드 IV

| 시간 제한 | 메모리 제한 | 제출 | 정답 | 맞힌 사람 | 정답 비율 |
|---|---|---|---|---|---|
| 1 초 | 256 MB | 63777 | 15091 | 9637 | 25.143% |

### 문제

N개의 도시가 있다. 그리고 한 도시에서 출발하여 다른 도시에 도착하는 버스가 M개 있다. 각 버스는 A, B, C로 나타낼 수 있는데, A는 시작도시, B는 도착도시, C는 버스를 타고 이동하는데 걸리는 시간이다. 시간 C가 양수가 아닌 경우가 있다. C = 0인 경우는 순간 이동을 하는 경우, C < 0인 경우는 타임머신으로 시간을 되돌아가는 경우이다.

1번 도시에서 출발해서 나머지 도시로 가는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

### 입력

첫째 줄에 도시의 개수 N (1 ≤ N ≤ 500), 버스 노선의 개수 M (1 ≤ M ≤ 6,000)이 주어진다. 둘째 줄부터 M개의 줄에는 버스 노선의 정보 A, B, C (1 ≤ A, B ≤ N, -10,000 ≤ C ≤ 10,000)가 주어진다.

# 마무리

## Check list

- 그래프 구조체 만들어보기
- 플로이드 워셜 알고리즘은 언제 쓰는가? 그리고 구현할 수 있는가?
- 다익스트라 알고리즘은 언제 쓰는가? 그리고 구현할 수 있는가?
- 벨만 포드 알고리즘은 언제 쓰는가? 그리고 구현할 수 있는가?

# 마무리

## 추천문제

- 11404 플로이드
- 11780 플로이드2 - 비슷하게 생긴 배열을 만들어 역추적하기
- 1504 특정한 최단경로
- 14221 편의점 - 좋은 아이디어를 담은 다익스트라
- 11657 타임머신 - 벨만포드 기본
- 1865 웜홀 - 음수사이클이 있다면?

# 마무리

## 추천문제

- 17835 면접보는 승범이네 - 어디부터 최단거리를 구할까
- 16681 등산 - 정답을 어떻게 구해야할지 생각해보자.
  - 힌트 : 1에서 한번 N에서 한번 다익스트라 돌리고 이 두 가지와 높이 배열로 정답을 찾아내보기
- 13141 Ignition - 플로이드 와샬 돌리고 생각해보자!!
- 1219 오민식의 고민 - 좀 어려운 벨만포드 문제