

## <알고리즘 실습> - 최단경로

### ※ 입출력에 대한 안내

- 특별한 언급이 없으면 문제의 조건에 맞지 않는 입력은 입력되지 않는다고 가정하라.
- 특별한 언급이 없으면, 각 줄의 맨 앞과 맨 뒤에는 공백을 출력하지 않는다.
- 출력 예시에서 □는 각 줄의 맨 앞과 맨 뒤에 출력되는 공백을 의미한다.
- 입출력 예시에서 ↪ 이 후는 각 입력과 출력에 대한 설명이다.

### 주의:

- 1) 프로그램 작성 시 사용 **데이터구조의 간편성과 효율성**은 모두 중요하다. 이 점에서 문제해결을 위해 사용한 데이터구조가 최선의 선택인지 여부는 채점 시 평가에 고려될 수 있다.
- 2) 예를 들어 그래프 알고리즘 구현 시, 그래프의 **인접 정보**(즉, 부착간선리스트 또는 인접행렬) 없이도 수행 가능한 문제라고 판단되면 **교재 13.4절의 간선리스트 구조**로 그래프를 **간편하게** 구현할 것을 우선적으로 고려하라. 그렇지 않고, **인접 정보**가 있어야 수행한다고 판단되면 **인접리스트 구조** 또는 **인접행렬 구조** 중에 해당 문제 해결에 **효율성** 면에서 유리하다고 판단되는 것을 선택하여 구현하라.

[ 문제 1 ] (**무방향 양의 가중그래프에서 최단거리 찾기**) 무방향 양의 가중그래프(undirected weighted graph) **G**와 출발정점이 주어지면, 출발정점에서 다른 모든 정점으로 가는 **최단거리**를 구하는 프로그램을 작성하라.

### 입력 그래프의 성질:

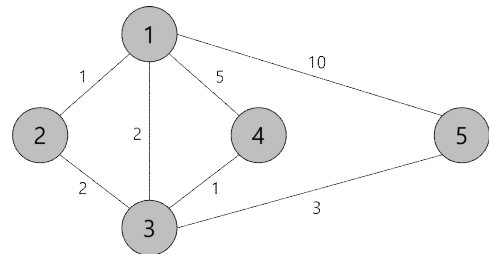
- $n(1 \leq n \leq 100)$ 개의 정점과  $m(1 \leq m \leq 1,000)$ 개의 간선으로 구성.
- 정점은 1 ~ n 사이의 정수로 번호가 매겨져 있고, 정점의 번호는 모두 다름.
- 모든 간선은 **무방향간선**이다.

### 입출력:

- 입력
  - 첫 줄에 정점의 개수 **n**, 간선의 개수 **m**, 출발정점 번호 **s**가 주어진다.
  - 이후 **m**개의 줄에 한 줄에 하나씩 간선의 정보(간선의 양 끝 정점 번호, 무게)가 주어진다. 최대로 가능한 가중치는 20을 넘지 않는다고 가정한다.  
간선은 **임의의 순서로 입력되고, 중복 입력되는 간선은 없다**(무방향간선이므로 간선 (u, v)와 (v, u)는 동일한 간선으로 취급).
- 출력
  - 출발정점 **s**에서 다른 모든 정점까지의 최단거리를 출력한다. 한 줄에 한 정점과 그 정점까지의 거리를 출력하되, 출력하는 순서는 정점의 번호의 오름차순으로 출력한다.  
도달할 수 없는 정점은 출력하지 않는다.

입력 예시 1

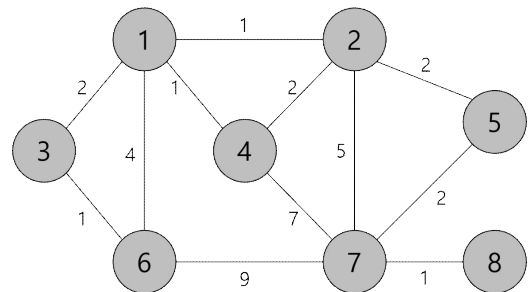
5 7 1 $\mapsto$ n=5, m=7, s=1	2 1
1 2 1	3 2
1 4 5	4 3
5 1 10	5 5
3 5 3	
4 3 1	
3 1 2	
2 3 2	



출력 예시 1

입력 예시 2

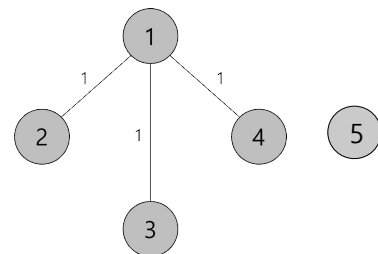
8 12 7 $\mapsto$ n=8, m=12, s=7	1 5
1 2 1	2 4
2 4 2	3 7
4 7 7	4 6
3 6 1	5 2
6 1 4	6 8
7 6 9	8 1
7 8 1	
1 3 2	
2 7 5	
1 4 1	
2 5 2	
7 5 2	



출력 예시 2

입력 예시 3

5 3 2 $\mapsto$ n=5, m=3, s=2	1 1
1 2 1	3 2
1 3 1	4 2
1 4 1	



출력 예시 3

알고리즘 설계 팁:

```

Alg DijkstraShortestPaths(G, s)
  input a simple undirected weighted graph G with nonnegative edge
        weights, a vertex s of G
  output label d(u), for each vertex u of G, s.t. d(u) is the distance
        from s to u in G

1. for each v ∈ G.vertices()
    d(v) ← ∞
2. d(s) ← 0
3. Q ← a priority queue containing all the vertices of G using d labels as
    keys
4. while (!Q.isEmpty())
    {pull a vertex into the sack}
    u ← Q.removeMin()
    for each e ∈ G.incidentEdges(u)
    {relax edge e}
    z ← G.opposite(u, e)
    if (z ∈ Q.elements())
        if (d(u) + w(u, z) < d(z))
            d(z) ← d(u) + w(u, z)
            Q.replaceKey(z, d(z))
    
```

**isEmpty**, **removeMin**, **replaceKey** 등 우선순위 큐 관련 알고리즘 설계는 교재 6장  
 힙으로 구현한 우선순위 큐의 내용을 참고할 것.

무한대값( $\infty$ ) 설정은 최대가중치  $\times$  최대간선 수를 초과하는 충분히 큰 값(예,  $30 \times 1000 = 30000$ )으로 하면 된다.

[ 문제 2 ] (**방향 가중그래프에서 최단거리 찾기**) 방향 가중그래프(directed weighted graph)  
 G와 출발정점이 주어지면, 출발정점에서 다른 모든 정점으로 가는 **최단거리**를  
 구하는 프로그램을 작성하라.

입력 그래프의 성질:

- $n(1 \leq n \leq 100)$ 개의 정점과  $m(1 \leq m \leq 1,000)$ 개의 간선으로 구성.
- 정점은 1 ~ n 사이의 정수로 번호가 매겨져 있고, 정점의 번호는 모두 다름.
- 모든 간선은 **방향간선**이고, 무게를 가진다(음의 가중치도 허용).
- 음의 사이클을 가지는 그래프는 입력되지 않는다고 가정.

입출력:

- 입력
  - 첫 줄에 정점의 개수 n, 간선의 개수 m, 출발정점 번호 s가 주어진다.
  - 이후 m개의 줄에 한 줄에 하나씩 간선의 정보(간선의 양끝 정점 번호, 무게)가  
 주어진다. 가중치의 양의 최대값은 20을 넘지 않는다고 가정한다.

- 간선은 임의의 순서로 입력되고, 중복 입력되는 간선은 없다(방향간선이므로 간선 (u, v)와 (v, u)는 다른 간선으로 취급)

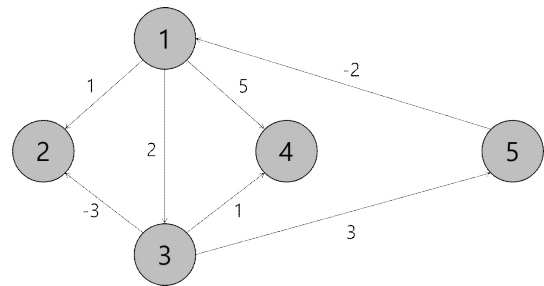
• 출력

- 출발정점 s에서 다른 모든 정점으로의 최단거리를 출력한다. 한 줄에 한 정점과 그 정점까지의 거리를 출력하되, 출력하는 순서는 정점의 번호의 오름차순으로 출력한다.  
도달할 수 없는 정점은 출력하지 않는다.

입력 예시 1

5 7 1	↦ n=5, m=7, s=1	2 -1
1 2 1		3 2
1 4 5		4 3
5 1 -2		5 5
3 5 3		
3 4 1		
1 3 2		
3 2 -3		

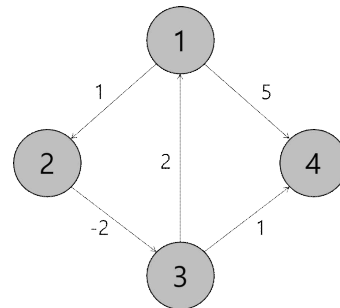
출력 예시 1



입력 예시 2

4 5 1	↦ n=2, m=2, s=1	2 1
1 2 1		3 -1
2 3 -2		4 0
3 1 2		
3 4 1		
1 4 5		

출력 예시 2



알고리즘 설계 팁:

**Alg BellmanFordShortestPaths(G, s)**  
input a weighted digraph G with n vertices, and a vertex s of G  
output label d(u), for each vertex u of G, s.t. d(u) is the distance from s to u in G

```

1. for each v ∈ G.vertices()
    d(v) ← ∞
2. d(s) ← 0
3. for i ← 1 to n - 1
    for each e ∈ G.edges()
        {relax edge e}
        u ← G.origin(e)
        z ← G.opposite(u, e)
        d(z) ← min(d(z), d(u) + w(u, z))

```

무한대값(∞) 설정은 최대가중치 × 최대간선 수를 초과하는 충분히 큰 값(예,  $30 \times 1000 = 30000$ )으로 하면 된다.