알고리즘 중급반 스터디

다익스트라 알고리즘(Dijkstra's Algorithm)

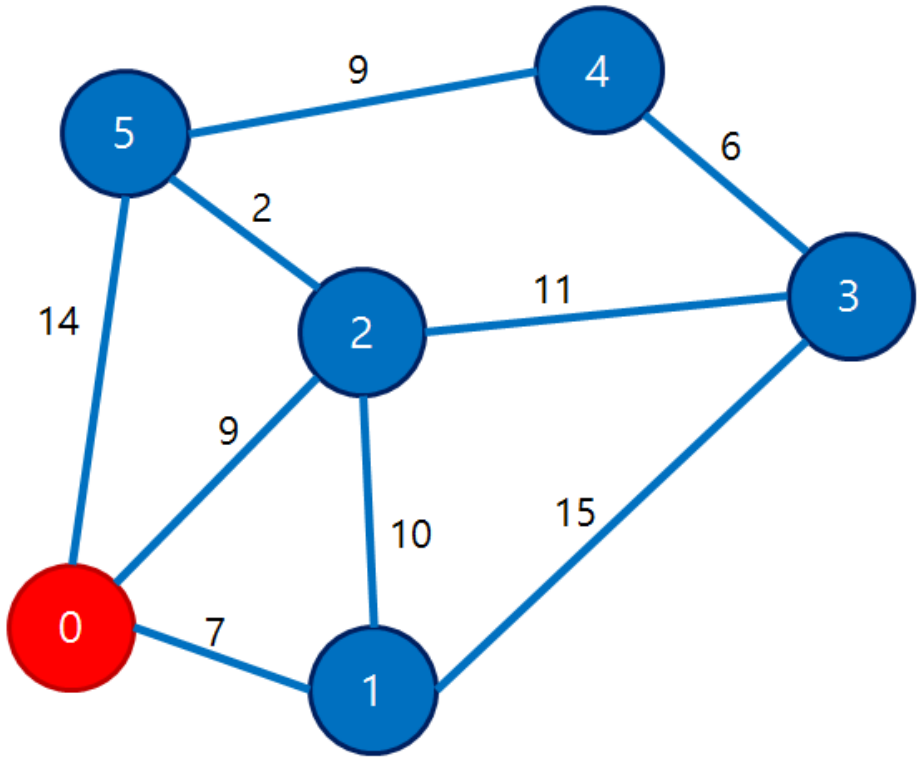
그래프 최단경로 알고리즘

- 다익스트라
- 플로이드 와샬
- 벨만포드

다익스트라 알고리즘(Dijkstra's Algorithm)

- 간선마다 이동거리(가중치)가 존재
- 모든 거리가 음수가 아닐 때
- 이동거리 대신 비용(cost)으로 용어 대체 가능 -> 최소비용 문제

다익스트라 알고리즘(Dijkstra's Algorithm)



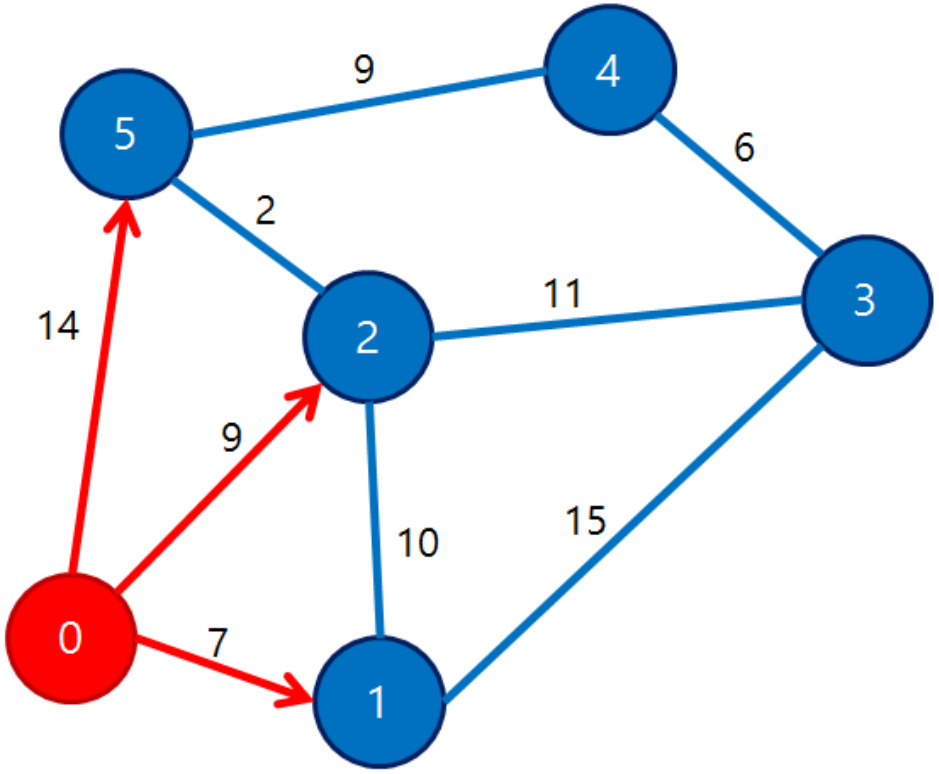
0	1	2	3	4	5
0	∞	∞	∞	∞	∞

- 방문하지 않은 정점 중 거리가 가장 짧은 정점을 방문
- 해당 정점에 인접하고 아직 방문하지 않은 정점들의 거리 갱신

* 초기 값

- 시작점의 거리 = 0
- 나머지의 거리 = 무한대

다익스트라 알고리즘(Dijkstra's Algorithm)

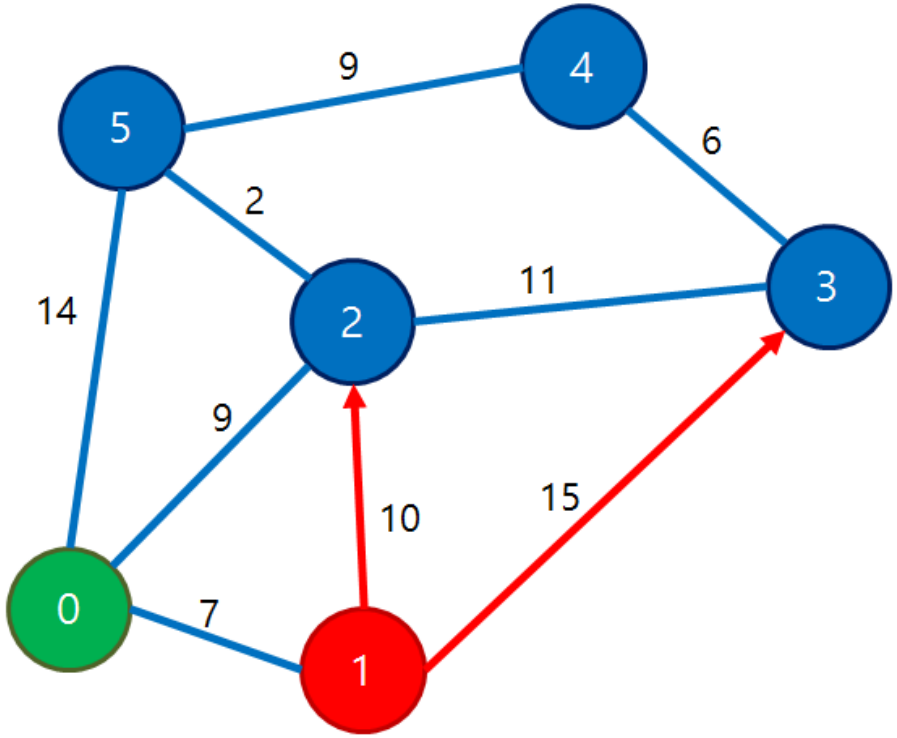


0	1	2	3	4	5
0	7	9	∞	∞	14

- Dist가 제일 작은 것은 시작점(0번)
- 0번에서 1, 2, 5번을 갈 수 있으므로 d[0]+d[0][k]를 dist[k]와 비교하여 dist[k]를 최소값으로 갱신

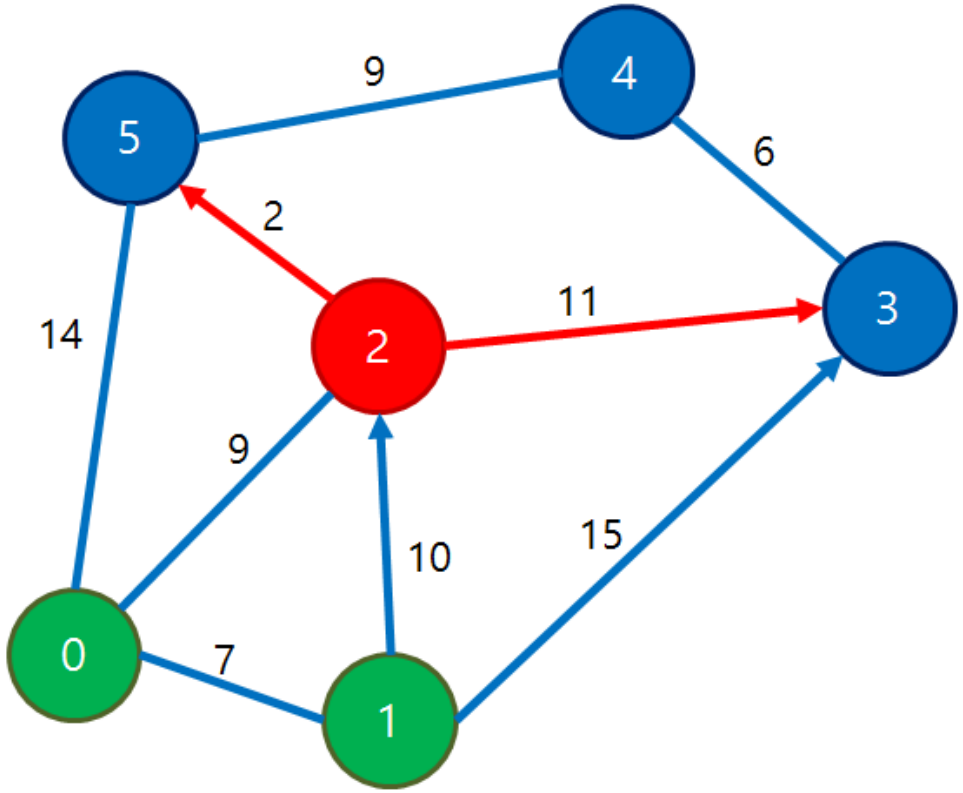
  * 현재 시작점을 제외한 모든 dist값이 무한대이므로 무조건 갱신

다익스트라 알고리즘(Dijkstra's Algorithm)



0	1	2	3	4	5
0	7	9	22	∞	14

다익스트라 알고리즘(Dijkstra's Algorithm)

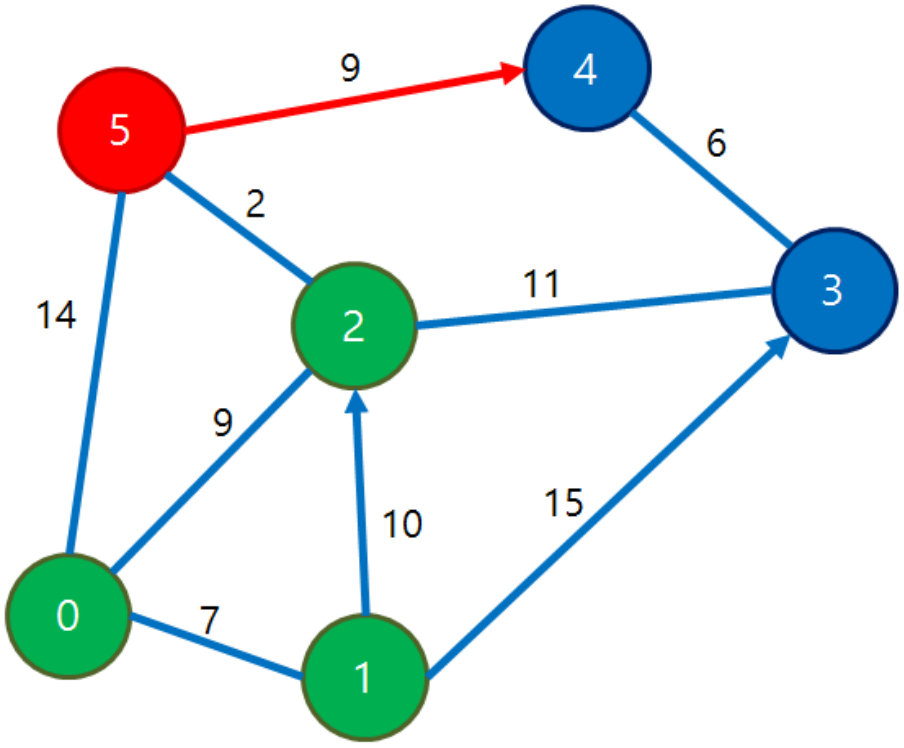


0	1	2	3	4	5
0	7	9	20	∞	11

- ㅅ

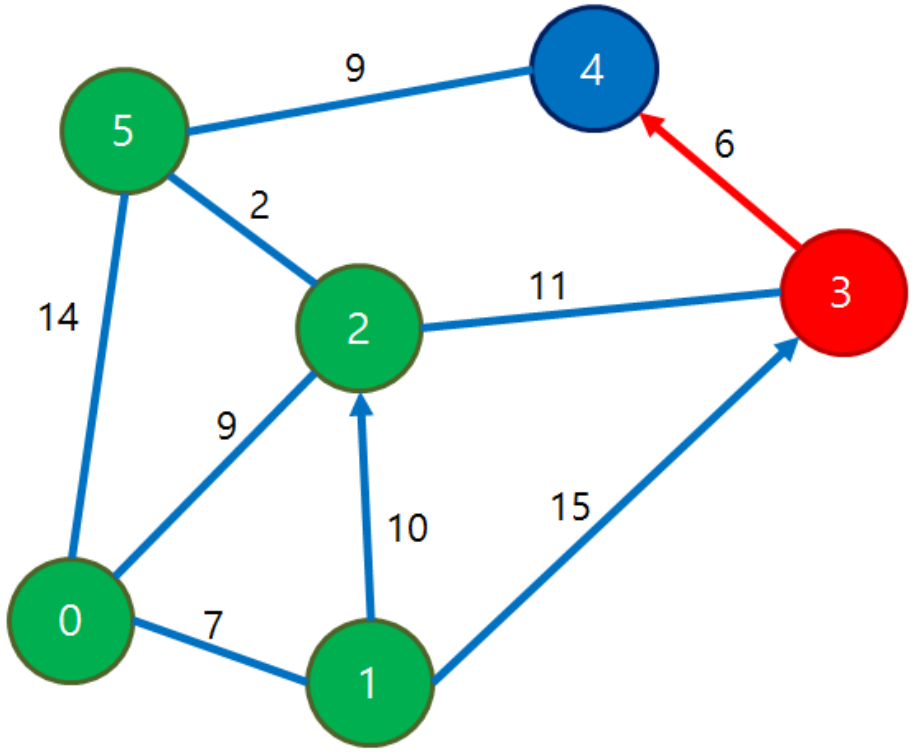
다익스트라 알고리즘(Dijkstra's Algorithm)

- ㅅ



0	1	2	3	4	5
0	7	9	20	20	11

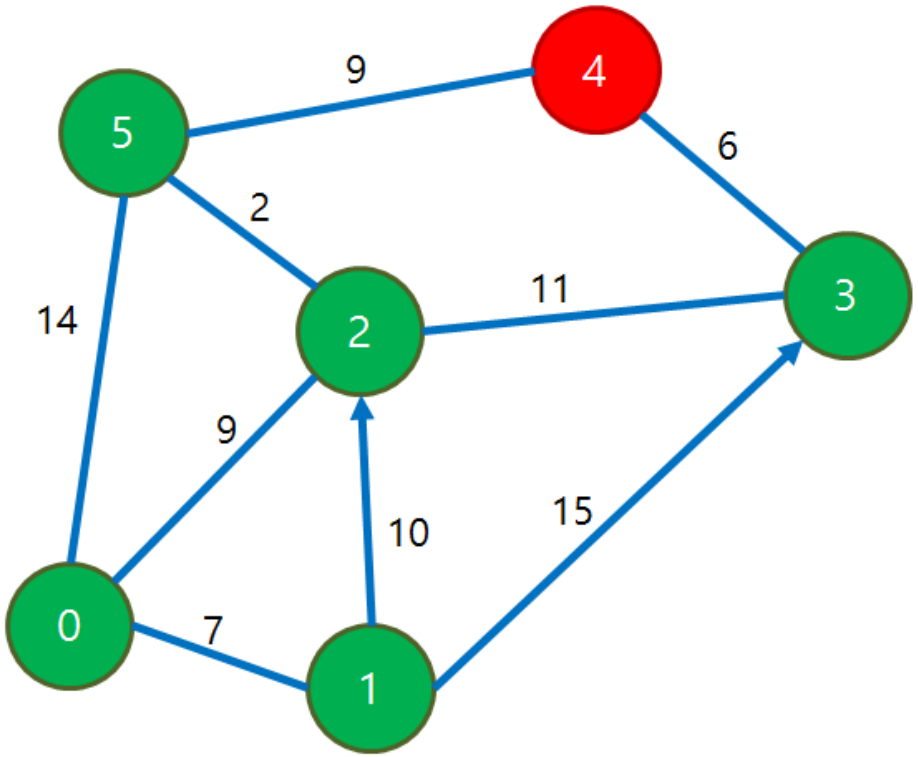
다익스트라 알고리즘(Dijkstra's Algorithm)



0	1	2	3	4	5
0	7	9	20	20	11

- 3번 4번의 dist가 같을 때는 그냥 아무거나 방문해도 됨
- 3번에서 4번의 거리 갱신을 시도했으나 실패!

다익스트라 알고리즘(Dijkstra's Algorithm)



0	1	2	3	4	5
0	7	9	20	20	11

- 모든 정점을 방문한 상태이므로 아무것도 하지 않고 끝
- 현재 dist배열에 있는 값들이 시작점(0번 정점)으로 부터의 최단 거리이다.
- 매 루프에서 방문된 상태(녹색)의 dist값은 이미 최단거리(절대 바뀌지 않음)

이게 되는게 맞을까? (d[u][v]는 0이상)

- 아직 방문하지 않은 정점 중 dist값이 제일 작은 정점이 u이고 dist[u]는 사실은 최단거리가 아니라고 가정을 해보자.(이게 성립하면 다익스트라 실패)
- 그렇다면 dist[u]가 아직 최단거리가 아니라는 말은, 다른 임의의 정점 v를 거치는 경로를 통해 최단거리로 갱신이 될 수 있다는 말이 됨
- 그럼 아직 방문하지 않은 정점 중 dist값이 제일 작은 게 u니까 v는 이미 방문을 했을 것이다.
- 그런데 v를 방문했을 때 dist[u]는 dist[v]+d [v][u]로 갱신이 되었을 것이다.
- 결국 어떤 방식으로든지 방문하지 않은 정점 중 가장 dist값이 작은 정점은 무조건 최단거리가 갱신이 되어있는 상태

다익스트라 알고리즘(Dijkstra’s Algorithm)

- Dist배열에서 가장 값이 작은 수를 찾는다. O(N)
- 간선을 보며 다음 정점으로의 값을 갱신한다. O(M) -> O(N-1)
- 시간복잡도 : O(N^2)

다익스트라 알고리즘(Dijkstra's Algorithm)

- min heap을 생성하고 거리 갱신을 할 때 마다 (dist[v], v)의 pair를 넣는다.
- 우선순위 큐에서 dist[v]를 기준으로 작은 것 부터 나온다. O(log(n))
- 우선순위 큐에서 꺼냈는데 이미 최단 거리가 확정되어 있다? -> 무시

결과적으로 O(nlogm)의 시간복잡도를 가지게 된다!

다익스트라 알고리즘(Dijkstra's Algorithm)

```cpp
void dijkstra(int sp) { // sp : 시작점의 번호
    dist[s] = 0; // 시작점의 dist를 0으로 초기화
    pq.push({ 0, sp }); // pq에 { dist, 정점 번호 } pair를 push
    while (!pq.empty()) {
        // pq에서 dist가 가장 작은 정점을 뽑음
        int cur = pq.top().second; // 해당 정점의 번호
        int curDist = pq.top().first; // 해당 정점의 dist값
        pq.pop();
        if (visit[cur]) // 이미 정점의 최단거리가 확정되었다면?
            continue;   // 무시하고 다음으로
        visit[cur] = true; // 최단거리가 확정되었다.
        // 큐에서 처음으로 빠진순간 최단거리가 확정 됨
        // 왜? 현재 방문하지 않은 정점 중 가장 dist값이 작기 때문

        for (int i = 0; i < arr[cur].size(); i++) {
            int nxt = arr[cur][i].second; // 다음 정점의 번호
            int nxtCost = arr[cur][i].first; // 다음 정점으로가는 비용
            if (dist[nxt] > curDist + nxtCost) { // nxt의 현재 dist값 보다 현재 정점을 통한 비용이 더 작다면?
                dist[nxt] = min(dist[nxt], curDist + nxtCost); // dist 갱신
                pq.push({ dist[nxt], nxt }); // pq에 push
            }
        }
    }
}
```

1753_최단경로

문제

방향그래프가 주어지면 주어진 시작점에서 다른 모든 정점으로의 최단 경로를 구하는 프로그램을 작성하시오. 단, 모든 간선의 가중치는 10 이하의 자연수이다.

입력

첫째 줄에 정점의 개수 V와 간선의 개수 E가 주어진다. ($1 \le V \le 20,000$, $1 \le E \le 300,000$) 모든 정점에는 1부터 V까지 번호가 매겨져 있다고 가정한다. 둘째 줄에는 시작 정점의 번호 K($1 \le K \le V$)가 주어진다. 셋째 줄부터 E개의 줄에 걸쳐 각 간선을 나타내는 세 개의 정수 (u, v, w)가 순서대로 주어진다. 이는 u에서 v로 가는 가중치 w인 간선이 존재한다는 뜻이다. u와 v는 서로 다르며 w는 10 이하의 자연수이다. 서로 다른 두 정점 사이에 여러 개의 간선이 존재할 수도 있음에 유의한다.

출력

첫째 줄부터 V개의 줄에 걸쳐, i번째 줄에 i번 정점으로의 최단 경로의 경로값을 출력한다. 시작점 자신은 0으로 출력하고, 경로가 존재하지 않는 경우에는 INF를 출력하면 된다.