SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

SW문제해결응용

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Day7. Dijkstra

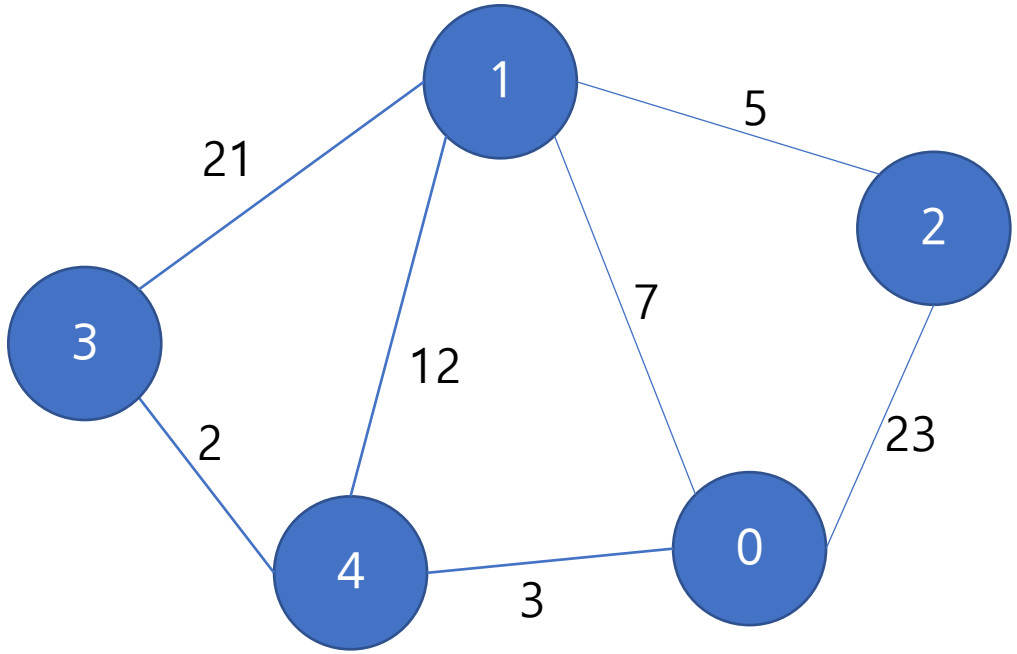
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

Dijkstra

특정 노드에서 모든 노드까지의 최단 거리를 구하는 알고리즘

- **Priority Queue 를 사용해서 간선의 가중치가 가장 작은 것을 우선으로 뽑아낸다.**

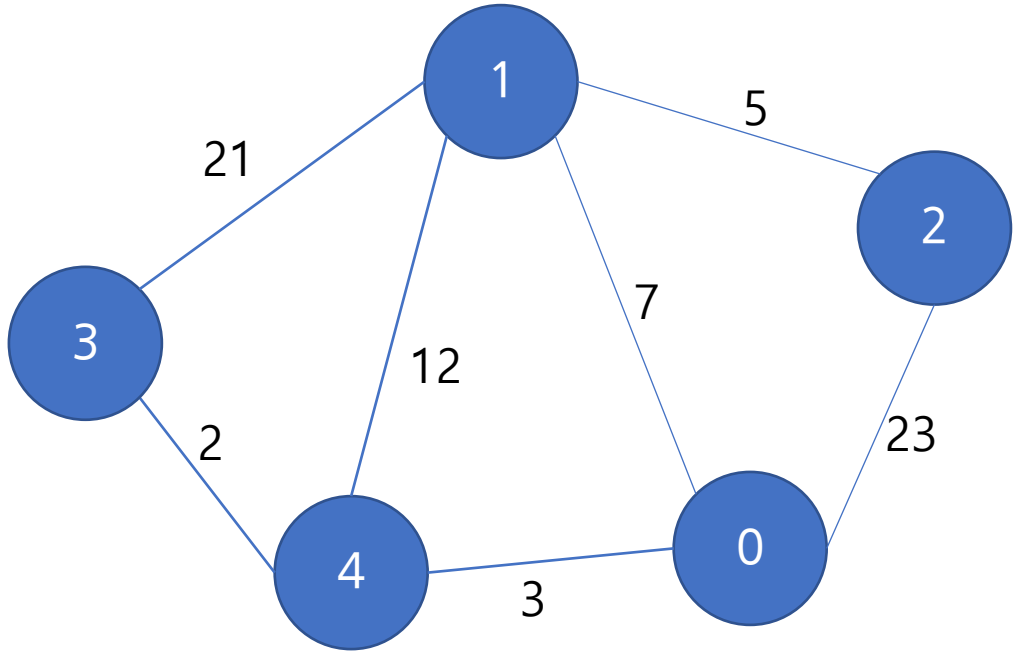


다익스트라 Dijkstra 준비

1. 간선의 가중치가 가장 작은 것을 우선 뽑을 PQ 준비
2. 시작 노드에서부터 모든 노드까지의 최단 거리를 기록할 배열
   - **누적 가중치가 기록된다.**

dist

0	1	2	3	4



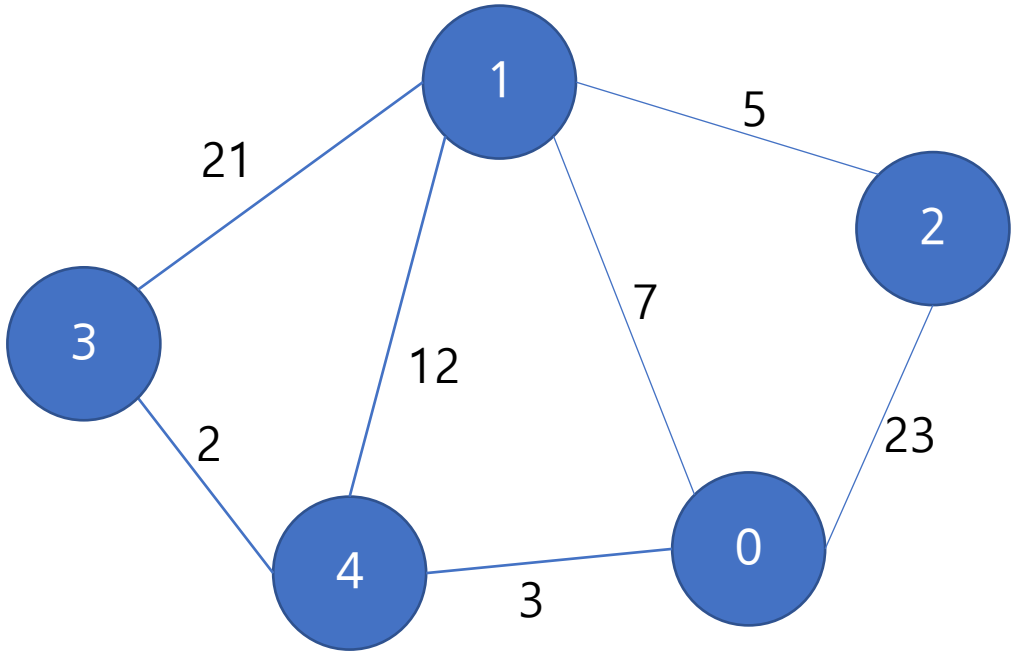
다익스트라 동작 방식 1

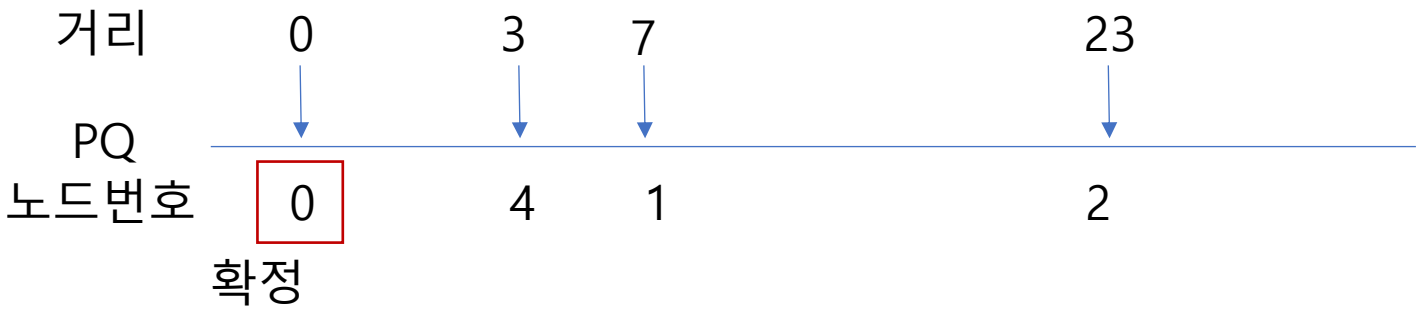
0번 노드(시작노드) 에서 갈 수 있는 노드들의 거리가 들어온다.

- **PQ에 들어오면서 순서가 나열되고, 다시 갱신된다.**

dist

0	1	2	3	4
0	7	23		3





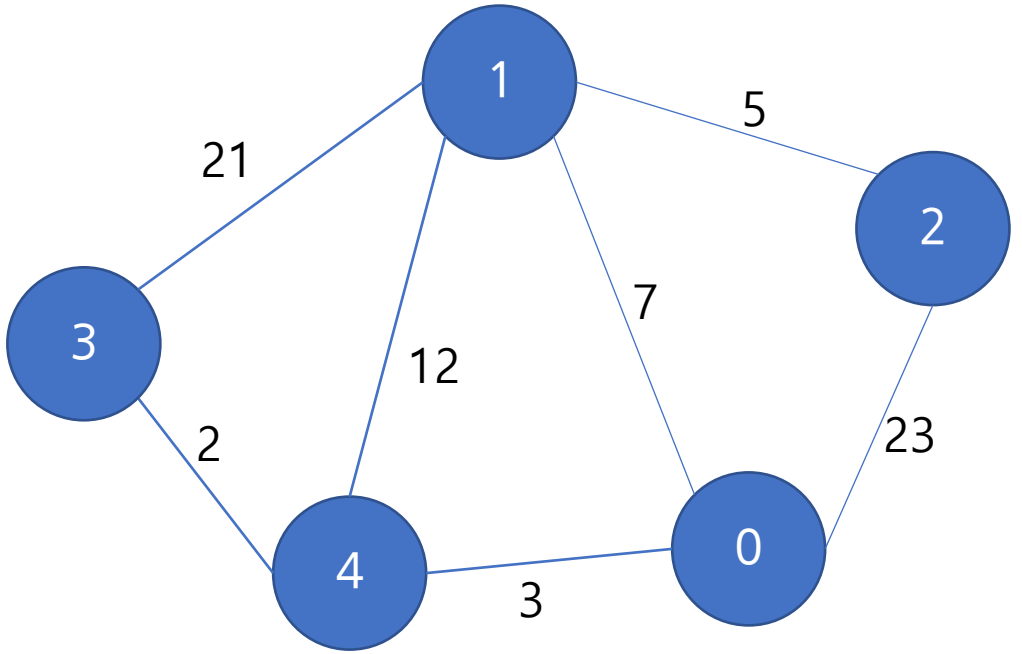
다익스트라 동작 방식 2

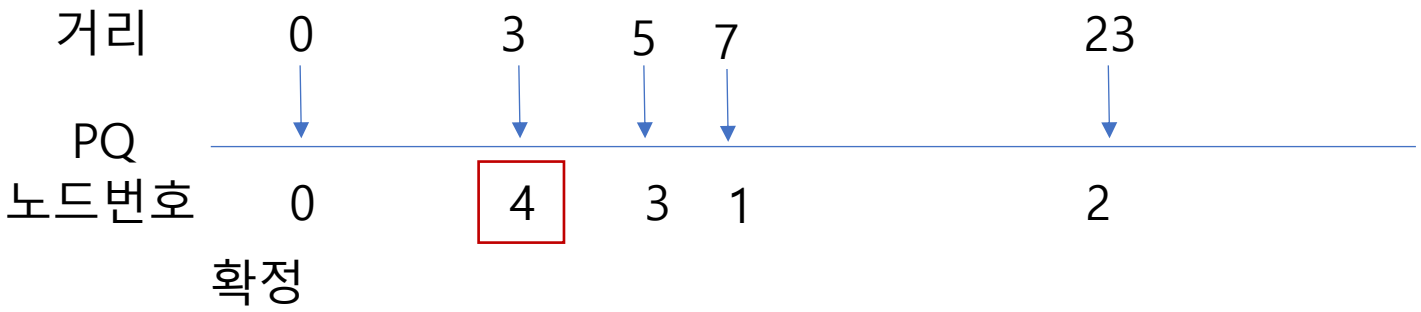
4번 노드에서 갈 수 있는 노드들의 경로가 갱신된다.

- **4 → 1 : 15 이므로 갱신 안됨**
- **4 → 3 : 3 + 2 = 5**

dist

0	1	2	3	4
0	7	23	5	3





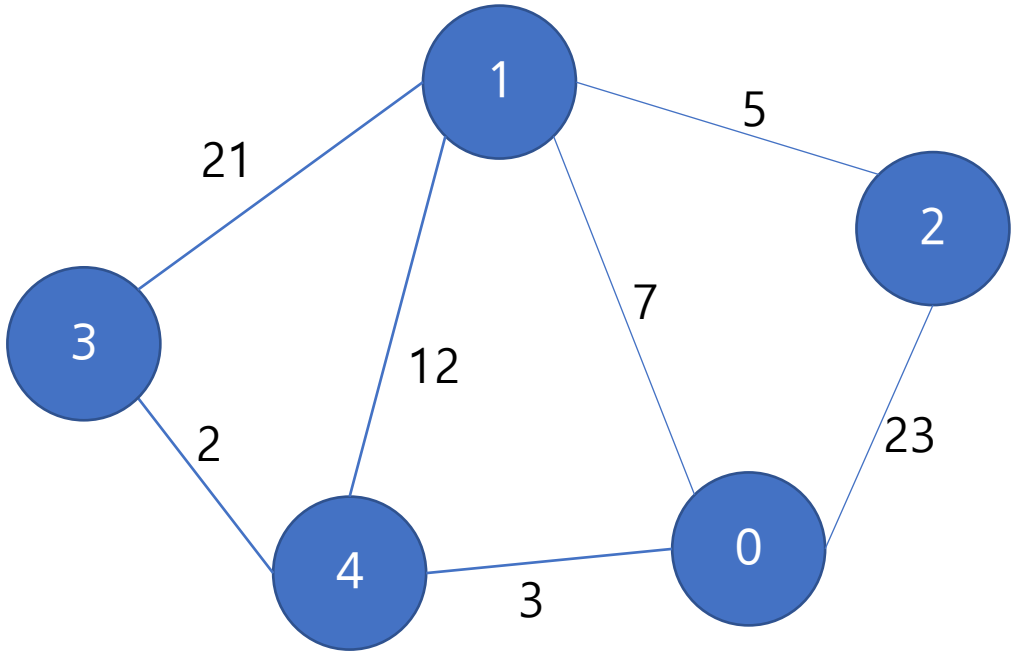
다익스트라 동작 방식 3

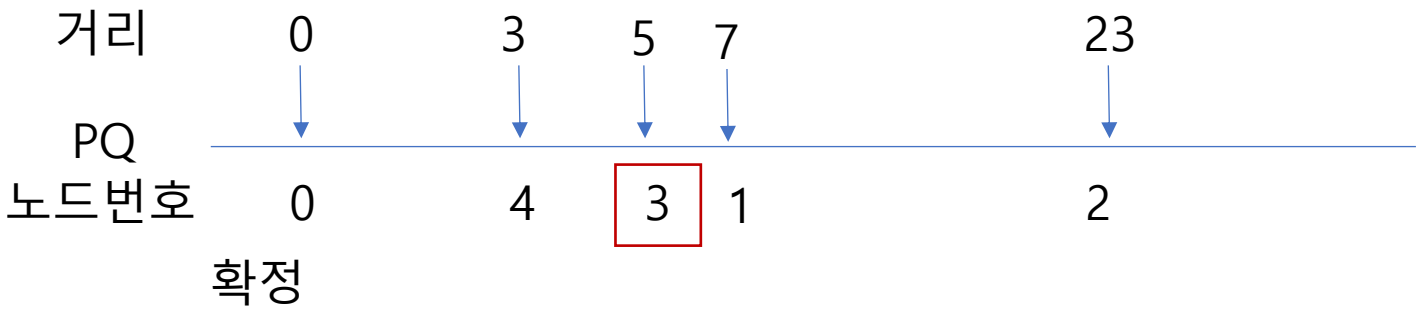
3번 노드에서 갈 수 있는 노드들의 경로가 갱신된다.

- **3 → 1 : 26, 갱신 안됨**

dist

0	1	2	3	4
0	7	23	5	3





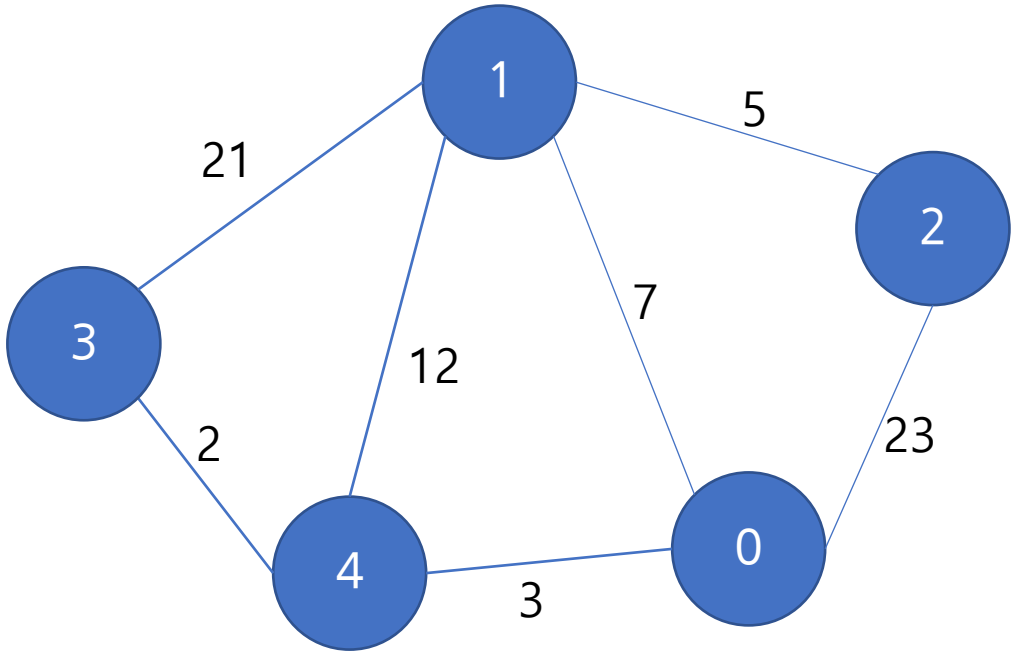
다익스트라 동작 방식 4

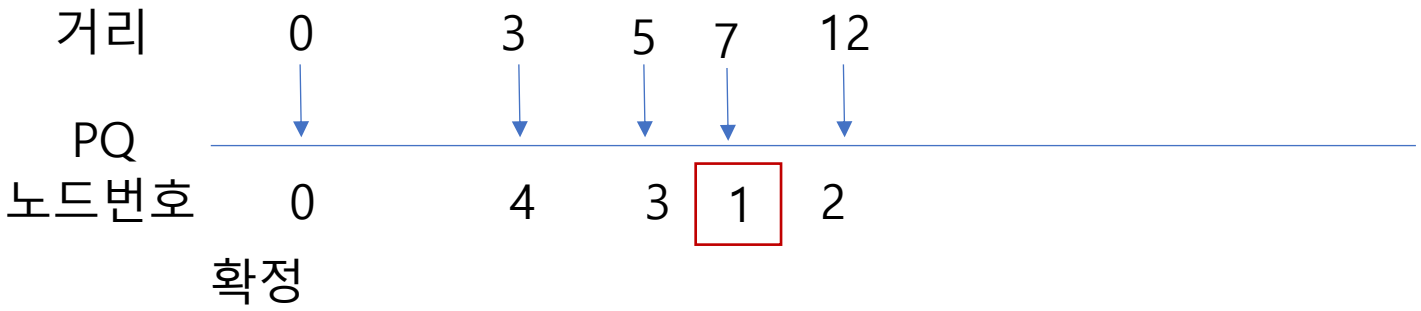
1번 노드에서 갈 수 있는 노드들의 경로가 갱신된다.

- 1 → 2 : 12 이므로 갱신
- 1 → 3 : 28이므로 갱신 X
- 1 → 4 : 19이므로 갱신 X

dist

0	1	2	3	4
0	7	12	5	3





코드로 작성하고 빌드한다.

0번 노드에서부터 최단 경로가 출력된다.


```
while (!pq.empty()) {
        //1. PQ의 top에 있는 노드 확인 및 추출 ( now )
        NODE now = pq.top(); pq.pop();

        //now까지 왔다면, 이미 거리가 확정
        //이 후 나오게 될 now들은 더 값이 작을 수 없다.
        if (dist[now.num] < now.cost)continue;

        //2. 갈 수 있는 next 후보 탐색
        for (int i = 0; i < arr[now.num].size(); i++) {
                //next 후보
                NODE next = arr[now.num][i];

                //누적합 구하기
                int nextCost = next.cost + dist[now.num];

                //해당 노드로 이동하려고 할 때,
                //이미 기록된 dist[next.cost] 와 비교해서
                //크다면 갈 필요없음
                if (dist[next.num] <= nextCost) continue;
                //next까지 최단경로 기록
                dist[next.num] = nextCost;
                //PQ 등록
                pq.push({ next.num, nextCost });
        }
}
```

https://gist.github.com/hoconoco/f9ae4cb7a42073df93a930810a5b71e2

Dijkstra의 시간복잡도 계산

기본적으로 다익스트라는 N개의 노드 간의 경로를 모두 알아야 한다.

→ **배열로 구현할 경우 N x N 회 수행 (N^2)**

하지만, 우린 Priority Queue 로 효율적으로 줄일 수 있다.

- **1) 우선순위 큐에서 최단거리 노드 추출**
  - 힙에서 최단 거리 노드 꺼내는 연산 (Heapify) logN
  - 이걸 N개 노드에서 수행 → NlogN
- **2) 인접한 노드 거리 갱신**
  - 최단 거리 노드 방문할 때마다 연결된 모든 간선을 확인
  - 총 간선 개수 E라 할 때, ElogN
- **최종 시간 복잡도(1+2) : (N+E)logN**

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

내일 방송에서 만나요!

삼성 청년 SW 아카데미