
문제 2. 다익스트라(Dijkstra) 알고리즘을 이용하여 아래에 제시된 그래프에서 최단 거리를 구하는 프로그램을 완성하려고 한다. [소스 작성 요령]에서 요구하는 사항을 구현하라. 단, 모든 문항에서 요구하는 기능들은 반드시 모범답안과 일치해야 한다. [배점 : 총 20점]

[문제설명]

[상세 조건 설명]에서 제시된 그래프와 [그림1]문제화면 예시 그리고 [그림2]최종화면 예시를 기준으로 [폴더]02_실행파일(참고용) 내의 실행파일을 참고하여 [폴더]01_문제(답안작성용) 프로젝트 내의 **///★** 주석 부분에 있는 (A), (B) 부분을 완성하시오.

[상세 조건 설명]

- ◎ 문제에 적용된 알고리즘은 다익스트라(Dijkstra) 알고리즘을 이용한 최단거리경로를 찾는 알고리즘이다.
- ◎ 그래프의 노드 수는 11개이고 **인접 매트릭스**로 표현되어 있으며, **0번 노드**가 출발점이다.
- ◎ 소스 코드 dijkstra_adj.cpp 내의 **실행화면 타이틀 상에 수험번호와 성명**이 출력되도록 수정하시오. (수정하지 않을 경우 **감점** 있음)
- ◎ 다음은 왼쪽 그래프의 노드 집합을 **인접 매트릭스**로 표현하고 있으며, **블라인드**된 부분을 정확한 거리값으로 올바르게 수정하여 **문제 (A)**를 완성하시오.

<p>그래프에서 ①은 노드 0을 나타내고 3 — 5는 간선의 거리를 나타냄</p>	<p>그래프 노드집합의 인접 매트릭스</p> <pre>int graphCost[NUM_NODE][NUM_NODE] = { { 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, { 5, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 7, 0, 2, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 5, 0, 0, 9, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0 }, { 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0 }, { 13, 9, 0, 7, 0, 15, 0, 0, 0, 0, 0 }, { 0, 0, 12, 0, 6, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } };</pre>
---------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

- ◎ 소스 코드 dijkstra_adj.cpp 내의 아래의 주석 부분을 수정하고 완성한다.

///

///★문제(A)는 소스 코드에서 0값으로 작성된 부분을 그래프에 맞게 수정하여 완성한다.

///

- ◎ 소스 코드 dijkstra_adj.cpp 내의 **///★문제(B)** 주석 부분의 지시에 맞게 수정하고 완성하시오

///

///★문제(B)

///

/// - 다음은 최단거리 값 갱신을 위한 조건식이다.

```

/// 1. 최단경로 트리 상의 아직 처리가 되지 않은 노드일 것
/// 2. curNode와 node간에 연결이 되어 있어야 할 것(비용 존재하는 간선일 것)
/// 3. Src로부터 curNode까지 경로가 존재해야 할 것(무한대가 아닐 것)
/// 4. 기존노드까지의 최단거리 값보다 새 간선거리 값이 추가되어도 더 작아 갱신(Update)이 필요한 경우
/// 갱신이 되도록( B )를 수정하여 완성하시오.

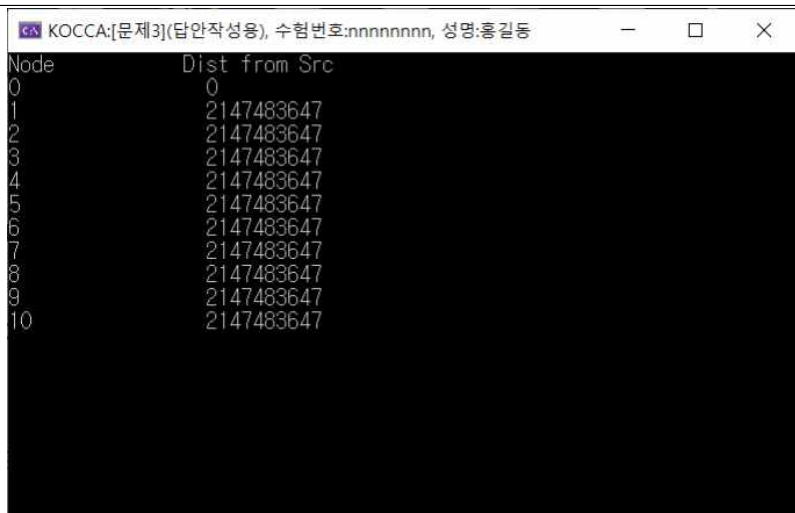
```

```

if ((!shortestPathTree[node]) &&
    edgeCost[curNode][node] &&
    distFromSrc[curNode] != INT_MAX &&
    distFromSrc[curNode] + edgeCost[curNode][node] < distFromSrc[node])
{
    distFromSrc[node]; /*( B ) */
}

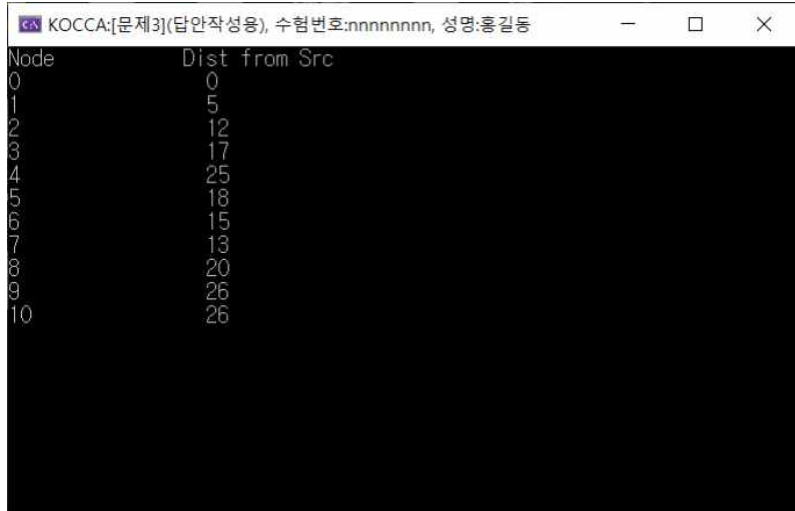
```

[그림1] 문제화면 예시



Node	Dist from Src
0	0
1	2147483647
2	2147483647
3	2147483647
4	2147483647
5	2147483647
6	2147483647
7	2147483647
8	2147483647
9	2147483647
10	2147483647

[그림2] 최종화면 예시



Node	Dist from Src
0	0
1	5
2	12
3	17
4	25
5	18
6	15
7	13
8	20
9	26
10	26