실습과제 10.2

[과제 설명]

그래프에서 최단 경로를 찾는 문제(Shortest Path Problem)에는 입력으로 (음수가 아닌) 가중치 그래프가 주어지면 한 시작 정점에서 모든 정점까지의 최단 비용을 계산하는 문제이다. 이 문제는 해결하기 위해 다익스트라 알고리즘(Dijkstra algorithm)을 사용한다.

다익스트라 알고리즘은 임의의 정점에서 다른 모든 정점으로 갈 수 있는 최단 경로를 계산한다. 이때, 최단 거리는 여러 개의 최단 거리의 집합으로 이루어진다. 다익스트라 알고리즘을 계산하기 위해, 다음으로 변수의 사용법을 정의한다.

-distance[u]: 시작 정점 v → 도착 정점 u까지의 최소 비용 거리
-weight[i][j]: 가중치 그래프에서 간선 (i, j)의 비용(가중치)
-S: 시작 정점 v에서 해당 목적지까지 이미 최단 경로가 발견된 정점들의 집합

다익스트라 알고리즘은 다음의 순서로 동작한다.

1. 출발 정점을 선정한다.
2. 출발 정점으로부터 각 정점까지의 최단 거리를 distance에 저장한다. (인접하지 않은 노드는 최소값 비교를 위해, 저장할 수 있는 최대값(INF, C에서는 INT_MAX)로 distance에 저장한다.)
3. 반복문을 이용해서 인접 정점 중 방문하지 않는 정점을 탐색하고, 그 중 가장 비용이 적은 정점을 선택해서, 방문 표시를 한다.
4. 해당 정점을 거쳐서 특정 정점으로 가는 경우를 고려해서 최단 거리 배열을 갱신한다.
5. 목표 정점에 도착할 때까지 3~4를 반복한다.

[코드]

메인 함수와 header 파일, 출력/변환 함수는 이전 실습과교안에서 미리 구현된 것을 사용했으므로 보고서에는 생략하고 새로 구현한 함수 세부 내용만 첨부한다.

```
void Shortest_Path(int weight_mat[][MAX_VERTICES], int n, int v, int* path) {
  int* distance = new int[n];
  int* visited = new int[n];

  for (int i = 0; i < n; i++) {
    distance[i] = weight_mat[v][i];
    path[i] = v;
    visited[i] = 0;
  }

  visited[v] = 1;
  distance[v] = 0;
  int step = 1;

  while (step < n) {
    int k;
    int min = INT_MAX; //해당 변수가 가질 수 있는 최대 값
    for (int i = 0; i < n; i++) {
      //방문하지 않은 정점 중 가장 가까운 정점을 찾는다.
      if (visited[i] == 0 && distance[i] < min) {
        min = distance[i];
        k = i;
      }
    }

    visited[k] = 1; //방문한 정점은 1로 표시

    for (int u = 0; u < n; u++) {
      if (visited[u] == 0) {
        if (weight_mat[k][u] == INT_MAX) continue;
        //최대값에 더하면 오버플로우가 일어나기 때문에 최대값이면 아래를 실행하지 않고 건너뛴다.
        if ((distance[k] + weight_mat[k][u]) < distance[u]) {
          //정점 u를 경유했다면, 최단 거리가 있을 때 업데이트
          distance[u] = distance[k] + weight_mat[k][u];
          path[u] = k;
        }
      }
    }
    step++;
  }

  for (int i = 0; i < n; i++) {
    printf("정점 0 -> 정점 %d: %d\n", i, distance[i]);
  }
}
```

[실행 결과]

```
##가중치 그래프##
        0    1    2    3    4    5    6
   0    0    5    6   10  INF  INF  INF
   1    5    0  INF    7    8  INF  INF
   2    6  INF    0    8  INF    7  INF
   3   10    7    8    0    2    6   10
   4  INF    8  INF    2    0  INF    5
   5  INF  INF    7    6  INF    0    9
   6  INF  INF  INF   10    5    9    0
##최단 거리##
정점 0 -> 정점 0: 0
정점 0 -> 정점 1: 5
정점 0 -> 정점 2: 6
정점 0 -> 정점 3: 10
정점 0 -> 정점 4: 12
정점 0 -> 정점 5: 13
정점 0 -> 정점 6: 17

##최단 거리 경로##
0
0 -> 1
0 -> 2
0 -> 3
0 -> 3 -> 4
0 -> 2 -> 5
0 -> 3 -> 4 -> 6
```

[과제에 대한 고찰]

처음에 min에 (C에서 구현할 수 있는) 최대값을 넣어두어서 distance[k] + weight_mat[k][u] 조건문으로 오버플로우가 돼서 음수로 변한 탓에 조건문이 참으로 인식되고, 그 다음 distance[u] = distance[k] + weight_mat[k][u]; 에 음수로 대입되는지 모르고 주소라고 생각했다. 코드를 고치기 전에 정점 0에서 정점 5, 정점 0에서 정점 6이 같은 음수 값으로 나와서 의아했는데 이런 이유 때문이었다. 그래서 만약에 distance가 계속 최대값이면 아래의 내용을 실행하지 말고 건너뛰라고 continue;를 넣었는데 다행히 제대로 되었다.

다른 과목에서 먼저 다익스트라 알고리즘을 배워서 손으로 계산할 때는 직접 계산을 한데다가, 오버플로우가 일어날 수가 없기 때문에 고려하지 않았는데 이런 것도 신경써야겠다는 생각을 하게 되었다.