실습과제 10.1

[과제 설명]

신장 트리는 그래프에서 모든 정점을 포함하는 (사이클이 존재하지 않는) 트리이다. 이 중 최소 비용 신장 트리(MST, Minimum Spanninb Tree)는 가중치 그래프가 주어질 경우 간선들의 비용 합이 최소가 되는 신장 트리다.

최소 비용 신장 트리의 조건으로는 그래프에 존재하는 모든 정점을 포함해야 하고, 신장 트리에 포함되는 간선의 가중치 함은 최소가 되어야 하고, 두 정점 간의 경로가 존재해야 한다는 것이다.

최소 비용 신장 트리를 구성하는 방법에는 프림 알고리즘(Prim's Algorithm)과 크루스칼 알고리즘(Kruskal's Algorithm)이 있는데, 이번 실습에서는 프림 알고리즘을 구현한다.

먼저 가중치 그래프가 주어지면, 인접행렬로 표현한 뒤 알고리즘을 실행한다. 인접행렬은 두 정점 사이에 직접적인 간선이 존재하면 해당하는 간선이 가진 가중치를 적고, 직접적인 간선이 없다면 무한대를 적어넣는다.

프림 알고리즘을 이용한 최소 비용 신장 트리는 다음과 같이 구성된다.

1. 임의의 시작 정점을 선정하고, 최소 비용 신장 트리에 정점 0을 추가한다.
2. 현재 MST에 추가된 정점들의 인접 정점 사이의 가중치를 조사하고, 가장 낮은 가중치를 가진 간선과 정점을 선택해서 MST에 추가한다.
3. 이것을 모든 정점이 포함될 때까지 반복한다.

[코드]

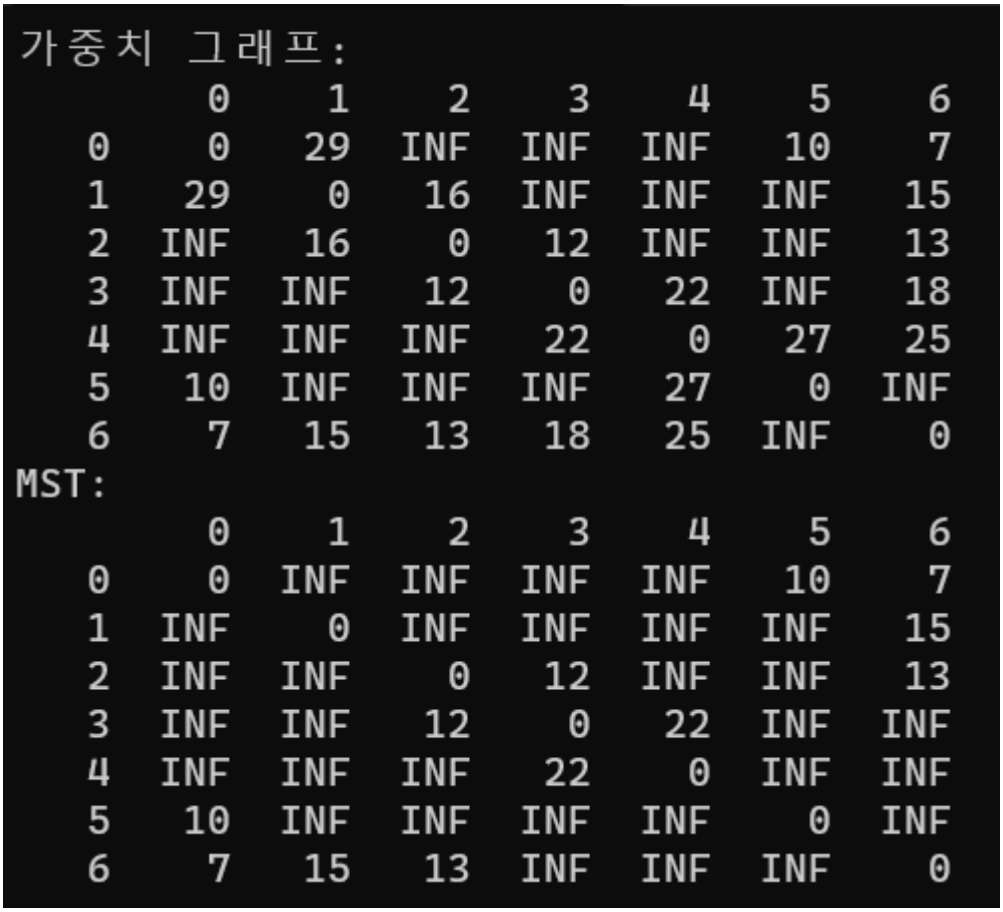
메인 함수와 header 파일은 교안에서 미리 구현된 것을 사용했으므로 보고서에는 생략하고 구현된 함수 세부 내용만 첨부한다.

```
#include "MyGraphMST.h"

//파일에 저장된 데이터로 가중치 그래프 구성
```

```
void ADJ_Create(const char* g_file, int weight_mat[][MAX_VERTICES]) {
  FILE* file;
  fopen_s(&file, g_file, "r"); //read mode
  if (file == NULL) {
    printf("파일이 없습니다. 프로그램을 종료합니다.");
    return;
  }
  while (1) {
    int i, j, k;
    int result = fscanf_s(file, "%d, %d, %d", &i, &j, &k);
    if (result == EOF) break;
    weight_mat[i][j] = k;
    weight_mat[j][i] = k;
  }
  fclose(file);
}
//출력함수
void ADJ_Print_Array(int weight_mat[][MAX_VERTICES], int n) {
  for (int i = -1; i < n; i++) {
    for (int j = -1; j < n; j++) {
      if (i == -1) {
        if (j != -1) printf(" %4d", j);
        else printf("    ");
      }
      else {
        if (j == -1) printf(" %3d ", i);
        else {
          if (weight_mat[i][j] == INT_MAX)
            printf(" INF ");
          else
            printf(" %3d ", weight_mat[i][j]);
        }
      }
    }
    printf("\n");
  }
}
//프림 알고리즘 기반 최소비용 신장트리
void MST_Prim(int weight_mat[][MAX_VERTICES], int MST[][MAX_VERTICES], int n) {
  int* selected = malloc(sizeof(int) * n);
  //MST에 선택된 정점을 표시한다.

  for (int i = 0; i < n; i++) {
    selected[i] = 0;
  }
  //calloc으로 할당과 동시에 초기화도 가능

  int edge_count = 0;
  selected[0] = 1;
  //시작 정점을 정점 0으로 설정한다.

  while (edge_count < n - 1) {
    int min = INT_MAX; //해당 변수가 가질 수 있는 최대 값 설정
    int u = 0;
    int v = 0;

    for (int i = 0; i < n; i++) {
      if (selected[i] == 1) { //이미 MST에 포함된 정점 중에서
```

```
        for (int j = 0; j < n; j++) {
          if (selected[j] == 0 && min > weight_mat[i][j]) {
            //선택되지 않은 다른 정점이나, 최소값보다 작은 경우
            min = weight_mat[i][j]; //새로운 최소값으로 설정
            u = i;
            v = j;
          }
        }
      }
    }
    selected[v] = 1; //새로운 정점을 선택 완료로 표시
    MST[u][v] = weight_mat[u][v];
    MST[v][u] = weight_mat[u][v];
    //인접행렬은 대칭행렬이기 때문에 같은 값 삽입
    edge_count++;
  }
}
```

[실행 결과]



[과제에 대한 고찰]

만약에 최소 비용 신장 트리를 구성할 때, 간선의 가중치 때문에 가장 작은 간선을 선택하느라 사이클이 만들어지면 어떻게 될까 라는 생각을 하게 되었다. 그러다, 차근차근하게 다시 알고리즘 부분을 짚어보았더니 이미 선택된 정점이 있으면 해당하는 건 피할 수 있는 부분이 있었다.

또, 다른 과목을 공부할 때 판매원 방문 문제(Travelling salesman problem, TSP)에 대해서도 흥미롭게 보았는데, 이 문제는 순회라서 사이클이 생기는 걸 제외하면 최소 비용 신장 트리와 유사한 부분을 가진 알고리즘이 나올 것 같다는 생각을 하게 되었다.