EECE 7205-Assignment 4

Sreejith Sreekumar: 001277209

November 7, 2018

1 Qn1. Prims Algorithm: Adjacency Matrix Implementation

1.1 Code

```
#include <stdio.h>
#include <limits.h>
#include <iostream>


int get_min_key(int key[], int priority_queue[], int vertex_count){

  int min = 9999, min_index;

  for (int v = 0; v < vertex_count; v++) {
    if (priority_queue[v] != -1 && key[v] < min) {
      min = key[v], min_index = v;
    }
  }

  return min_index;

}


void print_min_span_tree(int parent[],
                         int vertex_count,
                         int **graph) {

  printf("Edge  \tWeight of Edge\n");

  for (int i = 1; i <  vertex_count; i++) {
    printf("%d - %d \t%d \n", parent[i], i, graph[i][parent[i]]);
  }

}
```

```
void prims_min_span_tree(int** graph, int vertex_count){

  /**
   * Parent's and keys
   */
  int parent[vertex_count];
  int key[vertex_count];

  int priority_queue[vertex_count];

  /**
   * Initially set all keys to infinity - A large value instead
   * All values in priority_queue are initially filled with vertex ids
   * when used up we set it to -1
   */
  for (int i = 0; i < vertex_count; i++) {
    key[i] = 9999;
    priority_queue[i] = i;
  }

  /**
   * Setting first node to have a key 0 and
   * no parent
   */
  key[0] = 0;
  parent[0] = -1;

  int count=0;
  while(count < vertex_count-1){

    /**
     * Get element in the priority queue with minimum key
     * We are using u - set priority_queue to -1
     */
    int u = get_min_key(key, priority_queue, vertex_count);
    priority_queue[u] = -1;


    for (int v = 0; v < vertex_count; v++)

      /**
       * Conditions:
       * (i) graph[u][v] has a non zero value if u and v are adjacent
       * (ii) v has to be present in the priority queue
       * (iii) weight(u,v) has to be smaller than the key[v]
       */
      if (graph[u][v] && priority_queue[v] != -1 && graph[u][v] < key[v]) {
        parent[v] = u, key[v] = graph[u][v];
```

```
      }

    count++;
  }

  print_min_span_tree(parent, vertex_count, graph);
}



int main() {

  int vertices;

  std::cout<<"Enter the number of vertices: ";
  std::cin>>vertices;

  int **_graph = new int*[vertices];

  for(int i=0; i<vertices; i++){
    _graph[i] = new int[vertices];
  }

  for(int i=0; i<vertices; i++){
    std::cout<<"Enter the connections for vertex "<<i<<" (Please separate weights by a space): ";
    for(int j=0; j<vertices; j++){
      std::cin>>_graph[i][j];
    }
  }

  // {{0, 2, 0, 6, 0},
  //  {2, 0, 3, 8, 5},
  //  {0, 3, 0, 0, 7},
  //  {6, 8, 0, 0, 9},
  //  {0, 5, 7, 9, 0}};

  prims_min_span_tree(_graph, vertices);

}
```

1.2 Output