데이터 구조 과제

학번: 12170584

이름: 이진호

**Kruskal algorithm**

**문제정의:**

최소비용 신장트리(minimum spanning tree)를 구하는 알고리즘의 하나인 크루스칼(kruskal)알고리즘이다. stdin에서 input graph 데이타를 입력받으며, 프로그램 수행 후 적절한 답을 stdout에 출력.

**코드 설명:**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <iostream>  
  
using namespace std;  
  
enum {*a*,*b*,*c*,*d*,*e*,*f*,*g*};  
  
// a structure to represent a weighted edge in graph  
struct Edge  
{  
 int src, dest, weight;  
};  
  
// a structure to represent a connected, undirected  
// and weighted graph  
struct Graph  
{  
 // V-> Number of vertices, E-> Number of edges  
 int V, E;  
  
 // graph is represented as an array of edges.  
 // Since the graph is undirected, the edge  
 // from src to dest is also edge from dest  
 // to src. Both are counted as 1 edge here.  
 struct Edge\* edge;  
};  
  
// Creates a graph with V vertices and E edges  
struct Graph\* createGraph(int V, int E)  
{  
 struct Graph\* graph = new Graph;  
 graph->V = V;  
 graph->E = E;  
  
 graph->edge = new Edge[E];  
  
 return graph;  
}  
  
// A structure to represent a subset for union-find  
struct subset  
{  
 int parent;  
 int rank;  
};  
  
// A utility function to find set of an element i  
// (uses path compression technique)  
int find(struct subset subsets[], int i)  
{  
 // find root and make root as parent of i  
 // (path compression)  
 if (subsets[i].parent != i)  
 subsets[i].parent = find(subsets, subsets[i].parent);  
  
 return subsets[i].parent;  
}  
  
// A function that does union of two sets of x and y  
// (uses union by rank)  
void Union(struct subset subsets[], int x, int y)  
{  
 int xroot = find(subsets, x);  
 int yroot = find(subsets, y);  
  
 // Attach smaller rank tree under root of high  
 // rank tree (Union by Rank)  
 if (subsets[xroot].rank < subsets[yroot].rank)  
 subsets[xroot].parent = yroot;  
 else if (subsets[xroot].rank > subsets[yroot].rank)  
 subsets[yroot].parent = xroot;  
  
 // If ranks are same, then make one as root and  
 // increment its rank by one  
 else  
 {  
 subsets[yroot].parent = xroot;  
 subsets[xroot].rank++;  
 }  
}  
  
// Compare two edges according to their weights.  
// Used in qsort() for sorting an array of edges  
int myComp(const void\* a, const void\* b)  
{  
 struct Edge\* a1 = (struct Edge\*)a;  
 struct Edge\* b1 = (struct Edge\*)b;  
 return a1->weight > b1->weight;  
}  
  
// The main function to construct MST using Kruskal's algorithm  
void KruskalMST(struct Graph\* graph)  
{  
 int V = graph->V;  
 struct Edge result[V]; // Tnis will store the resultant MST  
 int e = 0; // An index variable, used for result[]  
 int i = 0; // An index variable, used for sorted edges  
  
 // Step 1: Sort all the edges in non-decreasing  
 // order of their weight. If we are not allowed to  
 // change the given graph, we can create a copy of  
 // array of edges  
 qsort(graph->edge, graph->E, sizeof(graph->edge[0]), myComp);  
  
 // Allocate memory for creating V ssubsets  
 struct subset \*subsets =  
 (struct subset\*) malloc( V \* sizeof(struct subset) );  
  
 // Create V subsets with single elements  
 for (int v = 0; v < V; ++v)  
 {  
 subsets[v].parent = v;  
 subsets[v].rank = 0;  
 }  
  
 // Number of edges to be taken is equal to V-1  
 while (e < V - 1)  
 {  
 // Step 2: Pick the smallest edge. And increment  
 // the index for next iteration  
 struct Edge next\_edge = graph->edge[i++];  
  
 int x = find(subsets, next\_edge.src);  
 int y = find(subsets, next\_edge.dest);  
  
 // If including this edge does't cause cycle,  
 // include it in result and increment the index  
 // of result for next edge  
 if (x != y)  
 {  
 result[e++] = next\_edge;  
 Union(subsets, x, y);  
 }  
 // Else discard the next\_edge  
 }  
  
 // print the contents of result[] to display the  
 // built MST  
 printf("Following are the edges in the constructed MST\n");  
 for (i = 0; i < e; ++i)  
 printf("%c -- %c == %d\n", result[i].src+97, result[i].dest+97,  
 result[i].weight);  
 return;  
}  
  
// Driver program to test above functions  
int main()  
{  
 /\* Let us create following weighted graph  
 10  
 a--------b  
 | \ |  
 6| 5\ |15  
 | \ |  
 c--------d  
 4 \*/  
 int V = 4; // Number of vertices in graph  
 int E = 5; // Number of edges in graph  
 struct Graph\* graph = createGraph(V, E);  
  
 graph->edge[0].src = *a*;  
 graph->edge[0].dest = *b*;  
 graph->edge[0].weight = 10;  
 graph->edge[1].src = *b*;  
 graph->edge[1].dest = *d*;  
 graph->edge[1].weight = 15;  
 graph->edge[2].src = *a*;  
 graph->edge[2].dest = *c*;  
 graph->edge[2].weight = 6;  
 graph->edge[3].src = *a*;  
 graph->edge[3].dest = *d*;  
 graph->edge[3].weight = 5;  
 graph->edge[4].src = *c*;  
 graph->edge[4].dest = *d*;  
 graph->edge[4].weight = 4;  
  
 KruskalMST(graph);  
  
 for(int i=0;i<E;i++){  
 int src,dest;  
 int weight;  
 printf("please input src, dest, weight: ");  
  
 cin >> src >> dest >> weight;  
 cout << src << dest << weight;  
  
 graph->edge[0].src = src;  
 graph->edge[0].dest = dest;  
 graph->edge[0].weight = weight;  
 }  
   
  
 KruskalMST(graph);  
  
 return 0;  
}

**결론 – 크루스칼 알고리즘을 통해 최단 경로를 구하려고 하였으나 입력에 따른 출력을 구현 하여야 하는데, 원하는 결과를 이뤄낼 수 없었다.**