HW6_ minimum spanning tree problem

Kruskal’ s

程式摘要:

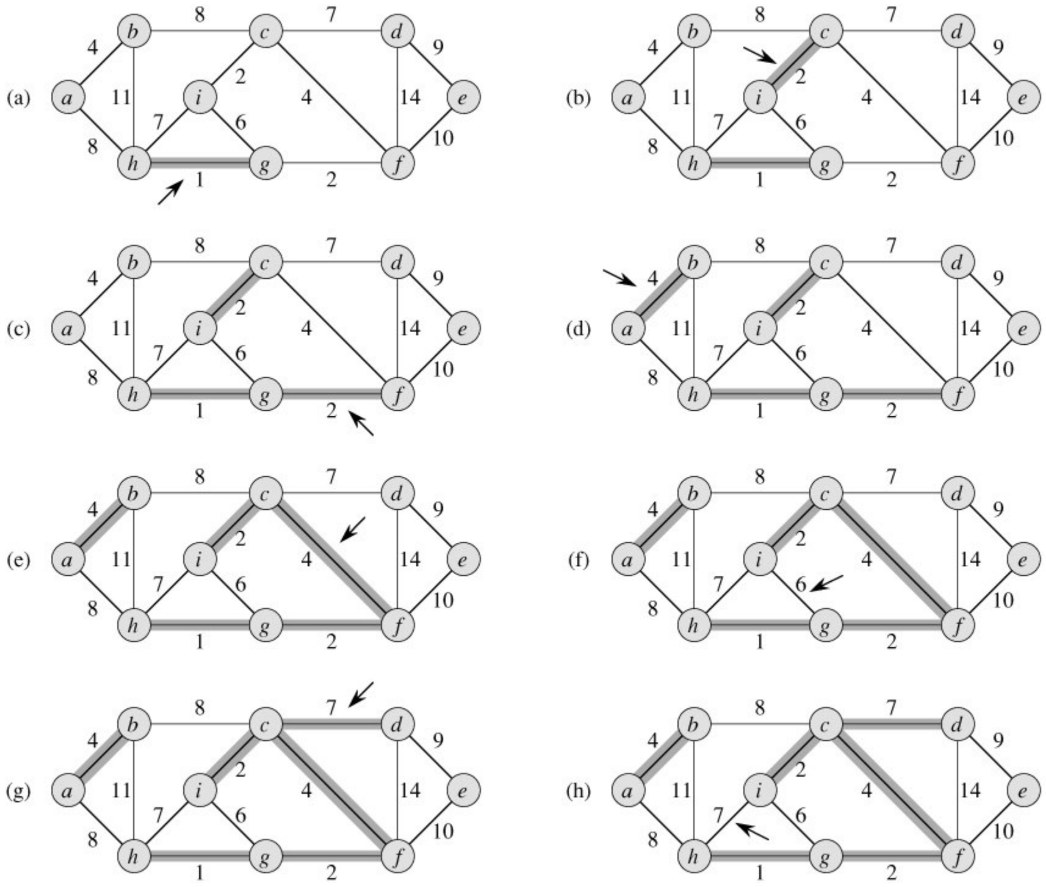
此程式由 C 語言所寫成，核心的觀念為 minimun spanning tree,做法是每次都選擇 weight 最小的 edge,直到每個 node 都被使用過

```c
#include <stdio.h>
#include<stdlib.h>


struct vertex {
    vertex* root;
    int nuber;
};

struct edge{
    int weight;
    vertex* v1;
    vertex* v2;
    bool used;
};

int comp(const void *a, const void *b){
    return (*(edge *)a).weight > (*(edge *)b).weight ? 1 : -1;
}

int main() {

    int n;
    scanf("%d", &n);

    while (n--) {
        int totalWeight = 0;
        int v, e, option;
        scanf("%d %d %d", &v, &e, &option);

        vertex *vertexs = new vertex[v];
        for (int i = 0; i < v; i++) {//初始化
            vertexs[i].nuber = i;
            vertexs[i].root = NULL;
        }

        edge *edges = new edge[e];
        for (int i = 0; i < e; i++) {//讀edge的側資
            int v1, v2;
            scanf("%d %d", &v1, &v2);
            edges[i].v1 = &vertexs[v1];
            edges[i].v2 = &vertexs[v2];
            scanf("%d", &edges[i].weight);
            edges[i].used = false;
        }
        qsort(edges, e, sizeof(edge), comp);//依照weight做排序

        for (int i = 0; i < e; i++) {
            vertex *root1 = edges[i].v1;
            vertex * root2= edges[i].v2;
            while (root1->root != NULL) {  //找該node的root

                root1 = root1->root;
            }
            while (root2->root != NULL) {  //找該node的root
                root2 = root2->root;
            }

            if (root1 != root2) {
                root2->root = root1;
                edges[i].used = true;
            }
        }

        for (int i = 0; i < v; i++) {//Kruskal's algorithm
            if (edges[i].used) {
                if (option == 1) {
                    if(edges[i].v1->nuber<= edges[i].v2->nuber)
                        printf("%d %d\n", edges[i].v1->nuber, edges[i].v2->nuber);
                    else
                        printf("%d %d\n", edges[i].v2->nuber, edges[i].v1->nuber);
                }
                totalWeight += edges[i].weight;
            }
        }
        printf("%d\n", totalWeight);
    }


    return 0;
}
```

Pseudo code:

```
Sort(edge,依照 weight)
totalWeight=0
For i=0 to vertice 數量
    If(edge[i]連接的兩個 node 其中一個的 color 是 1)
        將該 node 的 color 改為 2
        totalWeight+= edge[i].weight
```

圖解:



如圖:

每次都選擇 weight 最小的 edge

直到每個 node 都被使用過

Prim’s

程式摘要:

此程式由 C 語言所寫成，核心的觀念為 minimun spanning tree,做法是先隨機選一個 node,將該 node 鄰近的 edge 的 color 標為 1,並將該 node 的 color 標為 2;然後將 edge color 為 1 且 weight 最小的取出,不斷執行以上步驟 v-1 次

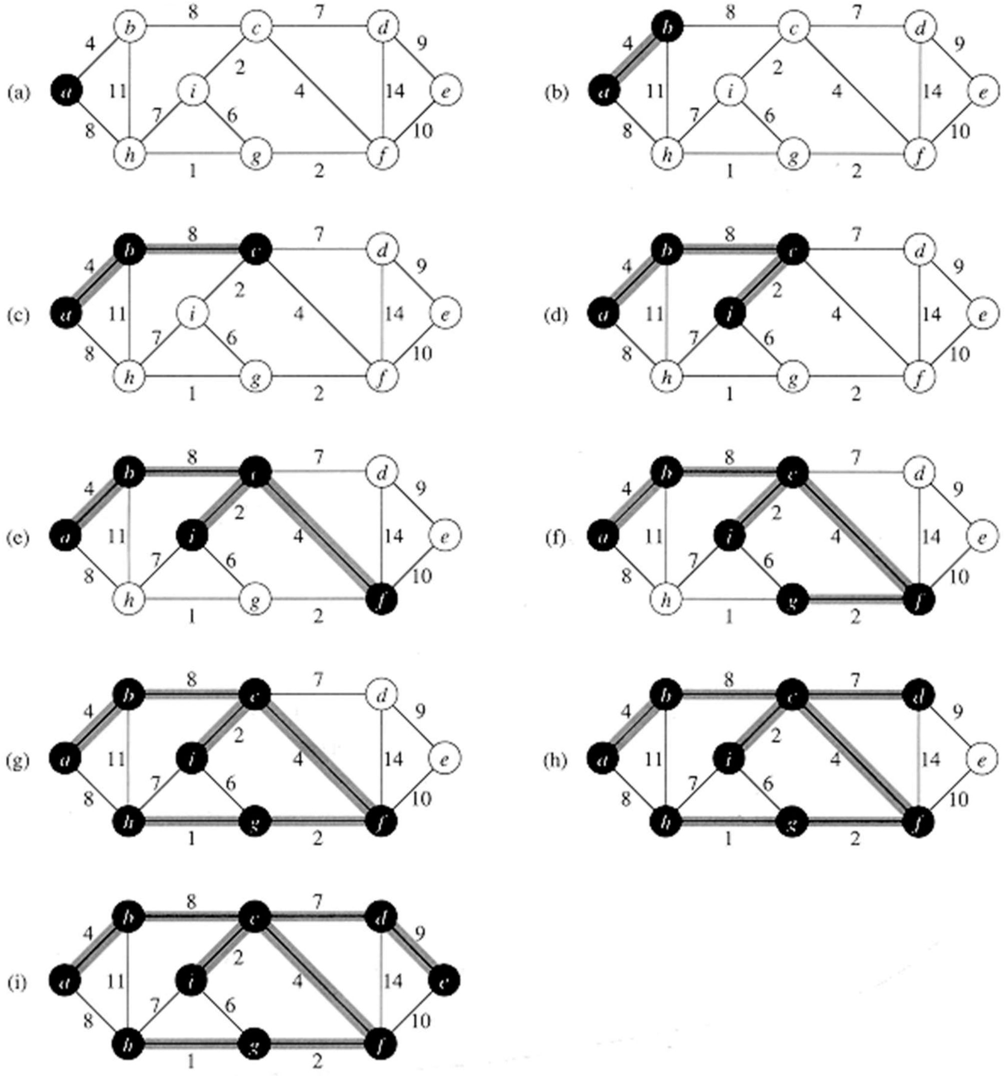
```
#include <stdio.h>
#include<stdlib.h>


struct vertex {
    vertex* parent;
    int nuber;
    int color;

};

struct edge {
    int weight;
    vertex* v1;
    vertex* v2;
    int color;
};

int main() {

    int n;
    scanf("%d", &n);

    while (n--) {
        int totalWeight = 0;
        int v, e, option;
        scanf("%d %d %d", &v, &e, &option);

        vertex *vertexs = new vertex[99];
        for (int i = 0; i < v; i++) {//初始化
            vertexs[i].nuber = i;
            vertexs[i].parent = NULL;
            vertexs[i].color = 0;
        }
        edge *edges = new edge[9999];
        for (int i = 0; i < e; i++) {//讀 edge 的側資
            int v1, v2;
            scanf("%d %d", &v1, &v2);
            if (v1 < v2) {
                edges[i].v1 = &vertexs[v1];
                edges[i].v2 = &vertexs[v2];
            }
            else {
                edges[i].v2 = &vertexs[v1];
                edges[i].v1 = &vertexs[v2];
            }

            scanf("%d", &edges[i].weight);
            edges[i].color = 0;
        }

        vertex *selectVertice = &vertexs[0];
        v--;
        while (v--) {

            selectVertice->color = 2;

            for (int i = 0; i < e; i++) {
                if (edges[i].v1 == selectVertice) {
                    if (edges[i].v2->color == 0)
                        edges[i].v2->color = 1;
                    if (edges[i].color == 0)
                        edges[i].color = 1;
                }
                else if (edges[i].v2 == selectVertice) {
                    if (edges[i].v1->color == 0)
                        edges[i].v1->color = 1;
                    if (edges[i].color == 0)
                        edges[i].color = 1;
                }
                if (edges[i].v1->color == 2 && edges[i].v2->color == 2) {
                    edges[i].color = 2;
                }
            }

            edge*selectEdge = NULL;

            for (int i = 0; i < e; i++) {
                if (selectEdge == NULL)
                    if (edges[i].color == 1)
                        selectEdge = &edges[i];
                    else
                        continue;
                else if (edges[i].color == 1 && edges[i].weight < selectEdge->weight) {
                    selectEdge = &edges[i];
                }
            }

            if (selectEdge != NULL) {
                selectEdge->color = 2;
                if (selectEdge->v1->color == 1)
                    selectVertice = selectEdge->v1;
                else
                    selectVertice = selectEdge->v2;

                totalWeight += selectEdge->weight;
                if (option == 1)
                    printf("%d %d\n", selectEdge->v1->nuber, selectEdge->v2->nuber);
            }
            else
                break;
        }
```

```

        printf("%d\n", totalWeight);
    }

    //system("pause");
    return 0;
}
```

Pseudo code:

```
Select=number 為 0 的 node
While(node 數量—)
    將 select 鄰近的 edge 的 color 標為 1
    選出 edge color 為 1 且 weight 最小的 edge
```

圖解:



如圖

將 select 鄰近的 edge 的 color 標為 1

選出 edge color 為 1 且 weight 最小的 edge