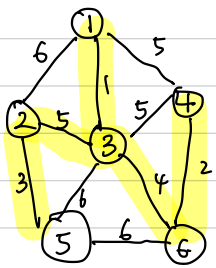
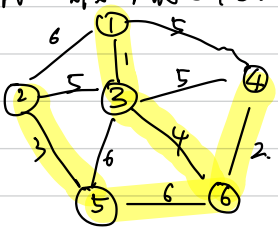
(1) 最短路径树是一个树状子图，是包含某个源节点到其他节点的最短路径

MST是找到包含所有节点的让边的权值和最小(而且可以不唯一)

例如



最短最小支撑树 那如构建6节点为起点SPT



显然不一样

2.

Exercise 2:

1. **图 1** 是一幅带权无向图，请写出用 **Dijkstra 算法**得到 **S点** 到其他各点最短路径的**过程**。
2. 请画出用 **Dijkstra 算法**得到 **H点** 到其他各点的**最短路径树**。

(1) 给结点编号

过程见下页

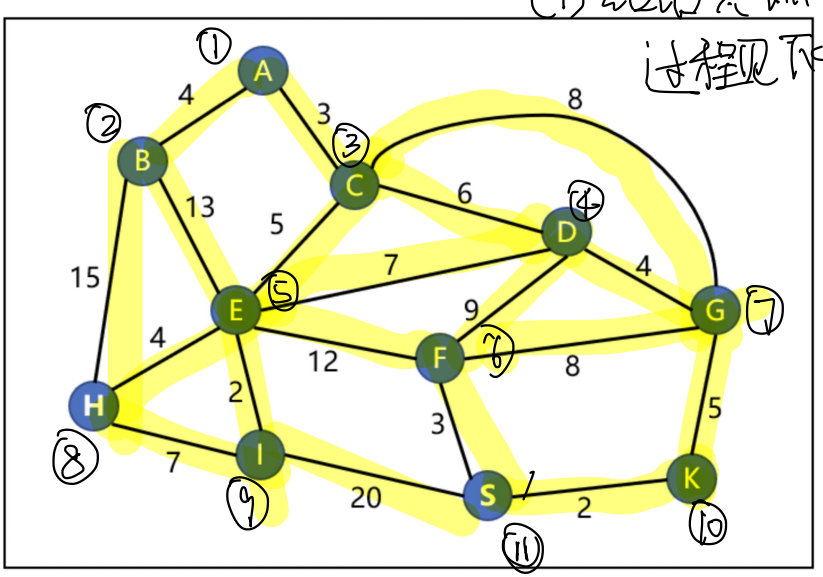


图 1

Exercise 2:

(2) 如下黄色路径是最短路径树

1. **图 1** 是一幅带权无向图，请写出用 **Dijkstra 算法**得到 **S点** 到其他各点最短路径的**过程**。
2. 请画出用 **Dijkstra 算法**得到 **H点** 到其他各点的**最短路径树**。

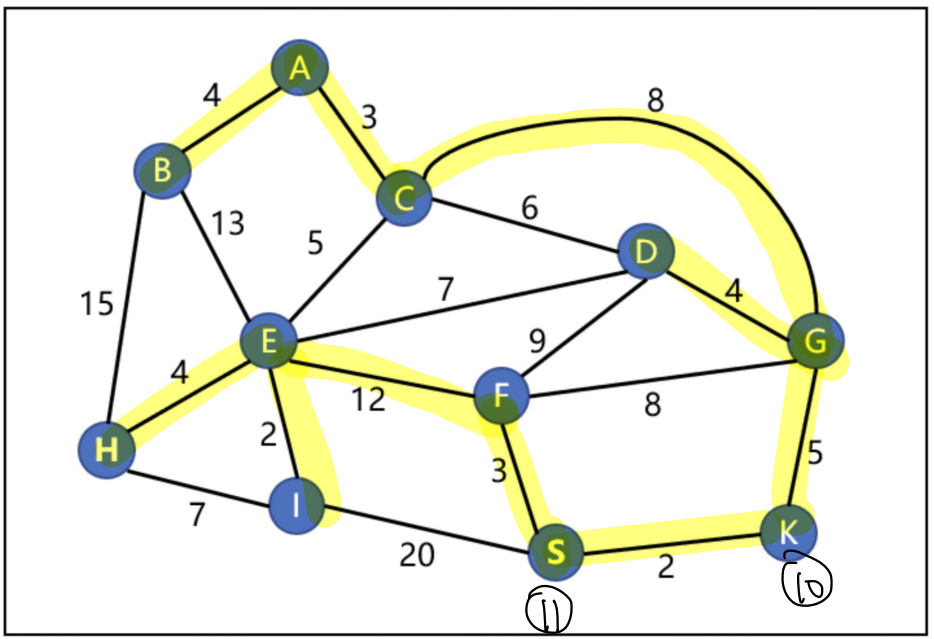


图 1

(3) 下面的有向图

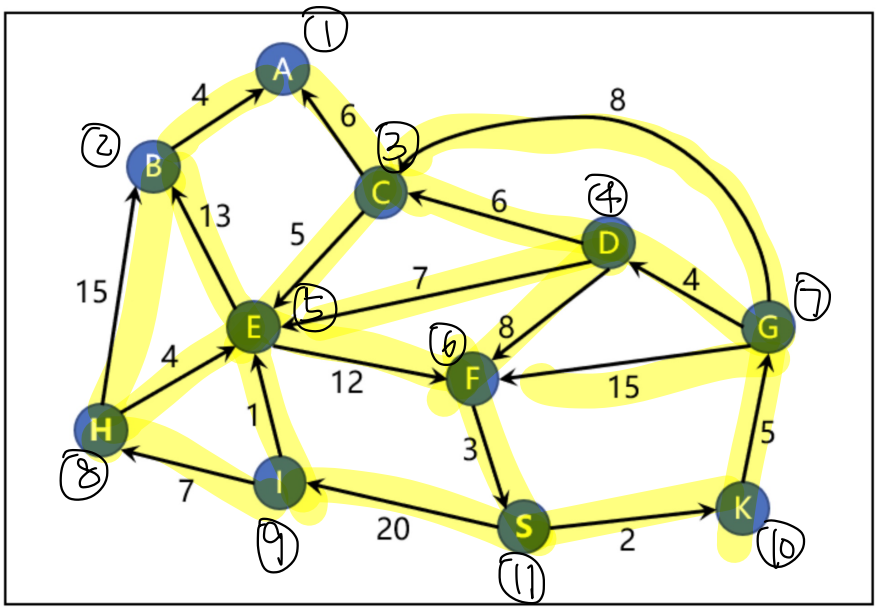


图 2

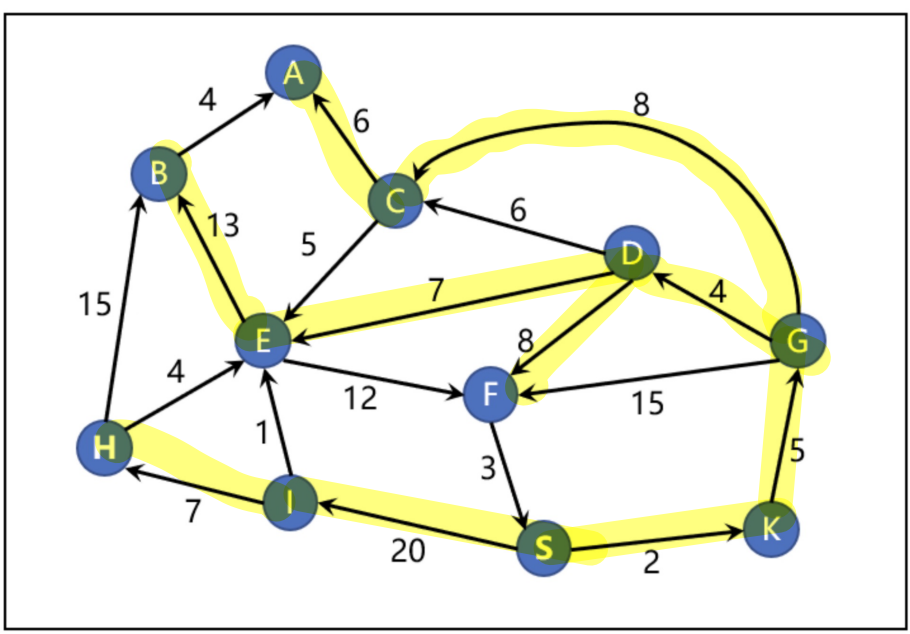
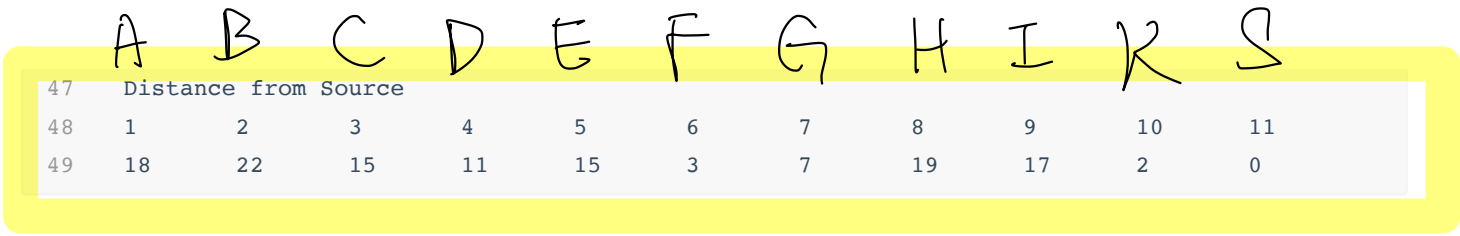


图 2

第2题

```
ziqianzhang@ziqiandeMac-Studio 作业12 Dijkstra算法 % ./a.out < 2.data
Step 1:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     INF     INF     INF     3       INF     INF     20      2       0
Step 2:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     INF     INF     INF     3       7       INF     20      2       0
Step 3:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     INF     12      15      3       7       INF     20      2       0
Step 4:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     15      11      15      3       7       INF     20      2       0
Step 5:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     15      11      15      3       7       INF     20      2       0
Step 6:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     28      15      11      15      3       7       19      17      2       0
Step 7:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
18      28      15      11      15      3       7       19      17      2       0
Step 8:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
18      28      15      11      15      3       7       19      17      2       0
Step 9:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
18      22      15      11      15      3       7       19      17      2       0
Step 10:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
18      22      15      11      15      3       7       19      17      2       0
Step 11:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
18      22      15      11      15      3       7       19      17      2       0
Final Result:
```




```
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
18      22      15      11      15      3       7       19      17      2       0
```

第三题

```
ziqianzhang@ziqiandeMac-Studio 作业12 Dijkstra算法 % ./3 < 3.data
Dijkstra Algorithm
Step 1:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     INF     INF     INF     INF     INF     INF     20      2       0
Step 2:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     INF     INF     INF     INF     7       INF     20      2       0
Step 3:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     15      11      INF     22      7       INF     20      2       0
Step 4:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
INF     INF     15      11      18      19      7       INF     20      2       0
Step 5:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      INF     15      11      18      19      7       INF     20      2       0
Step 6:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       INF     20      2       0
Step 7:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       INF     20      2       0
Step 8:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       27      20      2       0
Step 9:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       27      20      2       0
Step 10:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       27      20      2       0
Step 11:
```

```
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       27      20      2       0
Final Result:
Distance from Source
1       2       3       4       5       6       7       8       9       10      11
21      31      15      11      18      19      7       27      20      2       0
```

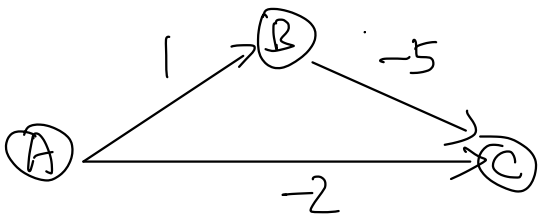
程序文件

```cpp
#include <iostream>
#include <limits.h>
using namespace std;

// 定义一个表示图的邻接矩阵的结构体
struct Graph {
    int V; // 图中节点数量
    int **adj; // 邻接矩阵

    // 添加边（无向图）
    void addNoDirectionEdge(int u, int v, int w) {
        adj[u][v] = w;
        adj[v][u] = w;
    }
    // 添加边（有向图）
    void addDirectionEdge(int u, int v, int w) {
        adj[u][v] = w;
    }

};

// 创建图
Graph* createGraph(int V) {
    Graph* graph = new Graph;
    graph->V = V;
    graph->adj = new int*[V];
    for (int i = 0; i < V; ++i) {
        graph->adj[i] = new int[V];
        for (int j = 0; j < V; ++j)
            graph->adj[i][j] = 0;
    }
    return graph;
}



// 找到dist数组中的最小值
int minDistance(int dist[], bool sptSet[], int V) {
    int min = INT_MAX, min_index;
```

```
    for (int v = 0; v < V; ++v)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    return min_index;
}

// 打印结果
void printSolution(int dist[], int V) {
    cout << "Distance from Source\n";
    for (int i = 1; i < V; ++i){
        // char ch = 'A' + i - 1;
        cout << i << "\t";
    }


    cout << endl;
    for (int i = 1; i < V; ++i){
        if (dist[i] == INT_MAX)
            cout << "INF" << "\t";
        else
            cout << dist[i] << "\t";
    }
    cout << endl;
}

// Dijkstra算法
void dijkstra(Graph* graph, int src) {
    cout << "Dijkstra Algorithm\n";
    int V = graph->V;
    int dist[V];
    bool sptSet[V];
    for (int i = 0; i < V; ++i)
        dist[i] = INT_MAX, sptSet[i] = false;
    dist[src] = 0;
    for (int count = 0; count < V - 1; ++count) {
        int u = minDistance(dist, sptSet, V);
        sptSet[u] = true;
        for (int v = 0; v < V; ++v) {
            // sptSet是Dijkstra算法中的一个bool类型的数组，用于表示每个节点是否已经被加入最短路径树（SPT）中。
            // 只有当sptSet[v]为false时，并且存在一条从u到v的边，且u节点的最短路径树中的距离加上u到v的边的权值小于dist[v]时，才更新dist[v]。
            if (!sptSet[v] && graph->adj[u][v] && dist[u] != INT_MAX && dist[u] + graph->adj[u][v] < dist[v])
                dist[v] = dist[u] + graph->adj[u][v];
        }
        // 打印中间结果
        cout << "Step " << count+1 << ":" << endl;
        printSolution(dist, V);
    }
    // 打印最终结果
    cout << "Final Result:" << endl;
```

```
89        printSolution(dist, V);
90    }
```

(4) 无法用于负权的边 例如



A	B	C
0	1	-2

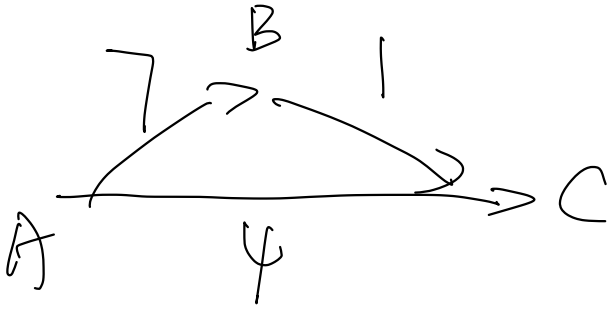
{A. C}

A	B	C
0	1	-2.

{A. B C}

所以会有. 那能不能加上一个正数值解决?

不能. 得到结果



1	2	3
0	7	4

但是能不能 -6 ? 没用

0	1	-2

所以 Dijkstra 算法不适合负. 因为会忽略到已经通过的一些点(比如上面C的) C被标记后. B再检测其他路的时候. 会忽略C. 所以不行.