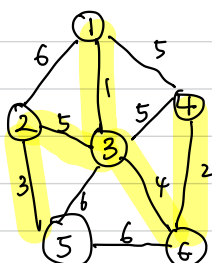
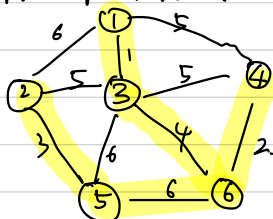


(1) 最短路径树是一个树状子图 是包含某个源节点到其他节点的最短路径
MST是找到包含所有节点的 让边的权值和最小(而且可以不唯一)

例如



黄线是支撑树 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

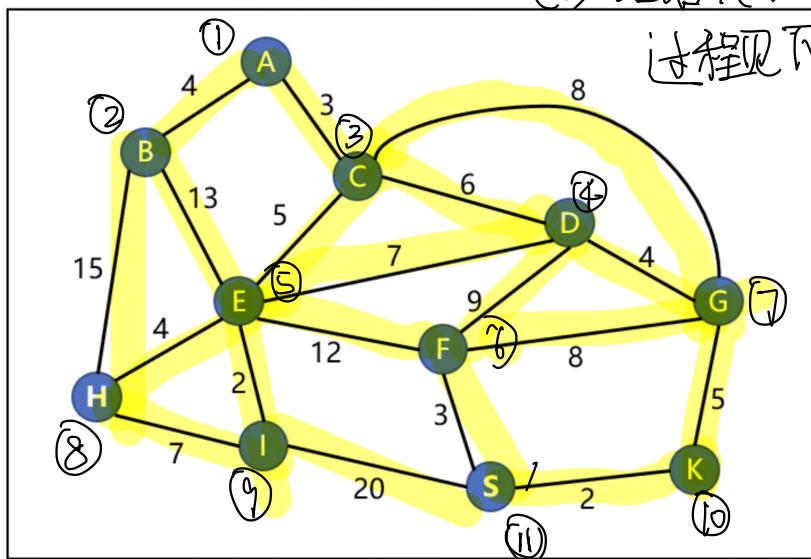


图 1

Exercise 2:

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。

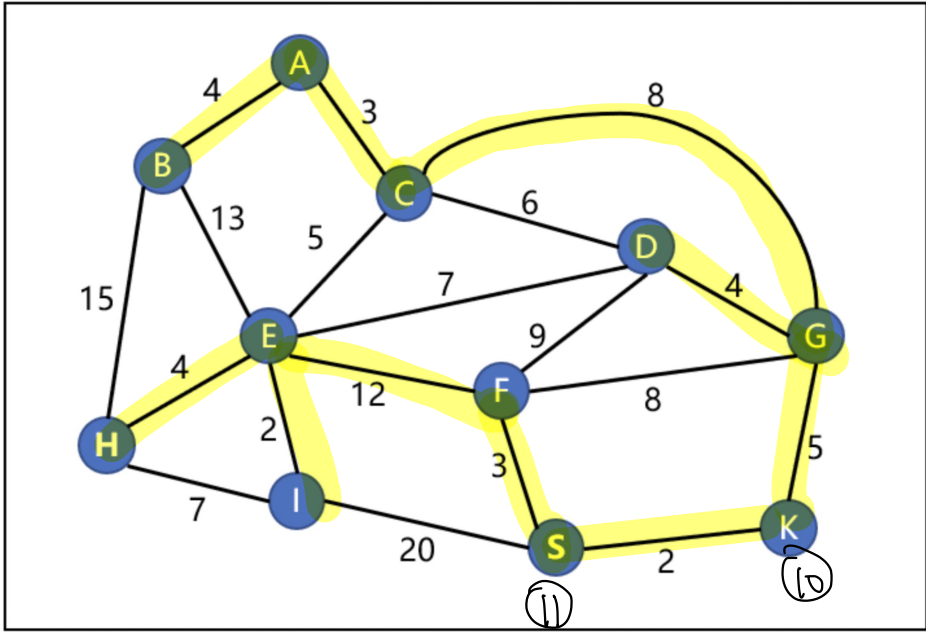


图 1

(2) 如下黄色路径
是最短路径树

(3) 下面的编号图

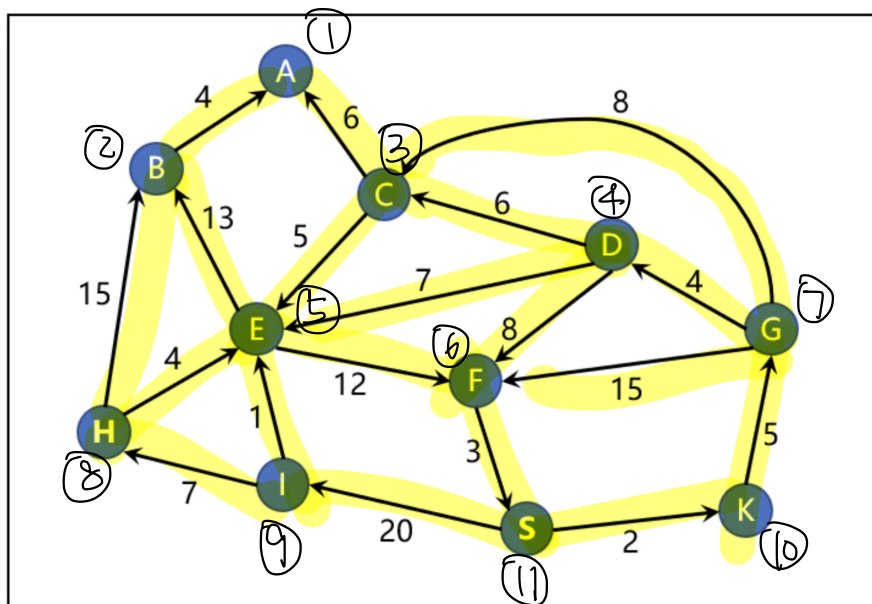


图 2

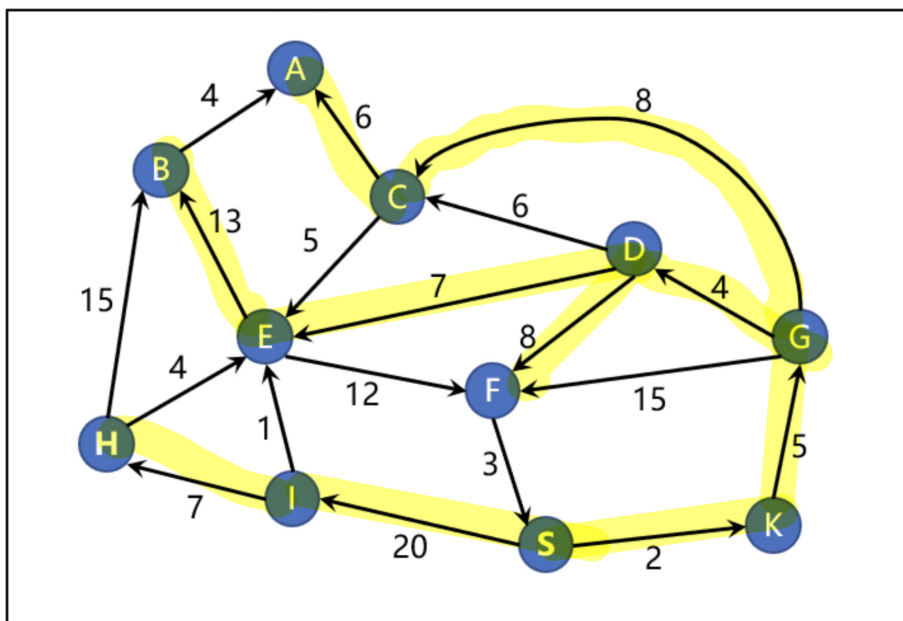


图 2

第2题

```
1  ziqianzhang@ziqiandeMac-Studio 作业12 Dijkstra算法 % ./a.out < 2.data
2  Step 1:
3  Distance from Source
4  1      2      3      4      5      6      7      8      9      10     11
5  INF    INF    INF    INF    INF    3      INF    INF    20    2      0
6  Step 2:
7  Distance from Source
8  1      2      3      4      5      6      7      8      9      10     11
9  INF    INF    INF    INF    INF    3      7      INF    20    2      0
10 Step 3:
11 Distance from Source
12 1      2      3      4      5      6      7      8      9      10     11
13 INF    INF    INF    12    15    3      7      INF    20    2      0
14 Step 4:
15 Distance from Source
16 1      2      3      4      5      6      7      8      9      10     11
17 INF    INF    15    11    15    3      7      INF    20    2      0
18 Step 5:
19 Distance from Source
20 1      2      3      4      5      6      7      8      9      10     11
21 INF    INF    15    11    15    3      7      INF    20    2      0
22 Step 6:
23 Distance from Source
24 1      2      3      4      5      6      7      8      9      10     11
25 INF    28    15    11    15    3      7      19    17    2      0
26 Step 7:
27 Distance from Source
28 1      2      3      4      5      6      7      8      9      10     11
29 18     28    15    11    15    3      7      19    17    2      0
30 Step 8:
31 Distance from Source
32 1      2      3      4      5      6      7      8      9      10     11
33 18     28    15    11    15    3      7      19    17    2      0
34 Step 9:
35 Distance from Source
36 1      2      3      4      5      6      7      8      9      10     11
37 18     22    15    11    15    3      7      19    17    2      0
38 Step 10:
39 Distance from Source
40 1      2      3      4      5      6      7      8      9      10     11
41 18     22    15    11    15    3      7      19    17    2      0
42 Step 11:
43 Distance from Source
44 1      2      3      4      5      6      7      8      9      10     11
45 18     22    15    11    15    3      7      19    17    2      0
46 Final Result:
```

A B C D E F G H I K S

47	Distance from Source										
48	1	2	3	4	5	6	7	8	9	10	11
49	18	22	15	11	15	3	7	19	17	2	0

第三题

```

1  ziqianzhang@ziquiandeMac-Studio 作业12 Dijkstra算法 % ./3 < 3.data
2  Dijkstra Algorithm
3  Step 1:
4  Distance from Source
5  1      2      3      4      5      6      7      8      9      10     11
6  INF    INF    INF    INF    INF    INF    INF    INF    20    2      0
7  Step 2:
8  Distance from Source
9  1      2      3      4      5      6      7      8      9      10     11
10 INF    INF    INF    INF    INF    INF    7      INF    20    2      0
11 Step 3:
12 Distance from Source
13 1      2      3      4      5      6      7      8      9      10     11
14 INF    INF    15    11    INF    22    7      INF    20    2      0
15 Step 4:
16 Distance from Source
17 1      2      3      4      5      6      7      8      9      10     11
18 INF    INF    15    11    18    19    7      INF    20    2      0
19 Step 5:
20 Distance from Source
21 1      2      3      4      5      6      7      8      9      10     11
22 21    INF    15    11    18    19    7      INF    20    2      0
23 Step 6:
24 Distance from Source
25 1      2      3      4      5      6      7      8      9      10     11
26 21    31    15    11    18    19    7      INF    20    2      0
27 Step 7:
28 Distance from Source
29 1      2      3      4      5      6      7      8      9      10     11
30 21    31    15    11    18    19    7      INF    20    2      0
31 Step 8:
32 Distance from Source
33 1      2      3      4      5      6      7      8      9      10     11
34 21    31    15    11    18    19    7      27    20    2      0
35 Step 9:
36 Distance from Source
37 1      2      3      4      5      6      7      8      9      10     11
38 21    31    15    11    18    19    7      27    20    2      0
39 Step 10:
40 Distance from Source
41 1      2      3      4      5      6      7      8      9      10     11
42 21    31    15    11    18    19    7      27    20    2      0
43 Step 11:

```

A B C D E F G H I K S

44	Distance from Source										
45	1	2	3	4	5	6	7	8	9	10	11
46	21	31	15	11	18	19	7	27	20	2	0
47	Final Result:										
48	Distance from Source										
49	1	2	3	4	5	6	7	8	9	10	11
50	21	31	15	11	18	19	7	27	20	2	0

程序文件

```

1  #include <iostream>
2  #include <limits.h>
3  using namespace std;
4
5  // 定义一个表示图的邻接矩阵的结构体
6  struct Graph {
7      int V; // 图中节点数量
8      int **adj; // 邻接矩阵
9
10     // 添加边（无向图）
11     void addNoDirectionEdge(int u, int v, int w) {
12         adj[u][v] = w;
13         adj[v][u] = w;
14     }
15     // 添加边（有向图）
16     void addDirectionEdge(int u, int v, int w) {
17         adj[u][v] = w;
18     }
19
20 };
21
22 // 创建图
23 Graph* createGraph(int V) {
24     Graph* graph = new Graph;
25     graph->V = V;
26     graph->adj = new int*[V];
27     for (int i = 0; i < V; ++i) {
28         graph->adj[i] = new int[V];
29         for (int j = 0; j < V; ++j)
30             graph->adj[i][j] = 0;
31     }
32     return graph;
33 }
34
35
36
37 // 找到dist数组中的最小值
38 int minDistance(int dist[], bool sptSet[], int V) {
39     int min = INT_MAX, min_index;

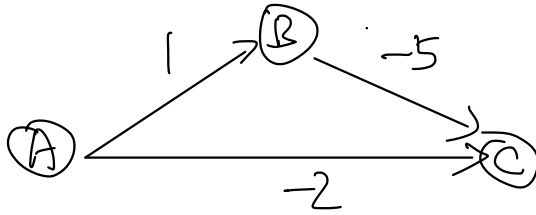
```

```

40     for (int v = 0; v < V; ++v)
41         if (sptSet[v] == false && dist[v] <= min)
42             min = dist[v], min_index = v;
43     return min_index;
44 }
45
46 // 打印结果
47 void printSolution(int dist[], int V) {
48     cout << "Distance from Source\n";
49     for (int i = 1; i < V; ++i){
50         // char ch = 'A' + i - 1;
51         cout << i << "\t";
52     }
53
54
55     cout << endl;
56     for (int i = 1; i < V; ++i){
57         if (dist[i] == INT_MAX)
58             cout << "INF" << "\t";
59         else
60             cout << dist[i] << "\t";
61     }
62     cout << endl;
63 }
64
65 // Dijkstra算法
66 void dijkstra(Graph* graph, int src) {
67     cout << "Dijkstra Algorithm\n";
68     int V = graph->V;
69     int dist[V];
70     bool sptSet[V];
71     for (int i = 0; i < V; ++i)
72         dist[i] = INT_MAX, sptSet[i] = false;
73     dist[src] = 0;
74     for (int count = 0; count < V - 1; ++count) {
75         int u = minDistance(dist, sptSet, V);
76         sptSet[u] = true;
77         for (int v = 0; v < V; ++v) {
78             // 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
79             // 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
80             // 的边的权值小于dist[v]时，才更新dist[v]。
81             if (!sptSet[v] && graph->adj[u][v] && dist[u] != INT_MAX && dist[u] + graph-
82             >adj[u][v] < dist[v])
83                 dist[v] = dist[u] + graph->adj[u][v];
84         }
85         // 打印中间结果
86         cout << "Step " << count+1 << ":" << endl;
87         printSolution(dist, V);
88     }
89     // 打印最终结果
90     cout << "Final Result:" << endl;

```

(4) 无法用于负权的边 例如



A	B	C
0	1	-2

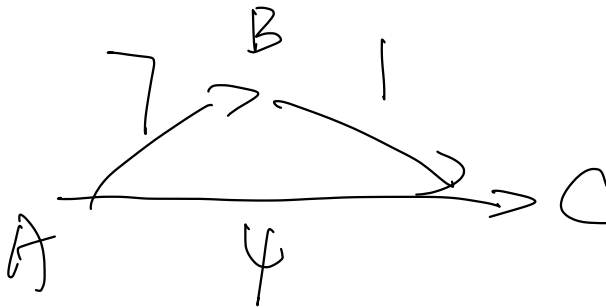
{A, C}

A	B	C
0	1	-2

{A, B, C}

所以会错, 那能不能加上一个正数解决?

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.