《数据结构与算法》第十六次作业练习题

开始时间	12/29/2023 4:30:00 PM	结束时间	01/02/2024 11:59:00 PM	答题时长	6209分钟
答卷类型	标准答案	总分	65		

判断题

得分：暂无　总分：11

1-1 在一个有权无向图中，若 b 到 a 的最短路径距离是12，且 c 到 b 之间存在一条权为2的边，则 c 到 a 的最短路径距离一定不小于10。 (3分)

(●) T　( ) F

1-2 P 是顶点 S 到 T 的最短路径，如果该图中的所有路径的权值都加 1，P 仍然是 S 到 T 的最短路径。(2分)

( ) T　(●) F

1-3 对于带权无向图 G = (V, E)，M 是 G 的最小生成树，则 M 中任意两点 V1 到 V2 的路径一定是它们之间的最短路径。(2分)

( ) T　(●) F

1-4 若连通图上各边的权值均不相同，则该图的最小生成树是唯一的。(2分)

(●) T　( ) F

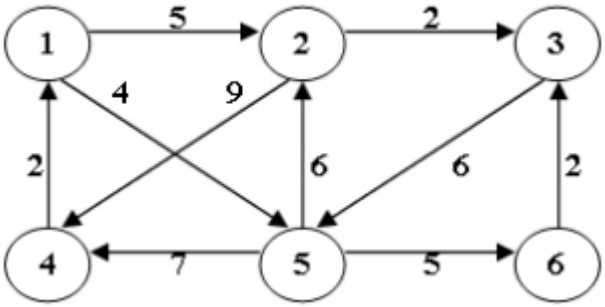
1-5 关于带权无向图的最小生成树问题，若图中某回路上的边权值各不相同，则其中权值最小的边一定在某最小生成树中。(2分)

( ) T　(●) F

单选题

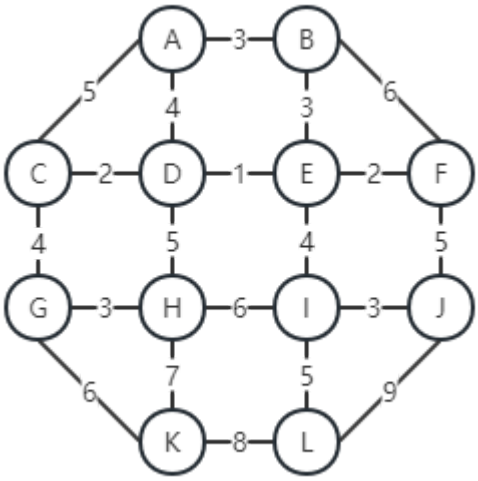
得分：暂无　总分：14

2-1 使用迪杰斯特拉（Dijkstra）算法求下图中从顶点1到其他各顶点的最短路径，当顶点2被纳入确定顶点时，顶点1到顶点4的最短路径长度（dist[4]）为（ ）。 (2分)



(●) A. 11　( ) B. 14　( ) C. 20　( ) D. 正无穷

2-2 (2分)



Apply Dijkstra's Algorithm from vertex S = {A}. Which of the following statement is false?

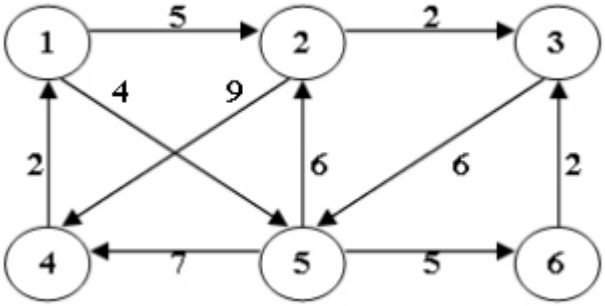
A. The 4^{th} vertex added to the S is E

B. The shortest path from A to L is 15

C. There is exactly one shortest path from A to F

D. The cost of the shortest path from A -> G, A->H, and A->I is the same

2-3 使用迪杰斯特拉（Dijkstra）算法求下图中从顶点2到其他各顶点的最短路径，则顶点2到顶点6的最短路径长度为（ ）。 (2分)



A. 13

B. 4

C. 11

D. 正无穷

2-4 已知无向连通图 G 中各边的权值均为 1。下列算法中，一定能够求出图 G 中从某顶点到其余各顶点最短路径的是： (2分)

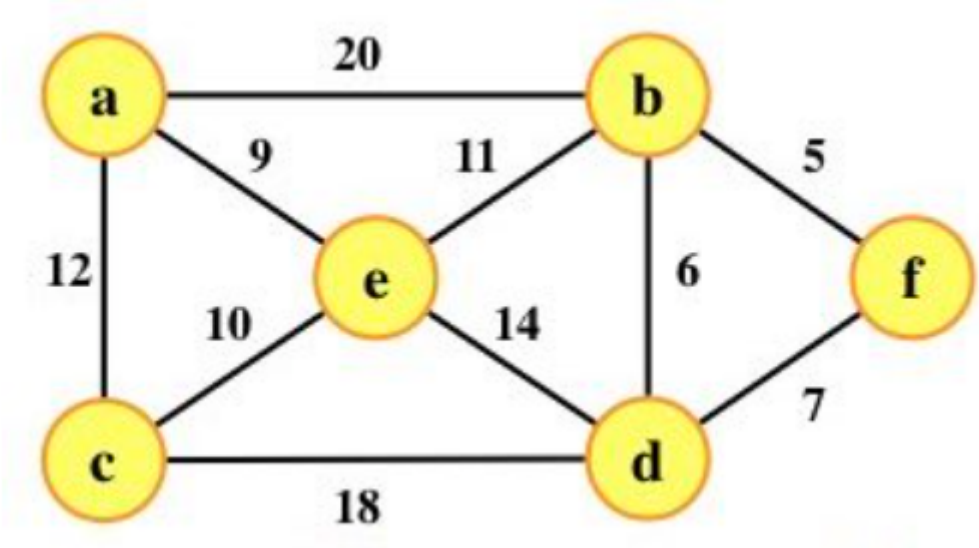
I、 普里姆（Prim）算法

II、 克鲁斯卡尔（Kruskal）算法

III、 图的广度优先搜索算法

A. 仅 I　　B. 仅 III　　C. 仅 I、II　　D. I、II、III

2-5 图G如下图所示，使用克鲁斯卡尔（Kruskal）算法求图G的最小生成树，加到最小生成树中的边依次是（ ）。 (2分)



A. (b, f), (b, d), (a, e), (c, e), (b, e)

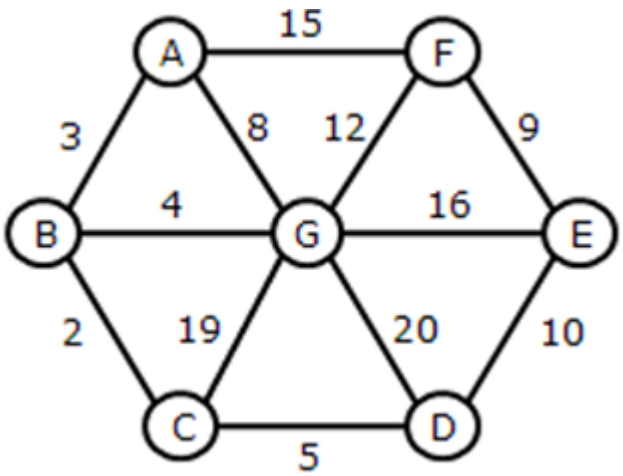
B. (b, f), (b, d), (b, e), (c, e), (a, e)

C. (a, e), (b, e), (c, e), (b, d), (b, f)

D. (a, e), (c, e), (b, e), (b, f), (b, d)

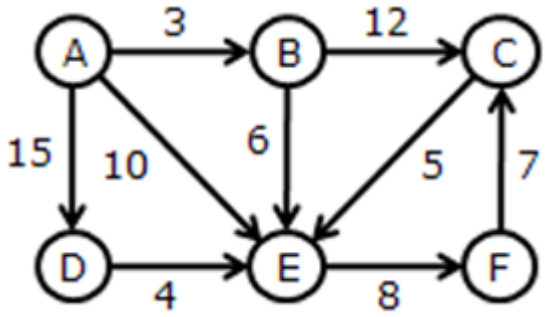
2-6 (2分)

用Prim算法，以G为初始生长点，求下图的最小生成树时，依次得到的树边为：



- A. GB4、BC2、AB3、CD5、ED10、EF9
- B. BC2、AB3、GB4、CD5、EF9、ED10
- C. GB4、BC2、CD5、ED10、EF9、AB3
- D. AB3、BC2、GB4、CD5、ED10、EF9

2-7 用Dijkstra算法求下图顶点A到其余各顶点的最短路径时，将按照什么的次序，依次求出A到它们的最短路径。 (2分)



- A. BEDFC
- B. BCEDF
- C. EDFCB
- D. BEDCF

编程题

得分：暂无　总分：40

7-1 信使（15分）

Background

Special for beginners, ^_^

Description

战争时期，前线有 n 个哨所，每个哨所可能会与其他若干个哨所之间有通信联系。信使负责在哨所之间传递信息，当然，这是要花费一定时间的（以天为单位）。指挥部设在第一个哨所。当指挥部下达一个命令后，指挥部就派出若干个信使向与指挥部相连的哨所送信。当一个哨所接到信后，这个哨所内的信使们也以同样的方式向其他哨所送信。直至所有 n 个哨所全部接到命令后，送信才算成功。因为准备充足，每个哨所内都安排了足够的信使（如果一个哨所与其他k个哨所有通信联系的话，这个哨所内至少会配备k个信使）。

现在总指挥请你编一个程序，计算出完成整个送信过程最短需要多少时间。

Format

Input

输入第 1 行有两个整数 n 和 m，中间用 1 个空格隔开，分别表示有 n 个哨所和 m 条通信线路。$1 \leq n \leq 100$。

第 2 至 $m+1$ 行：每行三个整数 i, j, k，中间用 1 个空格隔开，表示第 i 个和第 j 个哨所之间存在通信线路，且这条线路要花费 k 天。

Output

输出仅一个整数，表示完成整个送信过程的最短时间。如果不是所有的哨所都能收到信，就输出 -1。

Samples

样例输入1

```
4 4
1 2 4
2 3 7
2 4 1
3 4 6
```

样例输出1

```
11
```

Limitation

1s, 1024KiB for each test case.

7-2 校园最短路径（10分）

下图是校园地图，请创建地图模型，并基于该模型求出一教到其余各地的最短路径和长度。

输入样例:

```
14 14
ABCDEFGHIJKLMN
AB 1
AC 1
AD 3
BG 2
CE 1
DM 3
EF 1
FI 2
GJ 5
IL 3
JL 2
KL 2
KN 6
MN 5
```

输出样例:

```
dist[A][A]=0
A
dist[A][B]=1
A->B
dist[A][C]=1
A->C
dist[A][D]=3
A->D
dist[A][E]=2
A->C->E
dist[A][F]=3
A->C->E->F
dist[A][G]=3
A->B->G
dist[A][H]=256
H
dist[A][I]=5
A->C->E->F->I
dist[A][J]=8
A->B->G->J
dist[A][K]=10
A->C->E->F->I->L->K
dist[A][L]=8
A->C->E->F->I->L
dist[A][M]=6
A->D->M
dist[A][N]=11
A->D->M->N
```

参考答案（题目作者）：

编译器 GXX

代码

```
#include <stdio.h>
#define MAX_VERTEX_NUM 14                  //最大顶点数
#define INFINITY 256   //正无穷
#define ERROR -1
#define OK 1
#define TRUE 1
#define FALSE 0
typedef char VertexType;
typedef int Vertex;
typedef struct{
  char vexs[MAX_VERTEX_NUM];            //存放顶点的一维数组
  int arcs[MAX_VERTEX_NUM][MAX_VERTEX_NUM];     //邻接矩阵
  int Nv,Na;          //图的当前顶点数和边数
}MGraph;
int path[MAX_VERTEX_NUM];
int locate(MGraph G,char v)
{
  int i;
  for(i=0;i<G.Nv;i++)
    if(G.vexs[i]==v)
       return i;
   return -1;
```

```
}
void CreatMGraph(MGraph &G)
{
    int i,j,k;
    char v1,v2;
    int w;
    scanf("%d%d",&G.Nv,&G.Na);
    getchar();
    for(i=0;i<G.Nv;i++)
         scanf("%c",&G.vexs[i]);
    for(i=0;i<G.Nv;i++)
         for(j=0;j<G.Nv;j++)
            G.arcs[i][j]=INFINITY;
    for(k=0;k<G.Na;k++) {
         getchar();
        scanf("%c%c %d",&v1,&v2,&w);
        i=locate(G,v1);
        j=locate(G,v2);
        G.arcs[i][j]=w;
        G.arcs[j][i]=w;
    }
}
void printGraph(MGraph G)//打印图
{
    int i,j;
    for(i=0;i<G.Nv;i++)
    {
       for(j=0;j<G.Nv;j++)
          printf("%4d",G.arcs[i][j]);
       printf("\n");
    }
}
Vertex FindMinDist( MGraph Graph, int dist[], int collected[] )
    { /* 返回未被收录顶点中dist最小者 */
        Vertex MinV, V;
        int MinDist = INFINITY;

        for (V=0; V<Graph.Nv; V++) {
            if ( collected[V]==FALSE && dist[V]<MinDist) {
                /* 若V未被收录，且dist[V]更小 */
                MinDist = dist[V]; /* 更新最小距离 */
                MinV = V; /* 更新对应顶点 */
            }
        }
        if (MinDist < INFINITY) /* 若找到最小dist */
            return MinV; /* 返回对应的顶点下标 */
        else return ERROR;  /* 若这样的顶点不存在，返回错误标记 */
    }

    bool Dijkstra( MGraph Graph, int dist[],  Vertex S )
    {
        int collected[MAX_VERTEX_NUM];
        Vertex V, W;

        /* 初始化：此处默认邻接矩阵中不存在的边用INFINITY表示 */
        for ( V=0; V<Graph.Nv; V++ ) {
            dist[V] = Graph.arcs[S][V];
            if ( dist[V]<INFINITY )
```

```
                path[V] = S;
            else
                path[V] = -1;
            collected[V] = FALSE;
        }
        /* 先将起点收入集合 */
        dist[S] = 0;
        collected[S] = TRUE;
        while(1)
        {
         V=FindMinDist(Graph,dist,collected);
         if(V==ERROR)
            break;
         collected[V]=TRUE;
         for(W=0;W<Graph.Nv;W++)
            if(Graph.arcs[V][W]<INFINITY && collected[W]==FALSE)
            if(dist[V]+Graph.arcs[V][W]<dist[W])
            {dist[W]=dist[V]+Graph.arcs[V][W];
            path[W] = V;}
        }
        return OK; /* 算法执行完毕，返回正确标记 */
    }
void DisplayPath(MGraph G , int begin ,int end )
{
    if(path[end] != -1){
        DisplayPath(G , begin ,path[end]);
        printf("%c->",G.vexs[path[end]]);
       }
}
int main()
    {
        MGraph G;
        int i,j;
        int dist[MAX_VERTEX_NUM];
        CreatMGraph(G);
        i=0;
        Dijkstra(G,dist,i);
        for(j=0;j<G.Nv;j++)
            {printf("dist[%c][%c]=%d\n",G.vexs[i],G.vexs[j],dist[j]);
             DisplayPath(G , i, j);
             printf("%c",G.vexs[j]);
             printf("\n");
             }
        return 0;
    }
```

7-3 最小生成树-kruskal算法（15分）

某地对偏远地区实行“村村通”工程，目标是使整个地区任何两个村落间都可以实现快速交通（但不一定有直接的快速道路相连，只要互相间接通过快速路可达即可）。现得到拟修建道路的费用，现请你编写程序，计算出全地区畅通需要的最低成本。

输入格式:

输入的第一行给出村庄数目N (1≤N≤20)和拟修建的道路数M

接下来的M行对应修建每条村庄间道路的成本，每行给出3个正整数，分别是两个村庄的编号（从1编号到N），此两村庄间道路的成本。

输出格式:

输出需修建的道路，按kruskal算法得到的顺序，输出每条路，每行输出一条道路，形式如：道路1编号,道路2编号,费用。(编号小的放前面，编号大的放后面，逗号为英文状态下的逗号)

输入样例:

```
4 6
1 2 1
1 3 4
1 4 1
2 3 3
2 4 2
3 4 5
```

输出样例:

```
1,2,1
1,4,1
2,3,3
```