数据结构课程设计报告

求无向连通网的最小生成树

学号：_____

姓名：赵宏宇

专业：_____

日期：2017 年 06 月 12 日

评分
满分——**15** 分

报告目录

1. 课程设计内容与要求

用字符文件提供数据建立连通带权无向网络邻接矩阵存储结构。编写程序，求一棵最小生成树。要求输出最小生成树的各条边（用顶点无序偶表示）、各条边上的权值、最小生成树所有边上的权值之和。

学生可选提高要求：输出所有的最小生成树。

2. 程序设计报告

程序采用 C++语言（未使用面向对象）按提高要求进行设计编码：输出带权无向连通网所有不同构的最小生成树。

2.1 总体设计

程序采用 Prim 算法生成最小生成树。为了实现输出所有不同构的最小生成树，基于 closedge 辅助数组采用向下递推与向上回溯，反复尝试输出所有可能的最小生成树的边，详见 2.3。

为了减少源程序代码量，主要存储结构全部采用全局变量进行定义且所有数组空间均不采用动态存储分配方式建立，程序主函数实现全部功能代码（不采用任何子函数）。

主函数执行流程如图 1 所示。

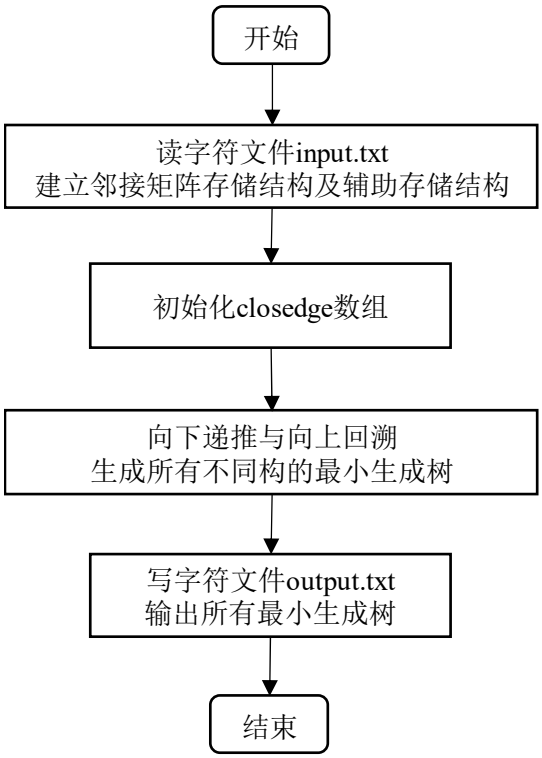

图 1 主函数流程图

为了实现最小生成树的表示与存储，每棵生成树用一个 32bit 整数（掩码）表示。对于 n 个顶点 e 条边的连通网，将 e 条边按输入文件中各边的输入次序依次编号为 0~e-1，对于第 q 条边（q=0, 1, … , e-1），若它包含于一棵最小生成树中，则令该生成树的 32bit 掩码的第 q 位为 1，否则掩码的第 q 位为 0。因为生成树的边数为 n-1，每个表示最小生成树的掩码中有且仅有 n-1 个 bit 位为 1。用掩码表示生成树边集合的优点是很容易判断两棵最小生成树是否同构，因

为同构的最小生成树它们的掩码是相同的。由于程序中掩码采用 int 型定义（int 型变量长度只为 32bit），程序必需限定连通网的边数 *e* 不能超过 32。

在用 Prim 算法求最小生成树时，为了产生所有不同构的最小生成树，每趟循环用 MST 性质加入一个顶点至集合 *S* 中（称为向下递推）时，需对每一条穿越集合 *S* 边界的最小权重边进行逐一选择（普通 Prim 算法只需任选其中一条最小权重边即可）。为此，算法引入辅助一维数组 p[0..n-1]，其元素 p[i]（i=0, 1, 2, …, n-2）满足

$$p[i]=\arg\min_{j}\{closedge[j].lowcost|closedge[j].lowcost>0且closedge[j].lowcost>0的j有且仅有n-i-1个\} \quad (式\ 1)$$

因此，p[i]记录了有 i+1 个顶点已加入集合 S 时（称为第 i 趟），集合 S 外所有顶点中，拥有至少一条连入集合 S 内权重最小边的某个顶点。

为了方便向上回溯时恢复上一趟的 closedge 数组值，算法令 closedge[i][p[i]].lowcost=-closedeg[i][p[i]].lowcost（lowcost 值由正变负）表示顶点 p[i]加入集合 S（由 i 趟向下递推至 i+1 趟）；反之，执行同样的语句（lowcost 值由负变正）表示顶点 p[i]从集合 S 中删除（由 i+1 趟向上回溯至第 i 趟）。向下递推与向上回溯时，由于顶点 p[i]加入或离开了集合 S，除了通过求负运算标记 close[p[i]].lowcost 由正变负或由负变正，还必须对所有 j∉S，考察 closedge[j]的信息是否需要更新，详细操作详见 2.3。总之，对于 closedge 数组的第 i 趟信息，closedge[0..n-1].lowcost 中有且仅有 i+1 个 lowcost 值小于 0，对应有 i+1 个顶点已加入集合 S。特别地，当 i=0 时，closedge[0..n-1].lowcost 中只有 closedge[k].lowcost<0，表示仅出发顶点 k 一个顶点已加入集合 S；当 i=n-1 时，所有 closedge[0..n-1].lowcost 值均小于 0，表示 n 个顶点已经全部加入了集合 S，即已经成功生成了一棵最小生成树。

2.2 详细数据结构设计

1. 邻接矩阵存储结构(均为全局变量)

n: 顶点数

e: 边数(要求 e≤32，因为用一个 32bit 整型值作为掩码，表示一棵生成树中的边集合)

w[i][j]: 边(i,j)上的权值（即邻接矩阵）

2. Prim 算法存储结构(均为全局变量)

k: Prim 算法出发顶点下标

closedge[0..n-1]: 每个元素有两个分量，vex 和 lowcost，含义如下：

closedge[j].lowcost>0，则顶点 j∉S，且

$$\begin{cases} closedge[j].lowcost=\min\limits_{v\in S} w[j][v] \\ closedge[j].vex=u,满足u\in S且w[j][u]=closedge[j].lowcost \end{cases} \quad (式\ 2)$$

closedge[j].lowcost<0，则顶点 j∈S。

3. Prim 算法辅助存储结构(均为全局变量)

p[i]: 记录有 i+1 个顶点已加入集合 S 时（称为第 i 趟），集合 S 外所有顶点中，拥有至少一条连入 S 内权重最小边的某个顶点，计算方法见(式 1)。

4. 最小生成树的掩码表示法所需辅助存储结构(均为全局变量)

t[0..m-1]: 生成的 m 棵不同构的最小生成树的掩码

m: 当前已生成的最小生成树的数目

r[i][j]: r[i][j]=-1 表示边(i,j)不属于图的边集 E；否则，0<=r[i][j]<e，表示边(i,j)的掩码位置为第 r[i][j]位，当掩码的 r[i][j]位为 1 时，表示边(i, j)属于该生成树；

v[q][0..1]：其中 q=0,1,...,e-1，v[q][0],v[q][1]分别表示第 q 条边的两个顶点，即边(v[q][0],v[q][1])

5. 输入文件 input.txt 格式

n　e

i　j　w_{ij}

… (三元组共 e 行)

6. 输出文件 output.txt 信息

不同构的生成树数目；每棵生成树的 n-1 边（边用顶点无序偶表示）；每棵生成树 n-1 边的权重之和。

2.3 详细算法设计

1. 任意第 i 趟逐一选择穿越集合 S 边界的最小权重边

由(式 1)知，满足(式 1)的 p[i]不一定惟一，由(式 2)知，满足(式 2)的 closedge[j].vex 也不一定惟一。所以，为了生成所有不同构的最小生成树，对任意第 i 趟（集合 S 口中已加入了 i+1 顶点，i=0, 1, … , n-2），需对满足(式 1)的所有 p[i]进行循环，对每个 p[i]，需对满足(式 2)的每户个 closedge[p[i]].vex 进行循环， 每选择一对 p[i]以及 closedge[p[i]].vex，则向下递推到第 i+1 趟，直至第 n-1 趟，实现一次生成树的输出。从第 n-1 趟开始，向上回溯，当回溯到任意第 i 趟时（i=0, 1, 2, … , n-2），根据 p[i]以及 closedge[p[i]].vex，可以选择下一对满足(式 1)和(式 2)的 p[i]以及 closedge[p[i]].vex，具体做法是：

先在 0..n-1 范围内查找最小的 p[i]满足(式 1)；

对每一个新的 p[i]，先在 0..n-1 范围内查找最小的 closedege[p[i]].vex 满足(式 2)；

固定 p[i]不变，在 closedge[p[i]].vex+1..n-1 范围内找新的最小的 closedge[p[i]].vex 满足(式 2)，若 closedge[p[i]].vex+1..n-1 范围内无 closedge[p[i]].vex 满足(式 2)，则在 p[i]+1..n-1 范围内找最小的 p[i]满足(式 1)，当 p[i]+1..n-1 范围内无新的 p[i]满足(式 1)，则当前第 i 趟需向上回溯至第 i-1 趟；

若回溯至 i=-1 趟，则算法停止。

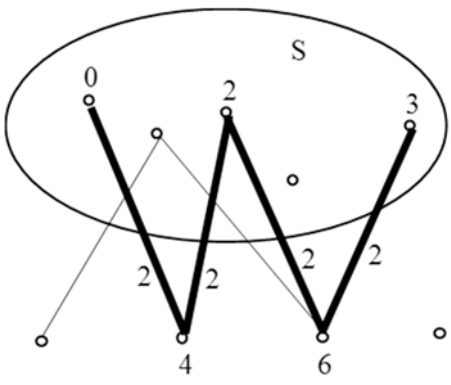

图 2 p[i]与 closedge[p[i]].vex 的示例

对任意第 i 趟，满足(式 1)和(式 2)的 p[i]，closedge[p[i]].vex 的一种情形如图 2 所示。图 2 中，假设穿越集合 S 边界的最小权重边共有 4 条，最小权重边上的权重为 2（图中加粗的黑边）。图 2 中，对集合 S 外的顶点，p[i]可选 4，6。当 p[i]=4 时，可选的 closedge[p[i]].vex 有 0，2；若 p[i]=6，则可选的 closedge[p[i]].vex 有 2，3。对于图 2 所示的情况，选择 4 条加粗黑边之一，则向下递推至 i+1 趟一次；当 i+1 趟回溯至第 i 趟，则搜索下一条加粗的黑边，然后再次

向下递推一次，若 4 条加粗的黑边都向下递推了一次（搜索第 5 条加粗黑边失败），则需向上回溯至 i-1 层。

2. 向下递推与向上回溯 Prim 算法

完整的向下递推与向上回溯的 Prim 算法流程如图 3 所示。

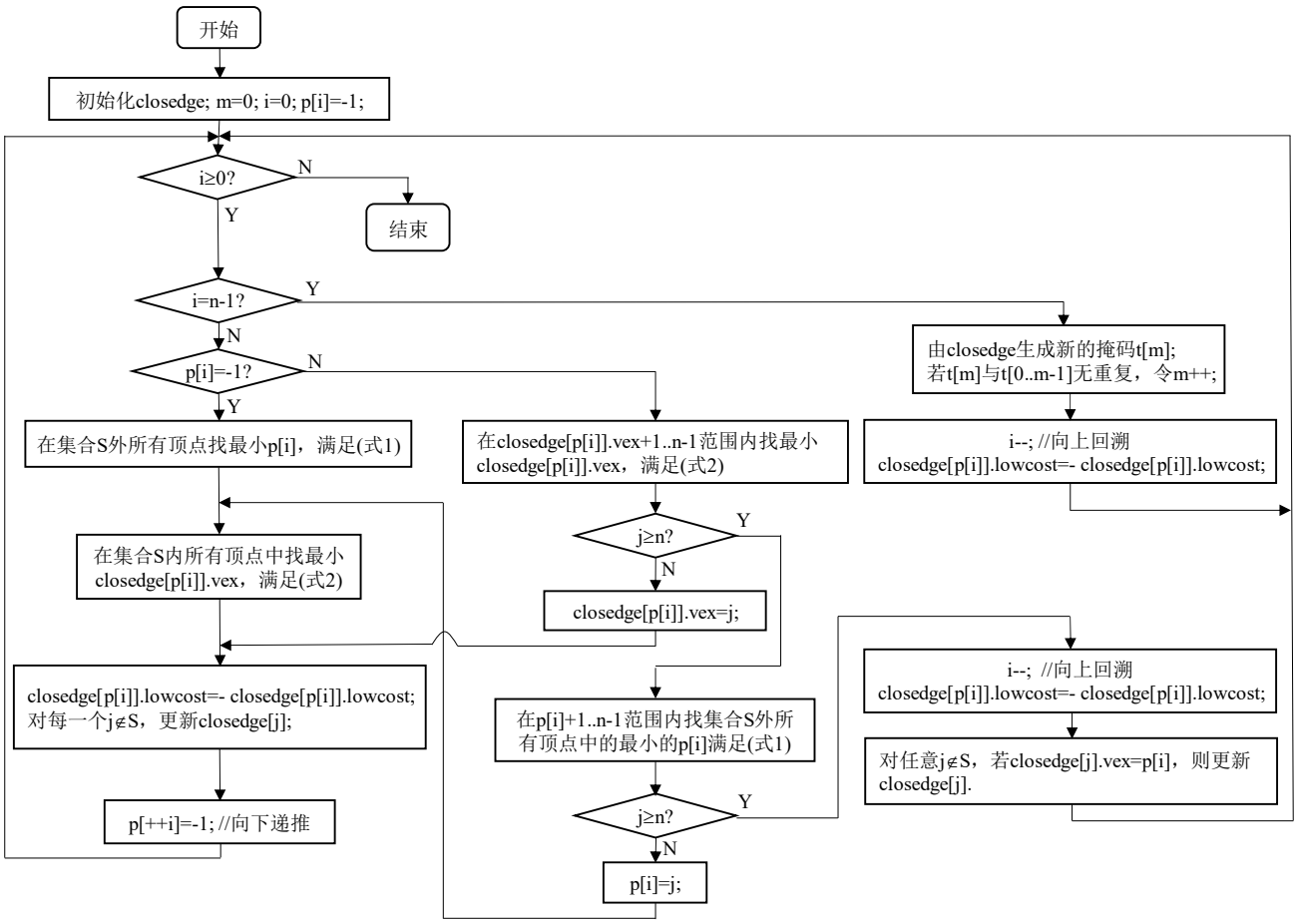

图 3 向下递推与向上回溯 Prim 算法流程图

图 3 中，向下递推时，首先令 p[i]=-1，表示当前第 i 趟满足(式 2)的第一个 p[i]尚未求得。而向上回溯时，若回溯出发点 i<n-1，则执行 closedge[p[i]].lowcost=-closedge[p[i]].lowcost;（顶点 p[i]离开集合 S，恢复成为 S 外的顶点），还需对任意 j∉S，若 closedge[j].vex=p[i]，则更新 closedge[j]，更新的方法如下面的(式 3)所示。

$$\begin{cases} \text{closedge[j].lowcost}=\min\limits_{v\in S} \text{w[j][v]} \\ \text{closedge[j].vex}=\arg\min\limits_{v\in S} \text{w[j][v]} \end{cases} \quad \text{(式 3)}$$

若回溯出发点为 i=n-1，则不需要执行(式 3)，因为 p[n-2]为最后加入集合 S 的顶点，当 p[n-2]离开集合 S 后（由集合 S 内变为集合 S 外），集合外无其它顶点 j 满足 j∉S 且 closedge[j].vex=p[n-2]。

执行图 3 所示算法前，顶点 k（出发顶点）已加入集合 S，向下递推过程中，顶点序列 p[0], p[1], … , p[n-2]（共 n-1 个顶点）依次加入集合 S，当 i=n-1 时，输出一次最小生成树。向上回溯一次，则一个顶点离开集合 S，当回溯至 i=-1 时，算法停止。

3. 生成树掩码生成算法

(1) t[m] = 0; //掩码初值置 0

(2) for (j = 0; j < n; j++) //置生成树中的边(j,closedge[n-2][j].vex)的掩码为 1

if (j != k) t[m]|=1<<r[j][closedge[n-2][j].vex];

(3) for (q = 0; q < m; q++) if (t[m] == t[q]) break; //搜索 t[m]与 t[0..m-1]是否有重复

(4) if (q >= m) m++;

3. 程序测试报告

为了测试程序的正确性，共使用了三个测试用例，测试结果如下。

1) 测试用例 1

输入数据文件内容如下：

6 10
0 1 6
0 2 1
0 3 5
1 2 5
1 4 3
2 3 5
2 4 6
2 5 4
3 5 2
4 5 6

期望输出：有且仅有一棵最小生成树，生成树的 5 条边为(0,2)(1,2)(1,4)(2,5)(3,5)。

程序实际输出：

所有不同构的最小生成树共有 1 棵:

第 1 棵:(0,2)(1,2)(1,4)(2,5)(3,5), 最小生成树权重和=15

测试结论：程序正确

2) 测试用例 2

输入数据文件内容如下：

9 14
0 1 4
0 7 4
1 2 3
1 7 11
2 3 7
2 5 4
2 8 2
3 4 9
3 5 14
4 5 9

5 6 2

6 7 1

6 8 6

7 8 7

期望输出：未人工求取

程序实际输出：

所有不同构的最小生成树共有 6 棵:

第 1 棵:(0,1)(1,2)(2,3)(2,5)(2,8)(3,4)(5,6)(6,7), 最小生成树权重和=32

第 2 棵:(0,1)(1,2)(2,3)(2,5)(2,8)(4,5)(5,6)(6,7), 最小生成树权重和=32

第 3 棵:(0,1)(0,7)(1,2)(2,3)(2,8)(3,4)(5,6)(6,7), 最小生成树权重和=32

第 4 棵:(0,1)(0,7)(1,2)(2,3)(2,8)(4,5)(5,6)(6,7), 最小生成树权重和=32

第 5 棵:(0,7)(1,2)(2,3)(2,5)(2,8)(3,4)(5,6)(6,7), 最小生成树权重和=32

第 6 棵:(0,7)(1,2)(2,3)(2,5)(2,8)(4,5)(5,6)(6,7), 最小生成树权重和=32

测试结论：6 棵最小生成树画图如图 4 所示，经人工检验，程序输出正确。

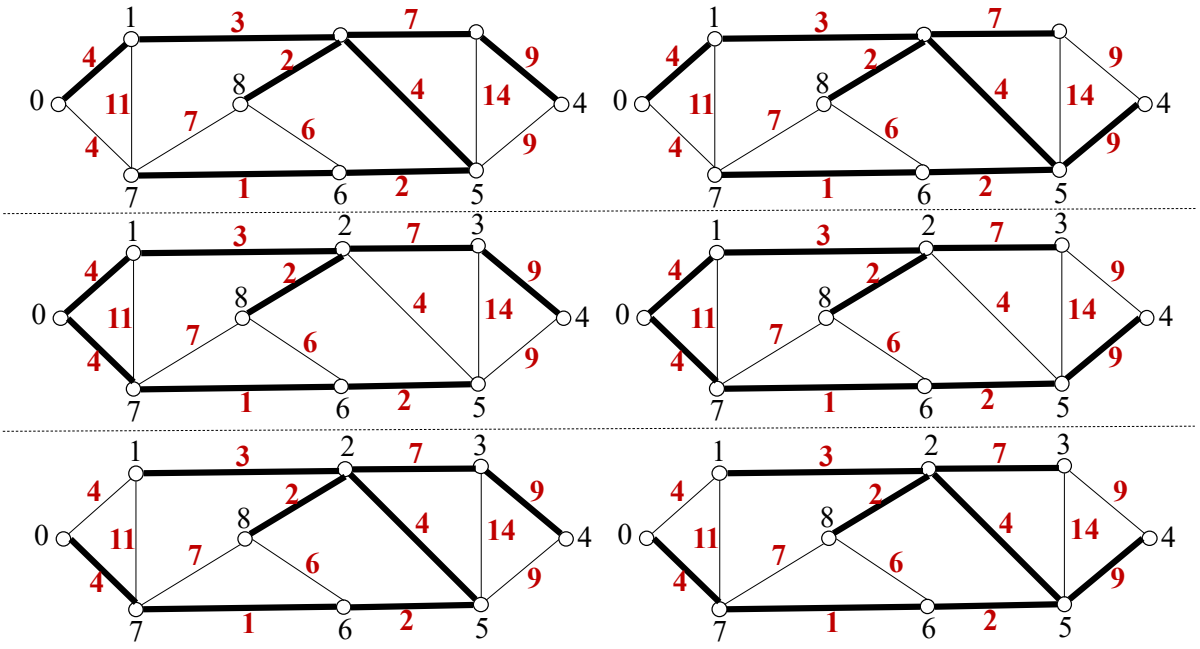

图 4 测试用例 2 的 6 棵不同构的最小生成树

3) 测试用例 3

数据输入文件内容：

将测试用例 1 中，所有边的权重统一修改为 8，即

6 10

0 1 8

0 2 8

0 3 8

1 2 8

1 4 8
2 3 8
2 4 8
2 5 8
3 5 8
4 5 8

期望输出：该图所有不同构的生成树（数目人工无法求取）

程序实际输出：

所有不同构的最小生成树共有 121 棵:

第 1 棵:(0,1)(0,2)(0,3)(1,4)(2,5), 最小生成树权重和=40

第 2 棵:(0,1)(0,2)(0,3)(1,4)(3,5), 最小生成树权重和=40

第 3 棵:(0,1)(0,2)(0,3)(1,4)(4,5), 最小生成树权重和=40

第 4 棵:(0,1)(0,2)(0,3)(2,4)(2,5), 最小生成树权重和=40

第 5 棵:(0,1)(0,2)(0,3)(2,4)(3,5), 最小生成树权重和=40

第 6 棵:(0,1)(0,2)(0,3)(2,4)(4,5), 最小生成树权重和=40

第 7 棵:(0,1)(0,2)(0,3)(2,5)(4,5), 最小生成树权重和=40

第 8 棵:(0,1)(0,2)(0,3)(3,5)(4,5), 最小生成树权重和=40

第 9 棵:(0,1)(0,2)(1,4)(2,3)(2,5), 最小生成树权重和=40

第 10 棵:(0,1)(0,2)(1,4)(2,3)(3,5), 最小生成树权重和=40

第 11 棵:(0,1)(0,2)(1,4)(2,3)(4,5), 最小生成树权重和=40

第 12 棵:(0,1)(0,2)(2,3)(2,4)(2,5), 最小生成树权重和=40

第 13 棵:(0,1)(0,2)(2,3)(2,4)(3,5), 最小生成树权重和=40

第 14 棵:(0,1)(0,2)(2,3)(2,4)(4,5), 最小生成树权重和=40

第 15 棵:(0,1)(0,2)(2,3)(2,5)(4,5), 最小生成树权重和=40

第 16 棵:(0,1)(0,2)(2,3)(3,5)(4,5), 最小生成树权重和=40

第 17 棵:(0,1)(0,2)(1,4)(2,5)(3,5), 最小生成树权重和=40

第 18 棵:(0,1)(0,2)(1,4)(3,5)(4,5), 最小生成树权重和=40

第 19 棵:(0,1)(0,2)(2,4)(2,5)(3,5), 最小生成树权重和=40

第 20 棵:(0,1)(0,2)(2,4)(3,5)(4,5), 最小生成树权重和=40

第 21 棵:(0,1)(0,2)(2,5)(3,5)(4,5), 最小生成树权重和=40

第 22 棵:(0,1)(0,3)(1,2)(1,4)(2,5), 最小生成树权重和=40

第 23 棵:(0,1)(0,3)(1,2)(1,4)(3,5), 最小生成树权重和=40

第 24 棵:(0,1)(0,3)(1,2)(1,4)(4,5), 最小生成树权重和=40

第 25 棵:(0,1)(0,3)(1,2)(2,4)(2,5), 最小生成树权重和=40

第 26 棵:(0,1)(0,3)(1,2)(2,4)(3,5), 最小生成树权重和=40

第 27 棵:(0,1)(0,3)(1,2)(2,4)(4,5), 最小生成树权重和=40

第 28 棵:(0,1)(0,3)(1,2)(2,5)(4,5), 最小生成树权重和=40

第 29 棵:(0,1)(0,3)(1,2)(3,5)(4,5), 最小生成树权重和=40

第 30 棵:(0,1)(1,2)(1,4)(2,3)(2,5), 最小生成树权重和=40
第 31 棵:(0,1)(1,2)(1,4)(2,3)(3,5), 最小生成树权重和=40
第 32 棵:(0,1)(1,2)(1,4)(2,3)(4,5), 最小生成树权重和=40
第 33 棵:(0,1)(1,2)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 34 棵:(0,1)(1,2)(2,3)(2,4)(3,5), 最小生成树权重和=40
第 35 棵:(0,1)(1,2)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 36 棵:(0,1)(1,2)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 37 棵:(0,1)(1,2)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 38 棵:(0,1)(1,2)(1,4)(2,5)(3,5), 最小生成树权重和=40
第 39 棵:(0,1)(1,2)(1,4)(3,5)(4,5), 最小生成树权重和=40
第 40 棵:(0,1)(1,2)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 41 棵:(0,1)(1,2)(2,4)(3,5)(4,5), 最小生成树权重和=40
第 42 棵:(0,1)(1,2)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 43 棵:(0,1)(0,3)(1,4)(2,3)(2,5), 最小生成树权重和=40
第 44 棵:(0,1)(0,3)(1,4)(2,3)(3,5), 最小生成树权重和=40
第 45 棵:(0,1)(0,3)(1,4)(2,3)(4,5), 最小生成树权重和=40
第 46 棵:(0,1)(0,3)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 47 棵:(0,1)(0,3)(2,3)(2,4)(3,5), 最小生成树权重和=40
第 48 棵:(0,1)(0,3)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 49 棵:(0,1)(0,3)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 50 棵:(0,1)(0,3)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 51 棵:(0,1)(0,3)(1,4)(2,4)(2,5), 最小生成树权重和=40
第 52 棵:(0,1)(0,3)(1,4)(2,4)(3,5), 最小生成树权重和=40
第 53 棵:(0,1)(0,3)(1,4)(2,4)(4,5), 最小生成树权重和=40
第 54 棵:(0,1)(0,3)(1,4)(2,5)(3,5), 最小生成树权重和=40
第 55 棵:(0,1)(0,3)(1,4)(2,5)(4,5), 最小生成树权重和=40
第 56 棵:(0,1)(0,3)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 57 棵:(0,1)(0,3)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 58 棵:(0,1)(0,3)(2,4)(3,5)(4,5), 最小生成树权重和=40
第 59 棵:(0,1)(1,4)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 60 棵:(0,1)(1,4)(2,3)(2,4)(3,5), 最小生成树权重和=40
第 61 棵:(0,1)(1,4)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 62 棵:(0,1)(1,4)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 63 棵:(0,1)(1,4)(2,4)(3,5)(4,5), 最小生成树权重和=40
第 64 棵:(0,1)(1,4)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 65 棵:(0,1)(1,4)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 66 棵:(0,1)(1,4)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 67 棵:(0,2)(0,3)(1,2)(1,4)(2,5), 最小生成树权重和=40

第 68 棵:(0,2)(0,3)(1,2)(1,4)(3,5), 最小生成树权重和=40
第 69 棵:(0,2)(0,3)(1,2)(1,4)(4,5), 最小生成树权重和=40
第 70 棵:(0,2)(0,3)(1,2)(2,4)(2,5), 最小生成树权重和=40
第 71 棵:(0,2)(0,3)(1,2)(2,4)(3,5), 最小生成树权重和=40
第 72 棵:(0,2)(0,3)(1,2)(2,4)(4,5), 最小生成树权重和=40
第 73 棵:(0,2)(0,3)(1,2)(2,5)(4,5), 最小生成树权重和=40
第 74 棵:(0,2)(0,3)(1,2)(3,5)(4,5), 最小生成树权重和=40
第 75 棵:(0,2)(1,2)(1,4)(2,3)(2,5), 最小生成树权重和=40
第 76 棵:(0,2)(1,2)(1,4)(2,3)(3,5), 最小生成树权重和=40
第 77 棵:(0,2)(1,2)(1,4)(2,3)(4,5), 最小生成树权重和=40
第 78 棵:(0,2)(1,2)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 79 棵:(0,2)(1,2)(2,3)(2,4)(3,5), 最小生成树权重和=40
第 80 棵:(0,2)(1,2)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 81 棵:(0,2)(1,2)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 82 棵:(0,2)(1,2)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 83 棵:(0,2)(1,2)(1,4)(2,5)(3,5), 最小生成树权重和=40
第 84 棵:(0,2)(1,2)(1,4)(3,5)(4,5), 最小生成树权重和=40
第 85 棵:(0,2)(1,2)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 86 棵:(0,2)(1,2)(2,4)(3,5)(4,5), 最小生成树权重和=40
第 87 棵:(0,2)(1,2)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 88 棵:(0,2)(0,3)(1,4)(2,4)(2,5), 最小生成树权重和=40
第 89 棵:(0,2)(0,3)(1,4)(2,4)(3,5), 最小生成树权重和=40
第 90 棵:(0,2)(0,3)(1,4)(2,4)(4,5), 最小生成树权重和=40
第 91 棵:(0,2)(0,3)(1,4)(2,5)(4,5), 最小生成树权重和=40
第 92 棵:(0,2)(0,3)(1,4)(3,5)(4,5), 最小生成树权重和=40
第 93 棵:(0,2)(1,4)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 94 棵:(0,2)(1,4)(2,3)(2,4)(3,5), 最小生成树权重和=40
第 95 棵:(0,2)(1,4)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 96 棵:(0,2)(1,4)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 97 棵:(0,2)(1,4)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 98 棵:(0,2)(1,4)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 99 棵:(0,2)(1,4)(2,4)(3,5)(4,5), 最小生成树权重和=40
第 100 棵:(0,2)(1,4)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 101 棵:(0,3)(1,2)(1,4)(2,3)(2,5), 最小生成树权重和=40
第 102 棵:(0,3)(1,2)(1,4)(2,3)(3,5), 最小生成树权重和=40
第 103 棵:(0,3)(1,2)(1,4)(2,3)(4,5), 最小生成树权重和=40
第 104 棵:(0,3)(1,2)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 105 棵:(0,3)(1,2)(2,3)(2,4)(3,5), 最小生成树权重和=40

第 106 棵:(0,3)(1,2)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 107 棵:(0,3)(1,2)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 108 棵:(0,3)(1,2)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 109 棵:(0,3)(1,4)(2,3)(2,4)(2,5), 最小生成树权重和=40
第 110 棵:(0,3)(1,4)(2,3)(2,4)(3,5), 最小生成树权重和=40
第 111 棵:(0,3)(1,4)(2,3)(2,4)(4,5), 最小生成树权重和=40
第 112 棵:(0,3)(1,4)(2,3)(2,5)(4,5), 最小生成树权重和=40
第 113 棵:(0,3)(1,4)(2,3)(3,5)(4,5), 最小生成树权重和=40
第 114 棵:(0,3)(1,2)(1,4)(2,5)(3,5), 最小生成树权重和=40
第 115 棵:(0,3)(1,2)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 116 棵:(0,3)(1,2)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 117 棵:(0,3)(1,4)(2,4)(2,5)(3,5), 最小生成树权重和=40
第 118 棵:(0,3)(1,4)(2,5)(3,5)(4,5), 最小生成树权重和=40
第 119 棵:(0,3)(1,2)(1,4)(3,5)(4,5), 最小生成树权重和=40
第 120 棵:(0,3)(1,4)(2,4)(3,5)(4,5), 最小生成树权重和=40
第 121 棵:(0,3)(1,2)(2,4)(3,5)(4,5), 最小生成树权重和=40

测试结论：因为本测试用例的输出等价于所有不同构的生成树，为了检验输出是否正确，另编写一个全部生成树的生成程序，该程序产生 10 中取 5 组合数（10 条边中任取 5 条边），以全部顶点和取出的 5 条边建立邻接矩阵，用 Warshall 算法求可达矩阵，若可达矩阵所有元素均为 1（说明 5 条边构成连通图），因为顶点数为 6，故必为生成树；否则，若可达矩阵存在 0 元素，说明 5 条边不构成生成树。于是，该程序可以输出所有生成树，每棵生成树以掩码方式存储。该生成树生成程序同样输出了 121 棵生成树。对最小生成树程序输出的 121 棵最小生成树掩码以及基于排列组合（穷举）以及可达矩阵检验的生成树生成程序输出的 121 棵生成树掩码分别进行由小到大排序，将排序结果输出到一个字符文件，人工检验发现两组掩码完全相同。由此得出结论，程序输出正确。

4. 结论

1) 经过三个测试用例，特别是通过测试用例 3 的检验，说明程序正确。
2) 算法辅助存储空间为 $S(n)=O(n^2+n+e)$。
3) 算法最坏时间复杂度近似为 $O(C_{n(n-1)/2}^{n-1}\times n^2)$，而最佳时间复杂度为 $O(n^2)$。