### 3.3Prim算法的设计

#### 3.3.1 题目以及要求

**题目描述：**

在使用图的邻接矩阵*ADT*的基础上，设计能指定起始结点*u*的*Prim*算法，用以求无向网的最小生成树，并以文本形式输出生成树中各条边以及它们的权值。将此算法加入到邻接矩阵*ADT*中，在邻接矩阵*ADT*中提供一个公有的成员函数*Prim(u, adjvex, lowcost)*。

提示：

（1）*Prim*算法的基本思想是从结点的角度出发。初始时，生成树中一个结点也没有，然后将一个个结点加入生成树，直到所有的结点都被加入，生成树就形成了。具体来讲，*Prim*算法由以下几个步骤组成：

1）初始时，生成树的结点集*U*为空；

2）随意选择一个结点，加入结点集*V*，然后重复下列工作，直到*U=V*；

3）把*(u,v)*加入生成树的边集，*v*加入到*U*。

（2）在*Prim*算法的执行过程中，需要保存哪些结点在*U*中，哪些结点不在*U*中的信息。这可以用一个布尔型的一维数组*flag*来保存。如果结点i已在生成树中，则*flag[i]*的值为*true*，否则为*false*。

（3）在*Prim*算法的执行过程中，还需要保存*U*中的结点到*V-U*中结点的权值最小的边，这可以用2个一维数组*lowcost*和*adjvex*来记录。*lowcost[i]*表示*U*中的结点到结点*i*的所有边中的最小权值。*adjvex[i]*表示从*U*中的哪一个结点出发到结点*i*的权值是最小的。

参考函数原型：

*template<class TypeOfVer, class TypeOfEdge>*

*bool adjmatrix\_graph<TypeOfVer, TypeOfEdge>::Prim(int u, int adjvex[], TypeOfEdge lowcost[]);*

**输出说明：**

第一行：图的类型

第二行：顶点集

空行

第三行：邻接矩阵（列与列之间用格式控制符'\t'分隔）

第四行：各条边以及它们的权值，输出格式为：

(c,b,5),(d,c,3),(e,d,8),(a,e,14),(d,f,21),(e,g,16)

其中，每个括号内的信息为一条边，格式为（邻接点, 自己, 权值），边的输出顺序为按照（自己这个）点的编号顺序输出。“自己这个”不包括起始点，比如范例中输出的边不包括起始点a，也就是没有(\*,a,\*)。

**3.3.2 概要设计**

设计一个*adjmartix\_graph*类，其中抽象数据类型包括：

数据对象：V为顶点数据类型，E为边的数据类型。

基本操作：

*adjmartix\_graph（）*构造函数，对图进行初始化。

*Prim（u）*普里姆算法，从输入的u结点开始寻找权值最小边并纳入最小生成树。

*print\_edge()*输出图的邻接矩阵*。*

*print\_vers()*输出结点集*。*

*get\_graph\_type()*得到图的类型*。*

基于*adjmartix\_graph*类，设计*make\_adjmartix\_graph（）*函数对图进行构造。

**3.3.3 算法分析**

*算法说明：*基于图类设计抽象数据类型*Prim（）*函数，从输入的一个节点出发，依次访问没有访问过的节点，通过比较权值将权值最小的边纳入最小生成树，直到所有结点都被访问过。

以下图为例:

图示

描述已自动生成

图片包含 图示

描述已自动生成

该算法需要注意几个难点:

1.如何区分没访问过的节点与访问过的节点:设置vis数组,用来标记该结点是否被访问过——用于遍历邻接矩阵时区别是否是访问过的节点,设置has\_visited数组存放已经访问顶点在节点集合中的位序——用来判断循环终止条件,即当has\_visited数组的大小等于顶点集数组大小时候,Prim算法的循环将终止;

2.如何判断每次要加入的顶点:只需要遍历has\_visited数组中的每一个已访问顶点的位序,用他们取探索未访问的顶点即可;

3.如何存放结果——小组使用**map<V,pair<V,E>>*res***存放结果,其三个参数分别为自身节点,邻接顶点,权值,最后使用迭代器输出

4.如何控制输出格式——有时候会遇到比如map中已经存在一个键值d对应(e,d,8),但是之后又有同样的键值d对应着(f,d,21),尽管fd也是我们要找的边,但是我们需要将键值从d变为f,利用map中的count函数可以判断是否已经存在该键值;

5.map模板类的输出——在使用模板类stl迭代器输出时候如本题应该在前加入typename关键字,如本题中是用**for(typename std::map<V,pair<V,E> >::iterator it =res.begin();it!=res.end();it++)**进行输出的.

*算法时间复杂度：*外层循环需要寻找条边，故外层循环时间复杂度为**，循环内，选择权值最短的边，故算法的时间复杂度为*。*

**3.3.4 使用说明**

**输入说明：**

第一行：图的类型

第二行：结点数

第三行：结点集

第四行：无边标记

第五行：边数

第六行：边集

第七行：权集

第八行：起始结点序号u

程序名称为*Prim.cpp* , 程序在*Clion 2023.1,MVSC 17.0*编译器下能正常调试运行,程序运行如图4.3

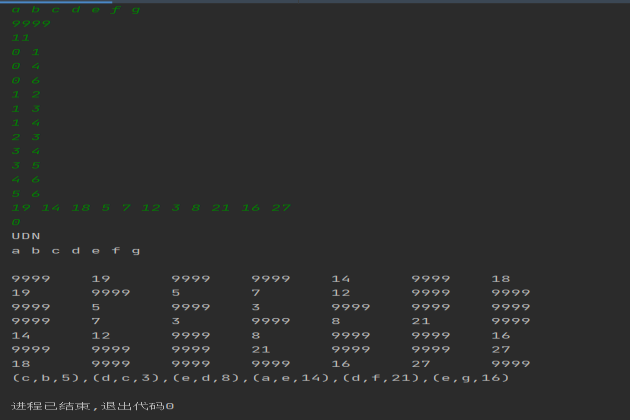


图4.3：运行截图

在输入完数据后,需要使用键盘的*Ctrl+D*(程序内)/*Ctrl+Z*(外部*exe*)即可显示结果。

**3.3.5测试结果以及分析**

* 第一组数据:已经自成一棵树,起始点序号为原点,简图如下:

图表

描述已自动生成

IN:

UDN

7

V1 V2 V3 V4 V5 V6 V7

9999

6

0 1

0 2

1 3

2 4

3 5

4 6

1 3 2 4 5 6

0

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7

9999 1 3 9999 9999 9999 9999

1 9999 9999 2 9999 9999 9999

3 9999 9999 9999 4 9999 9999

9999 2 9999 9999 9999 5 9999

9999 9999 4 9999 9999 9999 6

9999 9999 9999 5 9999 9999 9999

9999 9999 9999 9999 6 9999 9999

(V3,V1,3),(V1,V2,1),(V5,V3,4),(V2,V4,2),(V7,V5,6),(V4,V6,5)

* 第二组数据:成一个环,起始点不为序号原点,简图如下

图表, 形状, 雷达图, 正方形

描述已自动生成

IN:

UDN

7

V1 V2 V3 V4 V5 V6 V7

9999

7

0 1

1 2

2 3

3 4

4 5

5 6

6 0

1 3 2 4 5 6 8

2

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7

9999 1 9999 9999 9999 9999 8

1 9999 3 9999 9999 9999 9999

9999 3 9999 2 9999 9999 9999

9999 9999 2 9999 4 9999 9999

9999 9999 9999 4 9999 5 9999

9999 9999 9999 9999 5 9999 6

8 9999 9999 9999 9999 6 9999

(V1,V2,1),(V2,V3,3),(V3,V4,2),(V4,V5,4),(V5,V6,5),(V6,V7,6)

* 第三组数据,普通图,输入数据为int,输入量适中,起始点为源点,简图如下:

形状, 多边形

描述已自动生成

IN:

UDN

7

V1 V2 V3 V4 V5 V6 V7

9999

13

0 1

0 2

0 4

0 6

1 2

1 5

2 3

2 6

3 4

3 5

4 5

4 6

5 6

1 4 2 3 6 7 8 10 2 4 5 23 7

0

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7

9999 1 4 9999 2 9999 3

1 9999 6 9999 9999 7 9999

4 6 9999 8 9999 9999 10

9999 9999 8 9999 2 4 9999

2 9999 9999 2 9999 5 23

9999 7 9999 4 5 9999 7

3 9999 10 9999 23 7 9999

(V5,V1,2),(V1,V2,1),(V1,V3,4),(V6,V4,4),(V4,V5,2),(V1,V7,3)

* 第四组数据:普通图,输入数据类型为double,为了测试程序的广泛性,起始点不为源点,简图如下:

形状, 多边形

描述已自动生成

IN:

UDN

7

V1 V2 V3 V4 V5 V6 V7

9999

13

0 1

0 2

0 4

0 6

1 2

1 5

2 3

2 6

3 4

3 5

4 5

4 6

5 6

1.1 4.2 2 3.5 6.9 7 8.1 10.2 2.1 4 5 23 7

0

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7

9999 1.1 4.2 9999 2 9999 3.5

1.1 9999 6.9 9999 9999 7 9999

4.2 6.9 9999 8.1 9999 9999 10.2

9999 9999 8.1 9999 2.1 4 9999

2 9999 9999 2.1 9999 5 23

9999 7 9999 4 5 9999 7

3.5 9999 10.2 9999 23 7 9999

(V5,V1,2),(V1,V2,1.1),(V1,V3,4.2),(V6,V4,4),(V4,V5,2.1),(V1,V7,3.5)

* 第五组数据:图的类型不为”UDN”,直接不执行prim算法部分,测试鲁棒性

IN:

DN

7

V1 V2 V3 V4 V5 V6 V7

9999

13

0 1

0 2

0 4

0 6

1 2

1 5

2 3

2 6

3 4

3 5

4 5

4 6

5 6

1.1 4.2 2 3.5 6.9 7 8.1 10.2 2.1 4 5 23 7

0

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7

9999 1.1 4.2 9999 2 9999 3.5

1.1 9999 6.9 9999 9999 7 9999

4.2 6.9 9999 8.1 9999 9999 10.2

9999 9999 8.1 9999 2.1 4 9999

2 9999 9999 2.1 9999 5 23

9999 7 9999 4 5 9999 7

3.5 9999 10.2 9999 23 7 9999

* 第六组数据,全是孤立的点,则也不应该执行prim算法,或者prim算法得到的结果为空,测试鲁棒性,简图如下:

图表, 散点图

描述已自动生成

IN:

UDN

7

V1 V2 V3 V4 V5 V6 V7

9999

0

0

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7

9999 9999 9999 9999 9999 9999 9999

9999 9999 9999 9999 9999 9999 9999

9999 9999 9999 9999 9999 9999 9999

9999 9999 9999 9999 9999 9999 9999

9999 9999 9999 9999 9999 9999 9999

9999 9999 9999 9999 9999 9999 9999

9999 9999 9999 9999 9999 9999 9999

第七组数据:大数据压力测试680KB读入23ms运行成功,本样例考察时间复杂度是否良好,故数据类型全为int,且从源点开始,本题简图如下:

图示

描述已自动生成

IN:

UDN

20

V1 V2 V3 V4 V5 V6 V7 V8 V9 APPLE BANANA PEACH GRAPE PEOPLE FATHER ZLC LZR LX WQ LJL

9999

130

0 1

0 2

0 3

0 4

0 5

0 9

0 10

0 12

0 13

0 14

0 19

1 2

1 4

1 5

1 6

1 7

1 9

1 10

1 11

1 12

1 13

1 14

1 17

1 18

1 19

2 3

2 4

2 5

2 7

2 8

2 9

2 10

2 11

2 13

2 15

2 17

2 19

3 4

3 6

3 7

3 8

3 9

3 10

3 14

3 13

3 18

3 19

4 5

4 6

4 7

4 8

4 9

4 10

4 11

4 12

4 13

4 14

4 15

4 16

4 17

4 19

5 6

5 7

5 8

5 9

5 11

5 12

5 13

5 14

5 15

5 16

5 19

6 7

6 8

6 9

6 10

6 13

6 15

6 19

7 9

7 10

7 11

7 13

7 15

7 17

7 19

8 9

8 10

8 11

8 12

8 13

8 14

8 15

8 16

8 19

9 10

9 11

9 12

9 13

9 14

9 15

9 16

9 17

9 18

9 19

10 11

10 14

10 19

11 12

11 13

11 15

11 16

11 17

11 18

12 16

12 17

12 19

13 14

13 15

13 16

13 19

14 15

14 16

15 16

15 17

16 17

16 18

17 18

17 19

18 19

291 723 425 111 676 788 134 203 848 909 373 402 178 841 530 467 315 670 866 870 289 960 621 560 607 345 122 419 792 518 203 987 341 548 415 785 850 434 975 328 318 851 768 186 1 39 261 605 211 209 719 65 99 697 267 514 773 336 150 199 641 604 945 951 959 264 31 549 95 339 769 457 624 739 717 754 691 332 81 730 195 895 490 792 798 758 977 643 481 265 660 822 327 413 668 619 772 295 579 277 808 874 270 303 379 679 389 925 406 101 266 972 759 56 689 915 806 942 445 425 858 338 853 686 149 137 96 83 651 850

0

OUT:

UDN

V1 V2 V3 V4 V5 V6 V7 V8 V9 APPLE BANANA PEACH GRAPE PEOPLE FATHER ZLC LZR LX WQ LJL

9999 291 723 425 111 676 9999 9999 9999 788 134 9999 203 848 909 9999 9999 9999 9999 373

291 9999 402 9999 178 841 530 467 9999 315 670 866 870 289 960 9999 9999 621 560 607

723 402 9999 345 122 419 9999 792 518 203 987 341 9999 548 9999 415 9999 785 9999 850

425 9999 345 9999 434 9999 975 328 318 851 768 9999 9999 1 186 9999 9999 9999 39 261

111 178 122 434 9999 605 211 209 719 65 99 697 267 514 773 336 150 199 9999 641

676 841 419 9999 605 9999 604 945 951 959 9999 264 31 549 95 339 769 9999 9999 457

9999 530 9999 975 211 604 9999 624 739 717 754 9999 9999 691 9999 332 9999 9999 9999 81

9999 467 792 328 209 945 624 9999 9999 730 195 895 9999 490 9999 792 9999 798 9999 758

9999 9999 518 318 719 951 739 9999 9999 977 643 481 265 660 822 327 413 9999 9999 668

788 315 203 851 65 959 717 730 977 9999 619 772 295 579 277 808 874 270 303 379

134 670 987 768 99 9999 754 195 643 619 9999 679 9999 9999 389 9999 9999 9999 9999 925

9999 866 341 9999 697 264 9999 895 481 772 679 9999 406 101 9999 266 972 759 56 9999

203 870 9999 9999 267 31 9999 9999 265 295 9999 406 9999 9999 9999 9999 689 915 9999 806

848 289 548 1 514 549 691 490 660 579 9999 101 9999 9999 942 445 425 9999 9999 858

909 960 9999 186 773 95 9999 9999 822 277 389 9999 9999 942 9999 338 853 9999 9999 9999

9999 9999 415 9999 336 339 332 792 327 808 9999 266 9999 445 338 9999 686 149 9999 9999

9999 9999 9999 9999 150 769 9999 9999 413 874 9999 972 689 425 853 686 9999 137 96 9999

9999 621 785 9999 199 9999 9999 798 9999 270 9999 759 915 9999 9999 149 137 9999 83 651

9999 560 9999 39 9999 9999 9999 9999 9999 303 9999 56 9999 9999 9999 9999 96 83 9999 850

373 607 850 261 641 457 81 758 668 379 925 9999 806 858 9999 9999 9999 651 850 9999

(V5,BANANA,99),(V4,FATHER,186),(V6,GRAPE,31),(V7,LJL,81),(WQ,LX,83),(V5,LZR,150),(WQ,PEACH,56),(V4,PEOPLE,1),(V5,V1,111),(V5,V2,178),(V5,V3,122),(WQ,V4,39),(APPLE,V5,65),(FATHER,V6,95),(V5,V7,211),(BANANA,V8,195),(GRAPE,V9,265),(LZR,WQ,96),(LX,ZLC,149)