**数据结构课程设计**

**——小组设计报告**

**组 号:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**指导教师: \_\_\_\_\_\_\_\_ \_冯向阳\_\_\_\_\_\_\_\_\_\_\_\_\_**

**组成员及贡献度**

|  |  |  |
| --- | --- | --- |
| 姓名 | 学号 | 贡献度 |
| 周立成 | 221310332 | 1 |
| 吕嘉乐 | 221310403 | 1 |
| 刘子睿 | 221310404 | 1 |
| 刘璇 | 221310402 | 1 |
| 汪淇 | 221310405 | 1 |

日期: 2023年6月20日至2022年7月07日

目录：

[1.课程设计目的 3](#_Toc138116878)

[2.任务完成情况 3](#_Toc138116879)

[3. 设计报告 4](#_Toc138116880)

[3.5Dijkstra算法的设计 4](#_Toc138116881)

[3.5.1 题目以及要求 4](#_Toc138116882)

[3.5.2概要设计 5](#_Toc138116883)

[3.5.3算法分析 5](#_Toc138116884)

[3.5.4使用说明 5](#_Toc138116885)

[3.5.5 测试结果以及分析 5](#_Toc138116886)

## 1.课程设计目的

1. 学习获取知识的方法；
2. 提高发现问题、分析问题和解决实际问题的能力；
3. 加强创新意识和创新精神；
4. 加强团队的分工与合作；
5. 掌握面向实际背景思考问题的方法。

## 2.任务完成情况

|  |  |  |
| --- | --- | --- |
| 任务序号 | 任务名称 | 完成情况 |
| 1 | 操作系统任务调度问题 | AC |
| 2 | 找出从指定结点出发且长度为*m*的所有简单路径 | AC |
| 3 | Prim算法的设计 | AC |
| 4 | Kruskal算法的设计 | AC |
| 5 | Dijkstra算法的设计 |  |
| 6 | 获取AOE网的关键路径 |  |
| 7 | 判断给定的二叉树是否是二叉排序树 |  |
| 8 | AVL树的判断 |  |
| 9 | 二叉树的重建 |  |
| 10 | 第 K 个最小的素数分数 |  |
| 11 | LRU缓存 |  |
| 12 | 设计推特 |  |
| 13 | 单线程 CPU |  |
| 14 | 我的日程安排表 I |  |
| 15 | 设计电影租借系统 | TLE |

## 3. 设计报告

### 3.5 Dijkstra算法的设计

#### 3.5.1 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Dijkstra算法，用以解决单源最短路径问题，并以文本形式输出从源点到其余各个顶点的路径以及路径长度。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Dijkstra。

提示：

（1）单源最短路径问题：已知有向带权图(简称有向网)G=(V，E)，找出从某个源点s∈V到V中其余各顶点的最短路径。限定权值均为正整数。

（2）目的： 设一有向图G=（V, E），已知各边的权值，以某指定点v0为源点，求从v0到图的其余各点的最短路径。限定各边上的权值大于或等于0。应按路径“长度”递增的次序，逐步产生最短路径。

（3）Dijkstra算法的基本步骤：设V0是起始源点，U = 已求得最短路径终点集合。V-U = 未确定最短路径的顶点的集合，初始时 U ={V0}。

1）“长度”最短的最短路径是边数为1的长度最小的路径。

2）下一条“长度”最短的路径：

① Vi ∈ V - U ，先求出V0 到Vi 中间只经 U 中结点的最短路径；

② 上述最短路径中长度最小者即为下一条长度最短的路径；

③ 将所求最短路径的终点加入U 中；

3）重复2）直到求出所有的最短路径。

（4）实现方法：

1）图用带权邻接矩阵存储ad[][]；

2）数组dist[]存放当前找到的从源点V0到每个终点的最短路径长度，其初态为图中直接路径权值；

3）数组pre[]表示从V0到各终点的最短路径上，此顶点的前一顶点的序号；若从V0到某终点无路径，则用-1作为其前一顶点的序号。

参考函数原形：

（1）//Dijkstra算法（成员函数）

*template<class TypeOfVer, class TypeOfEdge>*

*bool adjmatrix\_graph<TypeOfVer, TypeOfEdge>::Dijkstra( int u, TypeOfEdge \*dist, int \*pre);* // u：源点的位序

（2）辅助函数

//最短路径输出（用户函数）

*template<class TypeOfVer, class TypeOfEdge>*

*void searchPath(TypeOfVer \*ver, int \*prev, TypeOfEdge \*dist, int v, int u);* // ver：输入的顶点集  v：源点的位序  u：终点的位序

**输出说明:**

第一行：顶点集

     空行

第二行：图的邻接矩阵

     空行

第三行：dist数组的初值

第四行：pre数组的初值

     空行

第五行：dist数组的值

第六行：pre数组的值

   空行

第七行：源点到其余各顶点的最短路径及最短路径长度（输出格式参见测试数据）

#### 3.5.2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。

*void Dijkstra(int u)* 执行*dji*算法

*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。

#### 3.5.3算法分析

***算法说明：***

假设路网中每一个顶点都有标号*（dt , pt）*，*dt*是从出发点*s*到点*t*的最短路径长度；*pt*表示从出发点*s*到点*t*的最短路径中*t*前的一个顶点。基本过程如下：

1. 初始化：出发点*ds=0，ps*为空；其它所有点，*di=*无边标记，*pi*未定义。起点设为已标记，记*k*=*s*,其它顶点均未被标记。
2. 计算相连路径的距离：检验从所有已标记的点k到其他直接连接的未标记的点j的距离，并设置:*dj = min[ dj , dk + len (k , j) ]*,其中*len (k , j)*表示从*k*到*j*的路径长度。
3. 选取下一个点：从所有未标记的顶点中选取最小的点 *i*，点*i*被选作最短路径中的一点，并设为已标记。
4. 回溯*i*前一个顶点：从已标记的顶点集合中找到与*i*直接相连的顶点x，并记作*pi=x*。
5. 判断算法是否结束:若所有点被标记，则算法结束；否则记*k=i*，执行步骤2）。

赋权有向图的顶点数和边数分别设为n、m，算法总共进行（n-1）步，每一步选出具有最小dist[s,pi]值的顶点放在集合中，需要的时间。

以图4-1所示测试用例进行分析

赋权有向图如图5-1所示，顶点集为*V={V1,V2,V3,V4,V5,V6}*,选定V1作为源点S，则从*V1*出发到*V*中所有顶点的最短路径如图5-2所示（途径顶点由灰色标记，起点与终点顶点由绿色标记，最短路径由绿色标记）。

V1

V4

V5

V6

V3

V2

3

7

7

4

1

6

3

2

5

10

V1

V4

V5

V6

V3

V2

3

7

7

4

1

6

3

2

5

10

图5-1:赋权有向图

V1

V4

V5

V6

V3

V2

3

7

7

4

1

6

3

2

5

10

V1

V4

V5

V6

V3

V2

3

7

7

4

1

6

3

2

5

10

V1

V4

V5

V6

V3

V2

3

7

7

4

1

6

3

2

5

10

V1

V4

V5

V6

V3

V2

3

7

7

4

1

6

3

2

5

10

图5-2:顶点V2-V6的最短路径

下面对图5-2进行具体说明：

首先我们还是选择 *V1* 为原点*S*，那么在算法的开始*S={V1}*。之后我们计算除了 *V1*以外的其余与*V1*直接相连的顶点到 *V1*的*dist[V2,V1]~dist[V6,V1]*。

由图可知：*dist[V1,V2]=10；dist[V1,V6]=3*。

此时检验*V-S*是否为空集，发现*V-S={V2,V3,V4,V5,V6}*。故在集合V-S中查找相对于集合*S*的最短路径中距离最短的那个顶点*Vj*，此时显然是*V6*。故将*V6*放入集合*S*，说明从源点*V1*到*V6*的最短路径已经找到。

此时集合*S*更新为*S={V1,V6}*，后续在计算相对于*S*的最短路径时，就有两个顶点可以被经过。后续算法同上，不再进行赘述。

算法执行至*V-S={}=*，算法结束。

***算法时间复杂度:***根据分析该算法的时间复杂度为。

#### 3.5.4使用说明

**输入说明:**

第一行：图的类型

第二行：顶点数

第三行：顶点集

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：源点位序

程序名称为*test\_04\_graph2.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4-1、4-2

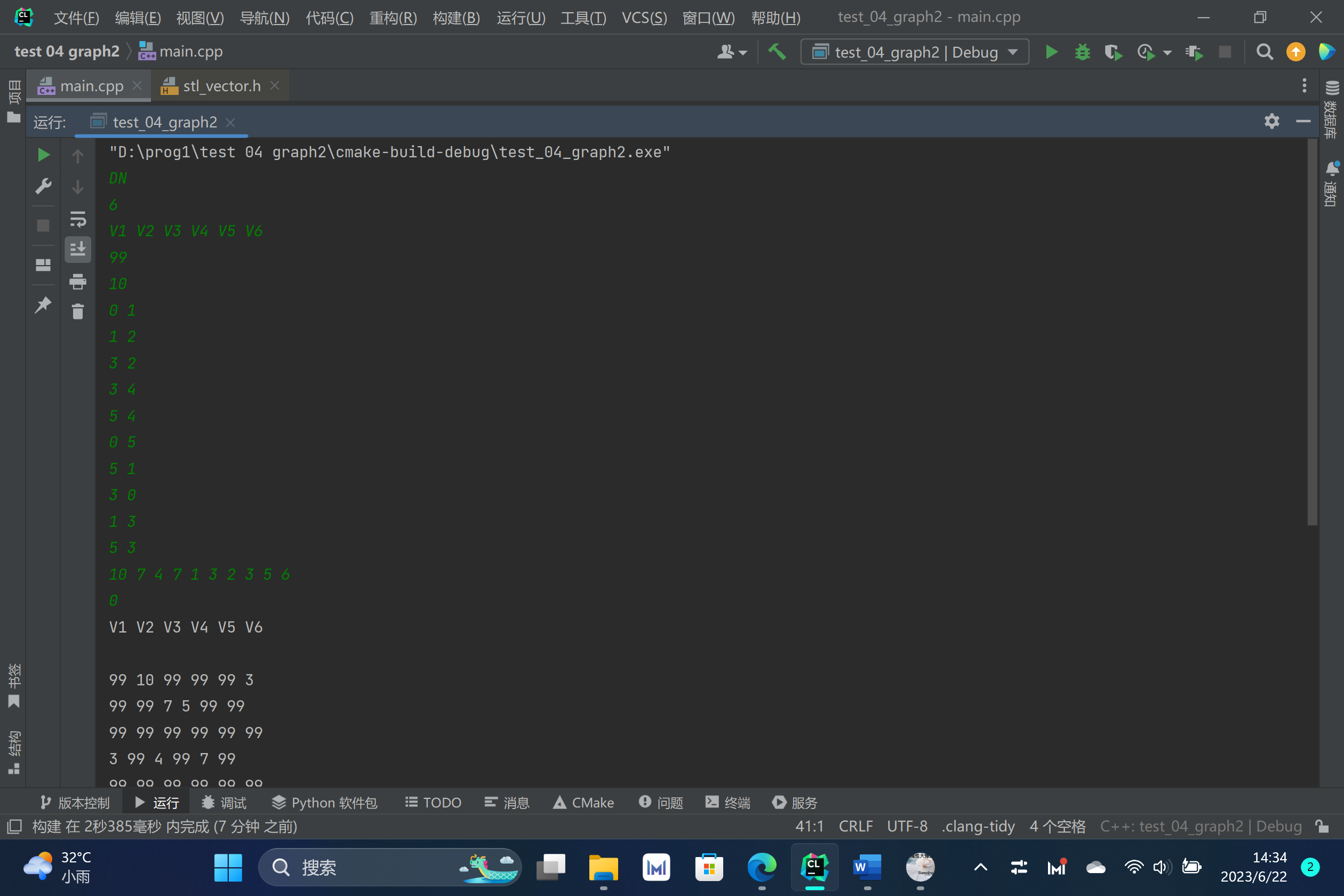


图4-1:运行截图（测试用例输入）

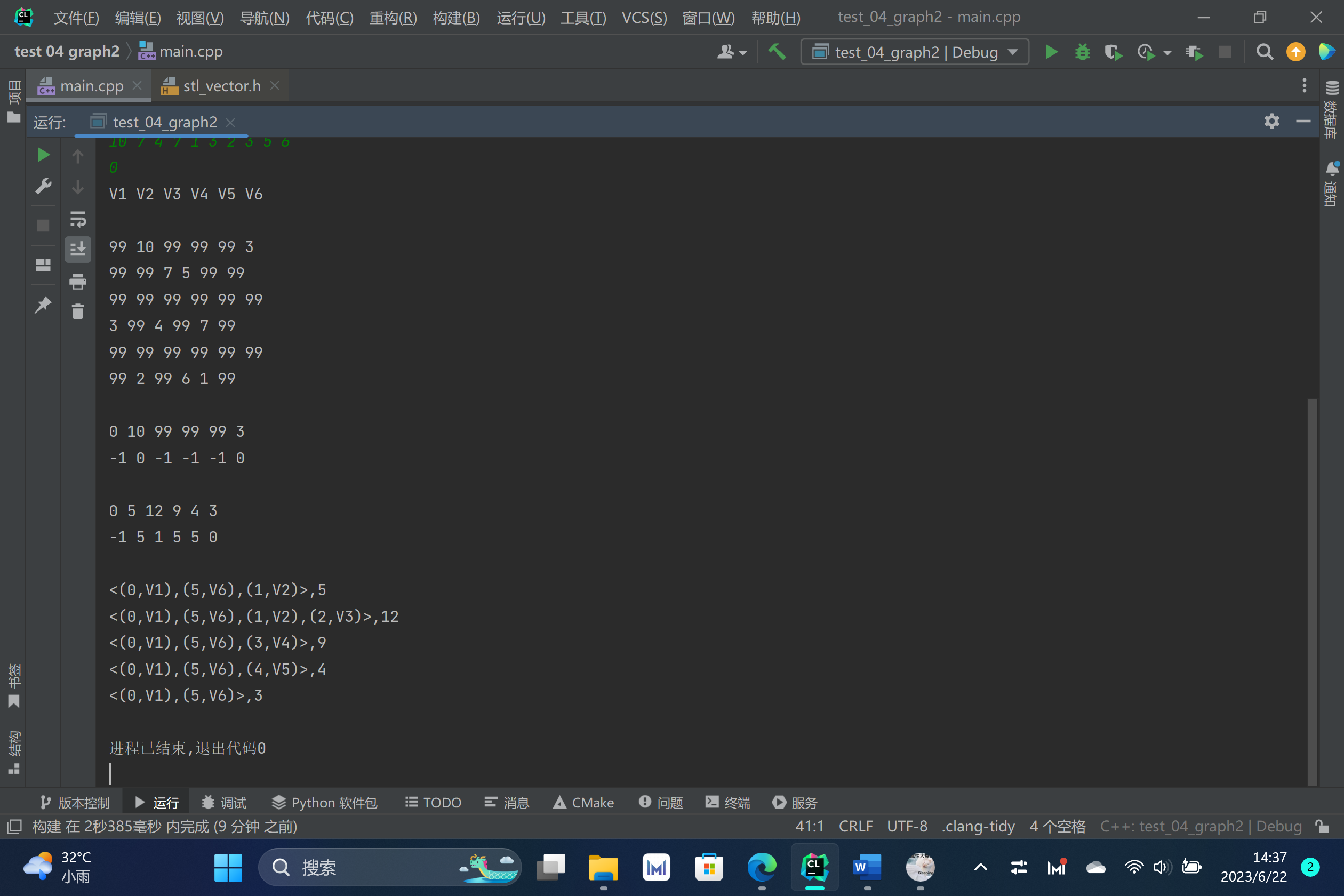


图4-2:运行截图（测试用例输出）

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.

#### 3.5.5 测试结果以及分析

IN1:普通测试用例

DN

7

V1 V2 V3 V4 V5 V6 V7

99

10

0 1

0 2

0 4

0 6

1 5

1 6

2 3

3 4

4 5

5 6

13 8 30 32 9 7 5 6 2 17

0

OUT:

V1 V2 V3 V4 V5 V6 V7

99 13 8 99 30 99 32

99 99 99 99 99 9 7

99 99 99 5 99 99 99

99 99 99 99 6 99 99

99 99 99 99 99 2 99

99 99 99 99 99 99 17

99 99 99 99 99 99 99

0 13 8 99 30 99 32

-1 0 0 -1 0 -1 0

0 13 8 13 19 21 20

-1 0 0 2 3 4 1

<(0,V1),(1,V2)>,13

<(0,V1),(2,V3)>,8

<(0,V1),(2,V3),(3,V4)>,13

<(0,V1),(2,V3),(3,V4),(4,V5)>,19

<(0,V1),(2,V3),(3,V4),(4,V5),(5,V6)>,21

<(0,V1),(1,V2),(6,V7)>,20

IN2:特殊用例,无向图,测试鲁棒性,不会进入dji算法模块

IN:

UDN

7

V1 V2 V3 V4 V5 V6 V7

99

10

0 1

0 2

0 4

0 6

1 5

1 6

2 3

3 4

4 5

5 6

13 8 30 32 9 7 5 6 2 17

0

OUT:

V1 V2 V3 V4 V5 V6 V7

99 13 8 99 30 99 32

99 99 99 99 99 9 7

99 99 99 5 99 99 99

99 99 99 99 6 99 99

99 99 99 99 99 2 99

99 99 99 99 99 99 17

99 99 99 99 99 99 99

0 13 8 99 30 99 32

-1 0 0 -1 0 -1 0

0 13 8 13 19 21 20

-1 0 0 2 3 4 1

IN 3:特殊用例,权值一样

IN:

DN

7

V1 V2 V3 V4 V5 V6 V7

99

10

0 1

0 2

0 4

0 6

1 5

1 6

2 3

3 4

4 5

5 6

1 1 1 1 1 1 1 1 1 1

0

V1 V2 V3 V4 V5 V6 V7

99 1 1 99 1 99 1

99 99 99 99 99 1 1

99 99 99 1 99 99 99

99 99 99 99 1 99 99

99 99 99 99 99 1 99

99 99 99 99 99 99 1

99 99 99 99 99 99 99

OUT:

0 1 1 99 1 99 1

-1 0 0 -1 0 -1 0

0 1 1 2 1 2 1

-1 0 0 2 0 1 0

<(0,V1),(1,V2)>,1

<(0,V1),(2,V3)>,1

<(0,V1),(2,V3),(3,V4)>,2

<(0,V1),(4,V5)>,1

<(0,V1),(1,V2),(5,V6)>,2

<(0,V1),(6,V7)>,1

IN4:特殊用例,连成一条线,且原点位序不为0



IN5:

DN

7

V1 V2 V3 V4 V5 V6 V7

99

6

0 1

1 2

2 3

3 4

4 5

5 6

1 2 3 4 5 6

2

OUT:

V1 V2 V3 V4 V5 V6 V7

99 1 99 99 99 99 99

99 99 2 99 99 99 99

99 99 99 3 99 99 99

99 99 99 99 4 99 99

99 99 99 99 99 5 99

99 99 99 99 99 99 6

99 99 99 99 99 99 99

99 99 0 3 99 99 99

-1 -1 -1 2 -1 -1 -1

99 99 0 3 7 12 18

-1 -1 -1 2 3 4 5

<(0,V1)>,99

<(1,V2)>,99

<(2,V3),(3,V4)>,3

<(2,V3),(3,V4),(4,V5)>,7

<(2,V3),(3,V4),(4,V5),(5,V6)>,12

<(2,V3),(3,V4),(4,V5),(5,V6),(6,V7)>,18

IN5:普通数据,但是是double 类型,测试成功

图示

描述已自动生成

IN:

DN

5

V1 V2 V3 V4 V5

9999

8

0 1

0 3

0 4

1 2

1 4

2 3

2 4

3 4

1.1 1.2 1.3 1.4 1.1 1.2 1.3 0.6

0

OUT:

V1 V2 V3 V4 V5

9999 1.1 9999 1.2 1.3

9999 9999 1.4 9999 1.1

9999 9999 9999 1.2 1.3

9999 9999 9999 9999 0.6

9999 9999 9999 9999 9999

0 1.1 9999 1.2 1.3

-1 0 -1 0 0

0 1.1 2.4 1.2 1.3

-1 0 1 0 0

<(0,V1),(1,V2)>,1.1

<(0,V1),(1,V2),(2,V3)>,2.4

<(0,V1),(3,V4)>,1.2

<(0,V1),(4,V5)>,1.3

进程已结束,退出代码0

IN6:大样例测试,数据集大小为1MB,23ms运行成功,与参考代码输出结果一致,如下图:

图示

描述已自动生成

IN:

DN

20

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 LX LJL WQ LZR ZLC HQB FXY LT ZJF

9999

152

0 1

0 2

0 3

0 4

0 5

0 6

0 7

0 10

0 11

0 15

0 16

0 17

0 18

0 19

1 2

1 3

1 4

1 5

1 6

1 7

1 8

1 9

1 10

1 11

1 12

1 13

1 14

1 15

1 16

1 17

1 18

1 19

2 3

2 4

2 5

2 7

2 8

2 9

2 10

2 15

2 19

3 4

3 5

3 6

3 7

3 8

3 9

3 10

3 11

3 12

3 15

3 16

3 17

3 18

3 19

4 5

4 6

4 7

4 8

4 9

4 10

4 11

4 12

4 13

4 15

4 19

5 6

5 7

5 8

5 9

5 10

5 11

5 12

5 13

5 14

5 15

5 16

5 17

5 18

5 19

6 7

6 8

6 9

6 10

6 11

6 12

6 13

6 14

6 15

6 19

7 8

7 9

7 10

7 11

7 12

7 13

7 14

7 15

7 16

7 19

8 9

8 10

8 11

8 12

8 13

8 14

8 15

8 19

9 10

9 11

9 12

9 13

9 14

9 15

9 16

9 17

9 19

10 11

10 12

10 13

10 14

10 15

10 16

10 17

10 18

10 19

11 12

11 14

11 18

11 19

12 13

12 14

12 16

12 18

12 19

13 14

13 15

13 17

13 18

13 19

14 15

14 18

14 19

15 16

15 17

15 19

16 17

16 18

16 19

17 18

17 19

18 19

38 71 66 24 1 75 59 19 25 17 88 63 75 63 80 79 4 77 48 100 28 82 7 41 95 79 45 87 84 78 80 15 73 28 96 98 89 87 44 100 73 19 61 48 47 72 79 27 88 41 75 27 27 98 12 37 89 29 33 81 2 53 29 46 35 50 34 74 23 69 24 50 72 77 59 26 78 6 36 51 54 19 42 23 90 66 60 12 67 31 6 6 23 70 29 19 23 51 66 37 23 71 80 27 28 31 18 45 66 1 59 9 26 91 41 73 13 58 80 74 62 94 16 15 82 9 67 50 97 36 88 5 37 28 1 4 98 12 34 85 20 98 36 17 81 63 75 91 81 37 34 11

0

OUT:

V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 LX LJL WQ LZR ZLC HQB FXY LT ZJF

9999 38 71 66 24 1 75 59 9999 9999 19 25 9999 9999 9999 17 88 63 75 63

9999 9999 80 79 4 77 48 100 28 82 7 41 95 79 45 87 84 78 80 15

9999 9999 9999 73 28 96 9999 98 89 87 44 9999 9999 9999 9999 100 9999 9999 9999 73

9999 9999 9999 9999 19 61 48 47 72 79 27 88 41 9999 9999 75 27 27 98 12

9999 9999 9999 9999 9999 37 89 29 33 81 2 53 29 46 9999 35 9999 9999 9999 50

9999 9999 9999 9999 9999 9999 34 74 23 69 24 50 72 77 59 26 78 6 36 51

9999 9999 9999 9999 9999 9999 9999 54 19 42 23 90 66 60 12 67 9999 9999 9999 31

9999 9999 9999 9999 9999 9999 9999 9999 6 6 23 70 29 19 23 51 66 9999 9999 37

9999 9999 9999 9999 9999 9999 9999 9999 9999 23 71 80 27 28 31 18 9999 9999 9999 45

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 66 1 59 9 26 91 41 73 9999 13

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 58 80 74 62 94 16 15 82 9

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 67 9999 50 9999 9999 9999 97 36

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 88 5 9999 37 9999 28 1

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 4 98 9999 12 34 85

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 20 9999 9999 98 36

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 17 81 9999 63

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 75 91 81

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 37 34

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 11

9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999 9999

0 38 71 66 24 1 75 59 9999 9999 19 25 9999 9999 9999 17 88 63 75 63

-1 0 0 0 0 0 0 0 -1 -1 0 0 -1 -1 -1 0 0 0 0 0

0 38 71 66 24 1 35 53 24 47 19 25 51 52 47 17 34 7 37 28

-1 0 0 0 0 0 5 4 5 8 0 0 8 8 6 0 15 5 5 10

<(0,V1),(1,V2)>,38

<(0,V1),(2,V3)>,71

<(0,V1),(3,V4)>,66

<(0,V1),(4,V5)>,24

<(0,V1),(5,V6)>,1

<(0,V1),(5,V6),(6,V7)>,35

<(0,V1),(4,V5),(7,V8)>,53

<(0,V1),(5,V6),(8,V9)>,24

<(0,V1),(5,V6),(8,V9),(9,V10)>,47

<(0,V1),(10,V11)>,19

<(0,V1),(11,LX)>,25

<(0,V1),(5,V6),(8,V9),(12,LJL)>,51

<(0,V1),(5,V6),(8,V9),(13,WQ)>,52

<(0,V1),(5,V6),(6,V7),(14,LZR)>,47

<(0,V1),(15,ZLC)>,17

<(0,V1),(15,ZLC),(16,HQB)>,34

<(0,V1),(5,V6),(17,FXY)>,7

<(0,V1),(5,V6),(18,LT)>,37

<(0,V1),(10,V11),(19,ZJF)>,28