数据结构综合应用

一．实现学生成绩管理系统

二．功能要求

(1)信息录入，包括记录的追加和插入　(2)信息删除　(3)信息修改　(4)信息排序和查询
(5)信息的保存和加载　(6)简单的说明和帮助。

2. 设计说明

顺序表和链表是线性表的两种基本形式，链表较灵活，插入和删除效率较高，但是链表空间利用率较低，适用于实现动态的线性表；顺序表结构简单，空间利用效率较高但是不易扩充，插入删除效率低。数组（包括结构体数组）的实质是一种线性表的顺序表示方式，它的优点是使用直观，便于快速、随机地存取线性表中的任一元素。

3. 功能说明

public void judge() throws IOExceptionadd() 判断是否有录入学生信息添加学生信息；public void menu()建立菜单；FileWriter fw=new FileWriter("D://桌面//student.txt",true)将学生信息录入指定的 txt 文件中 show() 显示学生信息表；delete()删除学生信息；look()查看学生信息；modif() 修改学生信息

4. 调试分析

系统中需要多次使用输出流，关于输出流的关闭我做了多次调试，在 add 方法的++n 的下一条语句，添加 fw.close（），发现放在这里的话，会造成这样的一个错误，由于输入流的关闭，就是会造成只能出入一次信息，而不能输入下一次，所以我修改了下，把 fw.close（）这句话放在这个方法 while（t==1）{ }成功解决，实现多次添加的目标。

5. 测试结果

```
*************学生信息管理系统*************
*****          1.录入学生信息                ******
*****          2.显示学生信息                ******
*****          3.修改学生信息                ******
*****          4.删除学生信息                ******
*****          5.查看学生信息                ******
*****          0.退出管理系统                ******
*****************************************
请选择(0~5):1
请输入学生学号:
331

请输入学生姓名:
304

请输入学生班级:
2011
请输入学生语文成绩:
532
请输入学生数学成绩:
321
请输入学生英语成绩:
511

是否继续添加(Y/N)
Y
请输入学生学号:
332

请输入学生姓名:
403

请输入学生班级:
2012
请输入学生语文成绩:
256
请输入学生数学成绩:
651
请输入学生英语成绩:
432

是否继续添加(Y/N)
```

```
432

是否继续添加(Y/N)
Y
请输入学生学号:
234

请输入学生姓名:
404

请输入学生班级:
2011
请输入学生语文成绩:
437
请输入学生数学成绩:
589
请输入学生英语成绩:
328

是否继续添加(Y/N)
N

是否返回系统主菜单(Y/N)Y
*************学生信息管理系统*************
*****          1.录入学生信息              ******
*****          2.显示学生信息              ******
*****          3.修改学生信息              ******
*****          4.删除学生信息              ******
*****          5.查看学生信息              ******
*****          0.退出管理系统              ******
*********************************************
请选择(0~5):1
请输入学生学号:
362

请输入学生姓名:
202

请输入学生班级:
2011
请输入学生语文成绩:
547
```

```
学号      姓名      班级      语文      数学      英语
331       304       2011      532       321       511
234       404       2011      437       589       328
362       202       2011      547       651       361
系统返回主菜单!
*************学生信息管理系统*************
*****          1.录入学生信息               ******
*****          2.显示学生信息               ******
*****          3.修改学生信息               ******
*****          4.删除学生信息               ******
*****          5.查看学生信息               ******
*****          0.退出管理系统               ******
*****************************************
请选择(0~5):3
请输入要修改的学号:
331
你要修改的学生信息如下:
学号      姓名      班级
331       304       2011
语文      数学      英语
532       321       511

你确定要修改(Y/N):
Y
*****************************************
*****          1.修改学号                *****
*****          2.修改班级                *****
*****          3.修改姓名                *****
*****************************************
请选择:
2
请输入新的班级:2012
数据已成功修改!
是否继续修改(Y/N)N


系统返回主菜单!
*************学生信息管理系统*************
*****          1.录入学生信息               ******
*****          2.显示学生信息               ******
*****          3.修改学生信息               ******
*****          4.删除学生信息               ******
```

```
系统返回主菜单!
**********学生信息管理系统**********
*****          1.录入学生信息                ******
*****          2.显示学生信息                ******
*****          3.修改学生信息                ******
*****          4.删除学生信息                ******
*****          5.查看学生信息                ******
*****          0.退出管理系统                ******
****************************************
请选择(0~5):2
本次操作共录入3位学生!
你录入的学生信息如下:

学号      姓名      班级      语文      数学      英语
331       304       2012      532       321       511
234       404       2011      437       589       328
362       202       2011      547       651       361
系统返回主菜单!
**********学生信息管理系统**********
*****          1.录入学生信息                ******
*****          2.显示学生信息                ******
*****          3.修改学生信息                ******
*****          4.删除学生信息                ******
*****          5.查看学生信息                ******
*****          0.退出管理系统                ******
****************************************
请选择(0~5):
```

student.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
        录入的学生信息列表

学号    姓名    班级    语文成绩      数学成绩    英语成绩
331     304     2011    532   532   511
332     403     2012    256   256   432
234     404     2011    437   437   328
        录入的学生信息列表

学号    姓名    班级    语文成绩      数学成绩    英语成绩
362     202     2011    547   547   361
```

6. 带注释的源程序

```
package sj;
import java.util.Scanner;
import java.lang.*;
import java.io.*;
class Students
{
    private static Students[] s=new Students[60]; //录入学生上限
    int n=0;
    private String name;
    private int num;
    private String classAge;
    private int chinese;
    private int  math;
    private int english;

    //判断是否有录入学生信息
    public void judge() throws IOException
    {
        int i;
        char ch;
        String str;
        Scanner In=new Scanner(System.in);
        if(n==0)
        {
            System.out.println("你还没有录入任何学生信息，是否录入(Y/N):");
            str=In.next();
            ch=str.charAt(0);
            while(ch!='Y'&&ch!='y'&&ch!='N'&&ch!='n')
            {
                System.out.println("输入有误，请重新输入:");
                str=In.next();
                ch=str.charAt(0);
            }
            if(ch=='Y'||ch=='y')
            {
                this.add();
            }
            if(ch=='N'||ch=='n')
            {
                this.menu();
            }
```

```
        }
    }

    //菜单
    public void menu() throws IOException  //将异常抛出，调用这个方法去处理异常,如果 main 方法也将异常抛出，则交给 Java 虚拟机来处理,下同.
    {
        int a;
        Scanner in=new Scanner(System.in);
        System.out.println("*************学生信息管理系统*************");
        System.out.println("*****        1.录入学生信息            ******");
        System.out.println("*****        2.显示学生信息            ******");
        System.out.println("*****        3.修改学生信息            ******");
        System.out.println("*****        4.删除学生信息            ******");
        System.out.println("*****        5.查看学生信息            ******");
        System.out.println("*****        0.退出管理系统            ******");
        System.out.println("******************************************");
        System.out.print("请选择(0~5):");
        a=in.nextInt();
        while(a<0||a>5)
        {
            System.out.print("输入无效，请重新输入:");
            a=in.nextInt();
        }
        switch(a)
        {
            case 1:this.add();break;
            case 2:this.show();break;
            case 3:this.modif();break;
            case 4:this.delete();break;
            case 5:this.look();break;
            case 0:System.out.println("成功退出系统！！！");System.exit(0);break;
        }
    }

    //录入学生信息
    public void add() throws IOException
    {
        String str,str1,str2;
        int i,num1,t=1;
        char ch,ch1;
        FileWriter fw=new FileWriter("D://桌面//student.txt",true);   //将学生信息录入指定的 txt 文件中
```

```
        fw.write("                录入的学生信息列表\r\n\r\n 学号     姓名     班级
语文成绩      数学成绩    英语成绩\r\n");
        Scanner In=new Scanner(System.in);
        while(t==1)
        {
            System.out.println("请输入学生学号:");
            num1=In.nextInt();

            //判断学号是否重复
            for(i=0;i<n;i++)
            {
                while(s[i].num==num1)
                {
                    System.out.println("已存在此学号，请重新输入");
                    System.out.print("请输入学号:");
                    num1=In.nextInt();
                }
            }
            s[n].num=num1;
            str2=String.valueOf(num1);
            fw.write(str2+"    ");
            System.out.println();
            System.out.println("请输入学生姓名:");
            s[n].name=In.next();
            fw.write(s[n].name+"      ");
            System.out.println();
            System.out.println("请输入学生班级:");
            s[n].classAge=In.next();
            fw.write(s[n].classAge+"  ");
            System.out.println("请输入学生语文成绩:");
            s[n].chinese=In.nextInt();
            fw.write(s[n].chinese+"  ");
            System.out.println("请输入学生数学成绩:");
            s[n].math=In.nextInt();
            fw.write(s[n].chinese+"  ");
            System.out.println("请输入学生英语成绩:");
            s[n].english=In.nextInt();
            fw.write(s[n].english+"\r\n");
            ++n;

            System.out.println();
            System.out.println("是否继续添加(Y/N)");
```

```
            str=In.next();
            ch=str.charAt(0);
            while(ch!='N'&&ch!='n'&&ch!='Y'&&ch!='y')
            {
                System.out.println("输入无效，请重新输入:");
                str=In.next();
                ch=str.charAt(0);
            }
            if(ch=='N'||ch=='n')
            {
                break;
            }
        }
        fw.close();
        System.out.println();
        System.out.print("是否返回系统主菜单(Y/N)");
        str1=In.next();
        ch1=str1.charAt(0);
        while(ch1!='Y'&&ch1!='y'&&ch1!='N'&&ch1!='n')
        {
            System.out.println("输入无效，请重新输入:");
            str1=In.next();
            ch1=str1.charAt(0);
        }
        if(ch1=='Y'||ch1=='y')
        {
            this.menu();
        }
        if(ch1=='N'||ch1=='n')
        {
            System.out.println("");
            System.out.println("你已退出系统！！！");
            System.exit(0);
        }
    }

    //显示学生信息
    public void show() throws IOException
    {
        int i;
        this.judge();
        System.out.println("本次操作共录入"+n+"位学生!");
        System.out.println("你录入的学生信息如下:");
```

```
        System.out.println();
        System.out.println("学号\t 姓名\t 班级\t 语文\t 数学\t 英语");
        for(i=0;i<n;i++)
        {
            System.out.println(s[i].num+"      "+s[i].name+"     "+s[i].classAge+"
"+s[i].chinese+"      "+s[i].math+"     "+s[i].english);
        }
        System.out.println("系统返回主菜单!");
        this.menu();
    }

    //删除学生信息
    public void delete() throws IOException
    {
        this.judge();
        int j=0,t=0,k=0,num1;
        char ch;
        String str;
        Scanner pin=new Scanner(System.in);
        System.out.println("请输入要删除的学号:");
        num1=pin.nextInt();
        for(j=0;j<n;j++)
        {
            if(s[j].num==num1)
            {
                k=1;
                t=j;
            }
        }
        if(k==0)
        {
            System.out.println("对不起! 你要删除的学号不存在! ");
            System.out.println("系统将返回主菜单! ");
            this.menu();
        }
        if(k==1)
        {
            System.out.println("你要删除的学生信息如下:");//打印管理员要删除的学生
信息
            System.out.println("学号\t 姓名\t 班级");//本功能暂时不备扩展性
            System.out.println(s[t].num+"                              "+s[t].name+"
"+s[t].classAge);
            System.out.println();
```

```
            System.out.println("你确定要删除(Y/N):");
            str=pin.next();
            ch=str.charAt(0);
            while(ch!='Y'&&ch!='y'&&ch!='N'&&ch!='n')
            {
                System.out.println("输入无效，请重新输入:");
                str=pin.next();
                ch=str.charAt(0);
            }
            if(ch=='N'||ch=='n')
            {
                System.out.println();
                System.out.println("系统返回主菜单！");
                this.menu();
            }
            if(ch=='Y'||ch=='y')
            {
                for(j=t;j<n-1;j++)
                {
                    s[j]=s[j+1];
                }
                n--;
                System.out.println("学生数据成功删除!");
                System.out.println("系统返回主菜单!");
                this.menu();
            }
        }
    }
    //查看学生信息
    public void look() throws IOException
    {
        FileReader fr=new FileReader("D://桌面//student.txt");  //查看 txt 中的学生
信息
        int a;
        while((a=fr.read())!=-1)
        {
            System.out.print((char)a);
        }
        fr.close();
        System.out.println("系统返回主菜单！");
        System.out.println();
        this.menu();
    }
```

```
    //修改学生信息
    public void modif() throws IOException
    {
        this.judge();
        int j=0,t=0,k=0,num2,num3,moi,c=1;
        char ch;
        String str,str1,str2;
        Scanner pin=new Scanner(System.in);
        System.out.println("请输入要修改的学号:");
        num2=pin.nextInt();
        for(j=0;j<n;j++)
        {
            if(s[j].num==num2)
            {
                k=1;
                t=j;
            }
        }
        if(k==0)
        {
            System.out.println("对不起! 你要修改的学号不存在! ");
            System.out.println("系统将返回主菜单! ");
            this.menu();
        }
        if(k==1)
        {
            //打印将要要删除的学生信息
            System.out.println("你要修改的学生信息如下:");
            System.out.println("学号\t 姓名\t 班级");
            System.out.println(s[t].num+"                                    "+s[t].name+"
"+s[t].classAge);
            System.out.println("语文\t 数学\t 英语");
            System.out.println(s[t].chinese+"                                    "+s[t].math+"
"+s[t].english);
            System.out.println();
            System.out.println("你确定要修改(Y/N):");
            str=pin.next();
            ch=str.charAt(0);
            while(ch!='Y'&&ch!='y'&&ch!='N'&&ch!='n')
            {
                System.out.println("输入无效, 请重新输入:");
                str=pin.next();
```

```
                ch=str.charAt(0);
            }
            if(ch=='N'||ch=='n')
            {
                System.out.println();
                System.out.println("系统返回主菜单！");
                this.menu();
            }
            while(c==1)
            {
                if(ch=='Y'||ch=='y')
                {

    System.out.println("****************************************");
                    System.out.println("*****                    1. 修 改 学 号
*****");
                    System.out.println("*****                    2. 修 改 班 级
*****");
                    System.out.println("*****                    3. 修 改 姓 名
*****");

    System.out.println("****************************************");
                    System.out.println("请选择:");
                    moi=pin.nextInt();
                    switch(moi)
                    {
                        case    1:System.out.print("  请  输  入  新  的  学
号:");num3=pin.nextInt();s[t].num=num3;break;
                        case    2:System.out.print("  请  输  入  新  的  班
级:");str1=pin.next();s[t].classAge=str1;break;
                        case    3:System.out.print("  请  输  入  新  的  姓
名:");str2=pin.next();s[t].name=str2;break;
                    }
                    System.out.println("数据已成功修改！");
                }
                System.out.print("是否继续修改(Y/N)");
                str=pin.next();
                ch=str.charAt(0);
                System.out.println();
                while(ch!='Y'&&ch!='y'&&ch!='N'&&ch!='n')
                {
                    System.out.print("输入无效，请重新输入:");
                    str=pin.next();
```

```
                    ch=str.charAt(0);
                }
                if(ch=='N'||ch=='n')
                {
                    break;
                }
            }
        }
        System.out.println();
        System.out.println("系统返回主菜单！");
        this.menu();
    }

    public static void main(String[] args) throws IOException
    {
        Students stu=new Students();
        for(int i=0;i<100;i++)
        {
            s[i]=new Students();
        }
        stu.menu();
    }
}
```

二．实现 Dijkstra 算法

1.功能要求

对有向网络用 Dijkstra 算法求出单源最短路径。

2.设计说明

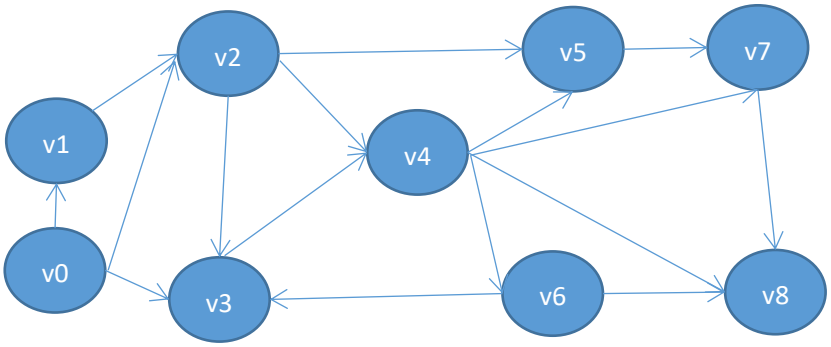
通过 Dijkstra 计算图 G 中的最短路径时，需要指定起点 vs(即从顶点 vs 开始计算)。此外，引进两个集合 S 和 U。S 的作用是记录已求出最短路径的顶点，而 U 则是记录还未求出最短路径的顶点(以及该顶点到起点 vs 的距离)。初始时，S 中只有起点 vs；U 中是除 vs 之外的顶点，并且 U 中顶点的路径是"起点 vs 到该顶点的路径"。然后，从 U 中找出路径最短的顶点，并将其加入到 S 中；接着，更新 U 中的顶点和顶点对应的路径。 然后，再从 U 中找出路径最短的顶点，并将其加入到 S 中；接着，更新 U 中的顶点和顶点对应的路径。 ... 重复该操作，直到遍历完所有顶点。

3.功能说明

`private void createGraph(int index)` 创建图

4.调试分析

Dijkstra 算法网课中 C 语言通过三个数组（最短路径数组，最短路径的 l 前驱顶点数组，final 数组）实现。
首先做出图，列出对应矩阵
图（格式原因，权值见矩阵）

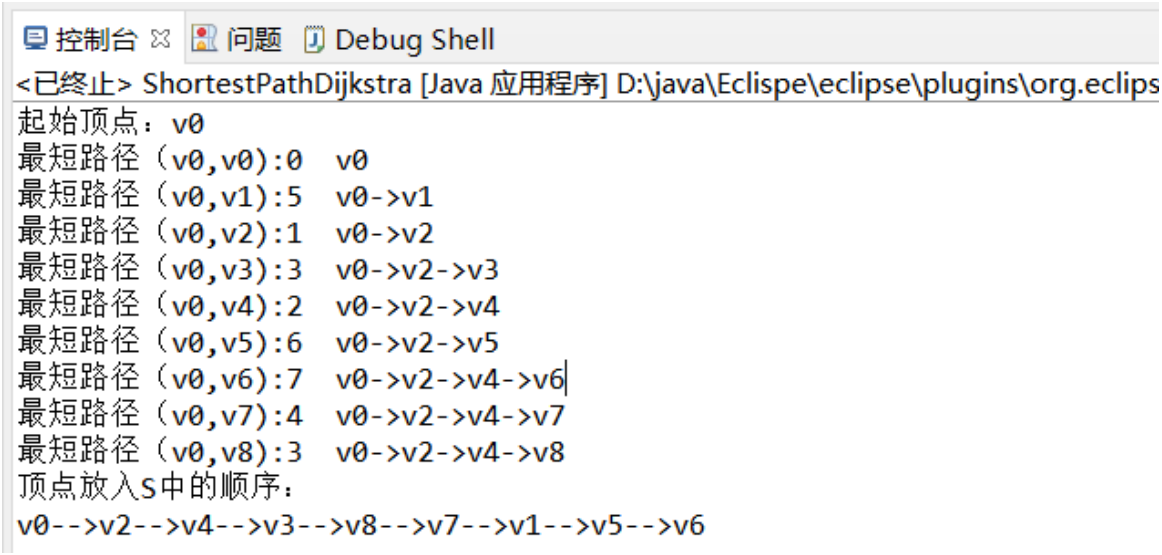


矩阵

	v0	v1	v2	v3	v4	v5	v6	v7	v8
v0	0	5	1	5	∞	∞	∞	∞	∞
v1	∞	0	3	∞	∞	∞	∞	∞	∞
v2	∞	∞	0	2	1	5	∞	∞	∞
v3	∞	∞	∞	0	7	∞	∞	∞	∞
v4	∞	∞	∞	∞	0	6	5	2	1
v5	∞	∞	∞	∞	∞	0	∞	1	∞
v6	∞	∞	4	∞	∞	∞	0	∞	3
v7	∞	∞	∞	∞	∞	∞	∞	0	6
v8	∞	∞	∞	∞	∞	∞	∞	∞	0

Java 中我采用以下四个步骤实现：
(1)　初始时，S 只包含起点 vs；U 包含除 vs 外的其他顶点，且 U 中顶点的距离为"起点 vs 到该顶点的距离"[例如,U 中顶点 v 的距离为(vs,v)的长度,然后 vs 和 v 不相邻,则 v 的距离为∞]。
(2)　从 U 中选出"距离最短的顶点 k"，并将顶点 k 加入到 S 中；同时，从 U 中移除顶点 k。
(3)　更新 U 中各个顶点到起点 vs 的距离。之所以更新 U 中顶点的距离，是由于上一步中确定了 k 是求出最短路径的顶点，从而可以利用 k 来更新其它顶点的距离；例如，(vs,v)的距离可能大于(vs,k)+(k,v)的距离。
(4)　重复步骤(2)和(3)，直到遍历完所有顶点

5.测试结果



6.带注释的源程序

```
package sj;
import java.util.*;
public class ShortestPathDijkstra {

    /** 邻接矩阵 */

    private int[][] matrix;

    /** 表示正无穷 */

    private int MAX_WEIGHT = Integer.MAX_VALUE;

    /** 顶点集合 */

    private String[] vertexes;


    /**

     * 创建图

     */
    private void createGraph(int index) {
        matrix = new int[index][index];
        vertexes = new String[index];


        int[] v0 = { 0, 5, 1, 5, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,
MAX_WEIGHT };
        int[] v1 = {MAX_WEIGHT,0, 3, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,
MAX_WEIGHT, MAX_WEIGHT };
        int[] v2 = { MAX_WEIGHT, MAX_WEIGHT, 0,2, 1, 5, MAX_WEIGHT, MAX_WEIGHT,
MAX_WEIGHT };
```

```
        int[] v3 = { MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, 0, 7, MAX_WEIGHT, MAX_WEIGHT,
MAX_WEIGHT, MAX_WEIGHT };
        int[] v4 = { MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,MAX_WEIGHT, 0, 6,5, 2, 1 };
        int[] v5 = { MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,
0, MAX_WEIGHT, 1, MAX_WEIGHT };
        int[] v6 = { MAX_WEIGHT, MAX_WEIGHT, 4, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,
0, MAX_WEIGHT,3};
        int[] v7 = { MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,
MAX_WEIGHT, MAX_WEIGHT, 0, 6 };
        int[] v8 = { MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT, MAX_WEIGHT,
MAX_WEIGHT, MAX_WEIGHT,MAX_WEIGHT, 0 };
        matrix[0] = v0;
        matrix[1] = v1;
        matrix[2] = v2;
        matrix[3] = v3;
        matrix[4] = v4;
        matrix[5] = v5;
        matrix[6] = v6;
        matrix[7] = v7;
        matrix[8] = v8;

        vertexes[0] = "v0";
        vertexes[1] = "v1";
        vertexes[2] = "v2";
        vertexes[3] = "v3";
        vertexes[4] = "v4";
        vertexes[5] = "v5";
        vertexes[6] = "v6";
        vertexes[7] = "v7";
        vertexes[8] = "v8";
    }

    /**
     * Dijkstra 最短路径。
     *
     * vs -- 起始顶点(start vertex) 即，统计图中"顶点 vs"到其它各个顶点的最短路径。
     */
    public void dijkstra(int vs) {
        // flag[i]=true 表示"顶点 vs"到"顶点 i"的最短路径已成功获取
        boolean[] flag = new boolean[vertexes.length];
```

```
        // U 则是记录还未求出最短路径的顶点(以及该顶点到起点 s 的距离)，与 flag 配合使用,flag[i] == true 表示 U 中 i 顶点已被移除
        int[] U = new int[vertexes.length];
        // 前驱顶点数组,即，prev[i]的值是"顶点 vs"到"顶点 i"的最短路径所经历的全部顶点中，位于"顶点 i"之前的那个顶点。
        int[] prev = new int[vertexes.length];
        // S 的作用是记录已求出最短路径的顶点
        String[] S = new String[vertexes.length];

        // 步骤一：初始时，S 中只有起点 vs；U 中是除 vs 之外的顶点，并且 U 中顶点的路径是"起点 vs 到该顶点的路径"。
        for (int i = 0; i < vertexes.length; i++) {
            flag[i] = false; // 顶点 i 的最短路径还没获取到。
            U[i] = matrix[vs][i]; // 顶点 i 与顶点 vs 的初始距离为"顶点 vs"到"顶点 i"的权。也就是邻接矩阵 vs 行的数据。

            prev[i] = 0; //顶点 i 的前驱顶点为 0
        }

        // 将 vs 从 U 中“移除”（U 与 flag 配合使用）
        flag[vs] = true;
        U[vs] = 0;
        // 将 vs 顶点加入 S
        S[0] = vertexes[vs];
        // 步骤一结束

        //步骤四：重复步骤二三，直到遍历完所有顶点。

        // 遍历 vertexes.length-1 次；每次找出一个顶点的最短路径。
```

```
        int k = 0;
        for (int i = 1; i < vertexes.length; i++) {
            // 步骤二：从 U 中找出路径最短的顶点，并将其加入到 S 中（如果 vs 顶点到 x 顶点
还有更短的路径的话，那么
            // 必然会有一个 y 顶点到 vs 顶点的路径比前者更短且没有加入 S 中
            // 所以，U 中路径最短顶点的路径就是该顶点的最短路径）
            // 即，在未获取最短路径的顶点中，找到离 vs 最近的顶点(k)。
            int min = MAX_WEIGHT;
            for (int j = 0; j < vertexes.length; j++) {
                if (flag[j] == false && U[j] < min) {
                    min = U[j];
                    k = j;
                }
            }

            //将 k 放入 S 中
            S[i] = vertexes[k];

            //步骤二结束


            //步骤三：更新 U 中的顶点和顶点对应的路径
            //标记"顶点 k"为已经获取到最短路径（更新 U 中的顶点，即将 k 顶点对应的 flag
标记为 true）
            flag[k] = true;

            //修正当前最短路径和前驱顶点（更新 U 中剩余顶点对应的路径）
            //即，当已经"顶点 k 的最短路径"之后，更新"未获取最短路径的顶点的最短路径和
前驱顶点"。
```

```
            for (int j = 0; j < vertexes.length; j++) {
                //以 k 顶点所在位置连线其他顶点，判断其他顶点经过最短路径顶点 k 到达 vs
顶点是否小于目前的最短路径，是，更新入 U，不是，不做处理
                int tmp = (matrix[k][j] == MAX_WEIGHT ? MAX_WEIGHT : (min +
matrix[k][j]));
                if (flag[j] == false && (tmp < U[j])) {
                    U[j] = tmp;
                    //更新 j 顶点的最短路径前驱顶点为 k
                    prev[j] = k;
                }
            }
            //步骤三结束
        }
        //步骤四结束

        // 打印 dijkstra 最短路径的结果
        System.out.println("起始顶点：" + vertexes[vs]);
        for (int i = 0; i < vertexes.length; i++) {
            System.out.print("最短路径（" + vertexes[vs] + "," + vertexes[i] + "):"
+ U[i] + "  ");
            List<String> path = new ArrayList<>();
            int j = i;
            while (true) {
                path.add(vertexes[j]);
                if (j == 0)
                    break;
                j = prev[j];
            }
            for (int x = path.size()-1; x >= 0; x--) {
                if (x == 0) {
                    System.out.println(path.get(x));
```

```
                } else {
                    System.out.print(path.get(x) + "->");
                }
            }

        }


        System.out.println("顶点放入 S 中的顺序：");
        for (int i = 0; i< vertexes.length; i++) {

            System.out.print(S[i]);

            if (i != vertexes.length-1)
                System.out.print("-->");
        }

    }

    public static void main(String[] args) {
        ShortestPathDijkstra dij = new ShortestPathDijkstra();
        dij.createGraph(9);
        dij.dijkstra(0);

    }
}
```