西安科技大学

《运筹学》课程设计实验报告

题目：基于Dijkstra算法的单源最短路径问题研究与实现

班级

姓名

学号

2024年 月 日

基于Dijkstra算法的单源最短路径问题研究与实现

1算法应用介绍

1.1 算法简介

Dijkstra算法是一种经典的最短路径算法，广泛应用于计算机网络、交通运输以及地图导航等领域。该算法通过迭代寻找未处理顶点中的最小值，逐步扩展已知最短路径的范围，最终得到源点到所有其他顶点的最短路径。

1.2 应用实例

在交通运输中，Dijkstra算法可以用于导航系统的路径规划，帮助用户找到从起点到目的地的最短路径。例如，在城市交通中，可以通过该算法优化公交线路，减少乘客的旅行时间。

2 方法理论

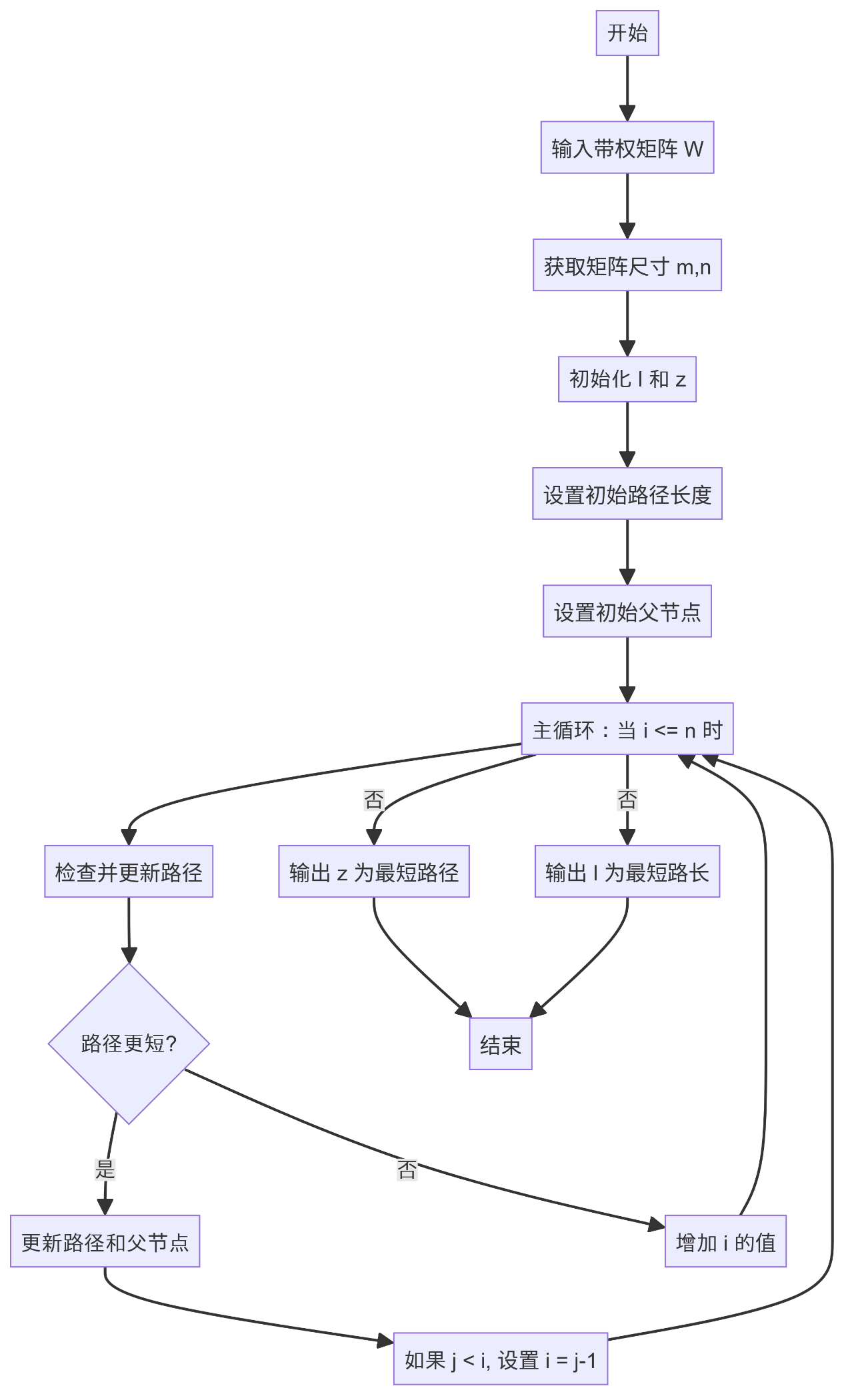
2.1 方法模型

* 模型来源：Dijkstra算法由荷兰计算机科学家Edger Dijkstra在1956年提出。
* 数学模型：该算法基于加权有向图，目标是找到从源点到其他顶点的最短路径。定义图𝐺=(𝑉,𝐸)，其中𝑉为顶点集，𝐸为边集，𝑊(𝑢,𝑣)为边权重（即从顶点𝑢到顶点𝑣的距离）。
* 模型局限性：Dijkstra算法适用于非负权重的图，对于存在负权重的图，该算法无法正确处理。

2.2 模型求解

* 求解思想：从源点开始，逐步找到距离源点最近的顶点，并更新其他顶点的最短路径。
* 核心问题：如何高效地选择当前距离最小的顶点。
* 难点：在于大规模图中，选择当前距离最小顶点的操作。

2.3 算法流程图



3 算法程序设计

3.1 参数和函数说明

* 全局参数：（带权矩阵）、（最短路径长度向量）、（父节点向量）
* 函数功能：
  + ：存储源点到顶点的路径长度。
  + ：存储顶点的父节点。

3.2 算法主程序

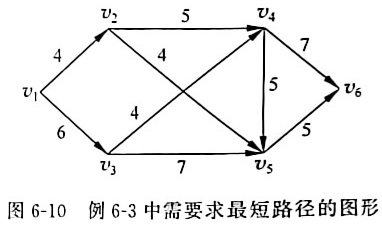
|  |
| --- |
| 主程序 |
| % Dijkstra 算法  % 输入带权矩阵 W  [m,n]=size(W);  % 初始化l(i),z(i)向量  for i=1:n  l(i)=W(1,i);  z(i)=0;  end  % 迭代更新最短路径  i=1;  while i<=n  for j=1:n  if l(i)>l(j)+W(j,i)  l(i)=l(j)+W(j,i);  z(i)=j-1;  if j<i  i=j-1;  end  end  end  i=i+1;  end  z=z+1;  disp('最短路径的父节点向量 z 为：');  disp(z);  disp('最短路径长度向量 l 为：');  disp(l); |

3.3 算法核心函数

**路径更新函数**：用于更新路径长度和父节点信息的双重循环

|  |
| --- |
| 路径更新函数 |
| while i<=n  for j=1:n  if l(i)>l(j)+W(j,i)  l(i)=l(j)+W(j,i);  z(i)=j-1;  if j<i  i=j-1;  end  end  end  i=i+1;  end |

4 算法测试

4.1 测试问题

用 Dijkstra算法求图中点到点的最短路径。

4.2 测试结果与分析

标号法

解:(1) 首先给 以 标号, 给其余所有点 标号，有

(2) 由于 边属于 ，且 为 标号，所以修改这两个点的标号

(3)比较所有 标号, 最小，所以令 并记录路径

(4) 为刚得到 标号的点,考查边( 的端点 ,,有

(5)比较所有 标号, 最小，所以令 并记录路径(

(6)考虑点有

(7)全部 标号中, 最小，令 记录路径

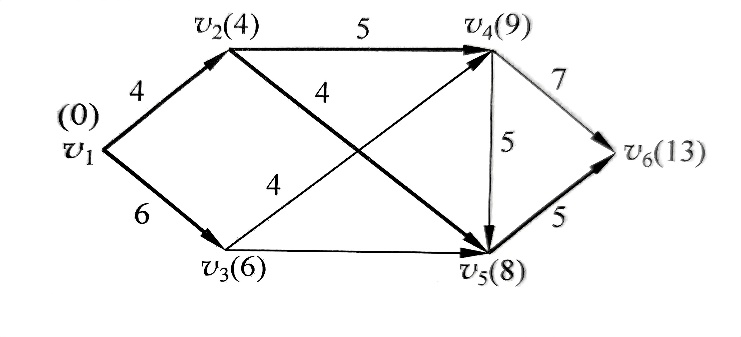
(8) 考查 ,有

(9)全部 T标号中, 最小，令 记录路径

(10) 考查,有

(11) 全部 标号中, 最小，令 记录路径 ，计算结束。

全部计算结果见图到之最短路径为路长 同时得到 点到其余各点的最短路径，如图中粗线所示。



使用matlab程序求解

我们将这些权重填入矩阵中，用无穷大表示不可达。

|  |
| --- |
| 0 4 6 ∞ ∞ ∞  ∞ 0 ∞ 5 4 ∞  ∞ ∞ 0 4 7 ∞  ∞ ∞ ∞ 0 5 7  ∞ ∞ ∞ ∞ 0 5  ∞ ∞ ∞ ∞ ∞ 0] |

用 MATLAB 代码表示：

|  |
| --- |
| % 使用 inf 表示不可达的距离  inf = 1e6; % 设定一个较大的值来表示无穷大  % 构建权重矩阵  W=[0 4 6 inf inf inf  inf 0 inf 5 4 inf  inf inf 0 4 7 inf  inf inf inf 0 5 7  inf inf inf inf 0 5  inf inf inf inf inf 0]; |

点击运行

运行结果如下：

|  |
| --- |
| z =  1 1 1 2 2 5  l =  0 4 6 9 8 13 |

最短路径为,即，最短路径长为13.

与上述标号法解相同。

5 总结和展望

适用性

Dijkstra 算法适用于具有以下特点的问题：

1. 正权重：算法假设图中所有边的权重均为非负数。
2. 单源最短路径：适用于需要计算从单一源点到其他所有顶点的最短路径问题。
3. 稠密图和稀疏图：可以处理稠密图和稀疏图，但在稀疏图中效率更高。

存在问题

尽管算法能够正确计算最短路径，但在实现过程中存在以下问题：

1. 未处理的边界情况：如图中某些顶点不可达时的处理不够完善。
2. 效率问题：嵌套循环的时间复杂度为，在处理大型图时可能效率较低。
3. 代码健壮性：对于权重矩阵中出现无穷大或其他异常情况时，缺乏异常处理机制。

改进建议

1. 增加异常处理：在初始化和迭代过程中，增加对无穷大或负权重边的处理。

|  |
| --- |
| if W(i, j) < 0  error('图中包含负权重边，Dijkstra算法不适用');  end |

1. 代码模块化：将初始化、迭代计算和结果输出部分模块化，提高代码可读性和可维护性。

|  |
| --- |
| 模块化后的代码（beta） |
| function [l, z] = initialize(W)  [m, n] = size(W);  l = inf(1, n);  z = zeros(1, n);  l(1) = 0;  for i = 1:n  l(i) = W(1, i);  z(i) = 0;  end  end  function [l, z] = iterate(W, l, z)  [m, n] = size(W);  for i = 1:n  for j = 1:n  if l(i) > l(j) + W(j, i)  l(i) = l(j) + W(j, i);  z(i) = j - 1;  if j < i  i = j - 1;  end  end  end  i = i + 1;  end  end  % 主函数  function [l, z] = dijkstra(W)  [l, z] = initialize(W);  [l, z] = iterate(W, l, z);  z = z + 1;  end |

参考文献：

[1] 胡运权，郭耀煌.运筹学教程[M].4版.北京：清华大学出版社，2012.

[2] 卓金武，段蕴珊，姜晓慧.MATLAB运筹学.北京：清华大学出版社，2022

附录：

其他程序源代码

|  |
| --- |
| Dijkstra.m |
| % Dijkstra 算法  % 输入带权矩阵 W  [m,n]=size(W);  %赋初值  %l(v)——顶点 v 的标号，表示从顶点 u0到 v 的一条路的权值；  %z(v)——顶点 v 的父节点标号，用以确定短路的路线。  n=size(W,1);  for i=1:n  l(i)=W(1,i);  z(i)=0;  end  %初始化l(i),z(i)向量  %l(i) 存储的是从源点（假设为顶点 1）到顶点 i 的初始路径长度。  %z(i) 存储的是从源点到顶点 i 的初始父节点，初始值为 0。  i=1;  while i<=n  for j=1:n  if l(i)>l(j)+W(j,i)  l(i)=l(j)+W(j,i);  z(i)=j-1;  if j<i  i=j-1;  end  end  end  i=i+1;  end  z=z+1;  z  l  %输出向量z为最短路径  %输出向量l为最短路长 |

|  |
| --- |
| test.m |
| W=[0 4 6 inf inf inf  inf 0 inf 5 4 inf  inf inf 0 4 7 inf  inf inf inf 0 5 7  inf inf inf inf 0 5  inf inf inf inf inf 0];  inf=1e6; |