数据结构中最短路径算法的实现

曹 阳

(陕西理工学院 计算机系,陕西 汉中 723000)

摘要:最短路径算法种类繁多,比较有名的算法包括:Dijkstra算法、Ford算法、Floyd算法、Moore算法、A* 算法、K值算法,而即使同一种算法也有多种不同的实现方式。本文介绍了求最短路径的 Dijkstra算法的设计思路及 Visual C++语言编程实现。实验表明:该算法能高效地求出一个顶点到其它各顶点的所有最短路径。

关键词:最短路径;Dijkstra算法;Visual C++程序语言

中图分类号:TP311.12 **文献标识码**:A **文章编号**:1671-5365(2007)06-0082-03

0 引言

最短路径问题是图论研究中的一个重要课题,它广泛应用于交通、网络寻优等领域。最短路径问题要解决的就是求加权图 G=《V, E, W》中两给定顶点之间的最短路径。求解图的最短路径在许多应用领域里有实际意义。例如:对于图 1表示的交通图,图中顶点表示城市,边线表示各城市之间的交通路线。边上的权表示交通路线的长度或花费代价等。对于这样的交通网络常常会关心如下问题:(1)若从 A城市到 B城市是否有公路可通?(2)在从 A城市到 B城市有若干条公路可通的情况下,哪一条公路的路程最短或花费的代价最低?这就是《数据结构》课程中求带权图的结点间最短路径的问题,此时路径的长度不是路径上的边的数目,而是路径上的边所带权值的总和。求最短路径的一个著名算法是 Dijkstra算法[1],它可以求出图中从一个顶点到其它各顶点的最短路径的长度及一条最短路径。本文就是用 Visual C++语言程序编程实现 Dijkstra算法。

1 编程思路

1.1 相关概念

定义 1 给定简单加权图 G=<V, E, W>,设 $v_0, v_1, \cdots v_m$ (E,其中 v_{i-1}、v_i是 e_i的结点,序列 $v_0, v_1, \cdots, v_m$ 称为连接 v_o到 v_m的路,记为 $v_0, v_1, \cdots, v_m$。路中边的数目称为该路的秩。$\omega_{01}+\omega_{12}+\cdots+\omega_{n-1})$n称为该路的长度。所有连接 v_0到 v_m的路中长度最小的路称为 v_0到 v_m的最短路径。

通常的无向图和有向图可以看成是加权图的特例。

定义 2 给定简单加权图 G=<V, E, W>, V={$v_0, v_1, \cdots, v_{n-1}$},称 A=($a_{ij}$)为图 G的邻接矩阵,其中:

$$a_{ij}=\begin{cases}\omega_{ij}, & \text{若 } v_i \text{和 } v_j \text{之间有边相连}\\ \infty, & \text{若 } v_i \text{和 } v_j \text{之间无边相连}\\ 0, & \text{若 } i=j\end{cases}$$

ω_{ij}表示 v_i和 v_j之间边的权值。

Dijksra算法的基本思想是按路径长度递增的次序产生最短路径,可以由下式给出:

$$D[i]=\min\{D[i], D[i]+\omega_{ji}\}$$

1.2 图的存储

编程之前,首先要考虑图的计算机存储问题。在此用邻接矩阵 cost[max][AX]来表示带权有向图,若从 V_i到 V_j有弧,则 cost[i][j]值为弧(V_i, V_j)上的权值,否则为∞。从图的邻接矩阵 cost出发,求图中从 V_i到 V_j的最短路径长度和结点序列。图 1的 cost矩阵如图 2所示。

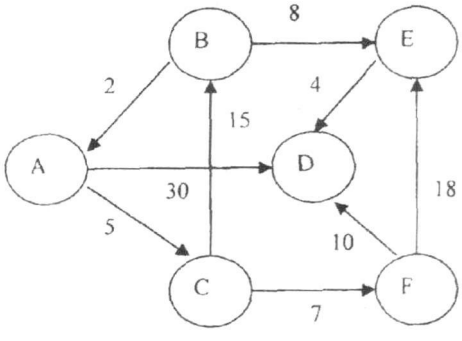

图 1 一个带权的有向图

收稿日期:2006-12-15

作者简介:曹阳(1978-),女,四川崇州人,讲师,电子科技大学在职硕士研究生,主要从事计算机应用与数据结构和算法研究。

$$cost=\begin{pmatrix} 0 & \infty & 5 & 30 & \infty & \infty \\ 2 & 0 & \infty & \infty & 8 & \infty \\ \infty & 15 & 0 & \infty & \infty & / \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 4 & 0 & \infty \\ \infty & \infty & \infty & 10 & 18 & \infty \end{pmatrix}$$

图 2　矩阵 cost

1.3　Dijkstra算法描述

1.3.1　Dijkstra算法描述如下:

(1) 设置当前点集初态,列出 v_0 到各点的距离。

(2) 从 v_0 到当前点集中找出距离最短的点 v_j 及相应的路径,并将其余各点的已有路径与经过该路径的新路径比较,若新路径短,则更新为新路径。

(3) 从当前点集中除去 v_j,并重复过程(2)进行类似的处理,直至当前点集为空,即确定了从 v_0 到所有各点中的最短距离。从上述算法中可以知道,从 v_0 到其他各点的最短路径是逐一求得的,对某一个顶点来说,其最短路径是从初始路径开始,经过若干次更新后求得的。

1.3.2　算法的实现要点如下:

(1)设置数组 dist[n],每个分量 dist[i] 存放 v_0 到 v_i 的当前最短路径长度,其初始值为:如果从 v_0 到 v_i 有一条路径,则为权值,否则为无穷大。

(2)设置集合型数组 path[n],每个分量 path[i]存放与 dist[i]对应的最短路径的点集,其初始值为:如果从 v_0 到 v_i 有一条路径,则为〈v_0, v_i〉,否则为空集。

(3)设置集合型变量 S,存放当前已确定了最短路径的那些点的点集,其初始值为 [v_0]。显然,在确定下一个最短路径的点时,应除去 S中的那些点。

(4)此外,还设置 cost为带权有向图的邻接矩阵,存放权值,V为指定的起始顶点

1.3.3　算法的处理过程如下:

(1) 设置数组 dist[n], path[n],及集合型变量 S初态。

(2)确定一个最短路径的顶点及相应的路径,将这个顶点划进 S,并以这条路径与其他各点的已有路径作比较,若通过这条路径可使达到某一点的距离减少,则更新该点的已有路径。

(3) 重复执行过程(2),直至 S中包含 n个顶点。

2　程序清单

```
#include<iostream.h>
#include<iomanip.h>
#include<stdlib.h>
#include"graph.cpp"
//网 G从下标 V0到其他顶点的最短距离 dist和最短路径下标 path
void PshortPath(AdjMatrix &G, int v0, int dist[], int path[])
{int n=G.NumV();
 int * s=new int[n];
 int mindis, i, j, u;
 for(i=0; i<n; i++)
  {dist[i]=G.GetWeight(v0, i);
  s[i]=0;
  if(i!=v0&&dist[i]<MaxValue) path[i]=v0;
  else path[i]=-1;      }s[v0]=1; //标记顶点 v0已从集合 T加入到集合 S中
 //在当前还未找到最短路径的顶点集中选取具有最短距离的顶点 u
 for(i=1; i<n; i++)
  {mindis=MaxValue;
  for(j=0; j<n; j++)
  if(s[j]==0&&dist[j]<mindis)
   {u=j;mindis=dist[j];}
  //当已不再存在路径时算法结束;此语句对非连通图是必需的
  if(mindis==MaxValue) return;
  s[u]=1; //标记顶点 u已从集合 T加入到集合 S中
  //修改从 v0到其他顶点的最短距离和最短路径
 for(j=0; j<n; j++)
 if(s[j]==0&&G.GetWeight(u, j)<MaxValue&&dist[u]+G.GetWeight(u, j)<dist[j])
 {//顶点 v0经顶点 u到其他顶点的最短距离和最短路径
 dist[j]=dist[u]+G.GetWeight(u, j); path[j]=u;}}}
void main()
{ cout<<"PshortPl.cpp运行结果:\n";
 int n=6, k1=1, k2=1; //n为顶点数, k1为有(无)向标志, k2为有(无)权标志
 AdjMatrix g(n, k2); //定义对象
 g.CreateMatrix(n, k1, k2)
 cout<<"\n输出邻接矩阵相应图的每个顶点:\n";
 g.Creatgraph(n, k2);
 int m=g.NumV();
 int * dist=new int[m];
 int * path=new int[m];
 int v0=0;
 PshortPath(g, v0, dist, path);
 Cout<<"从顶点"<<g.GetValue(v0)<<"到其他各顶点的最短距离为:\n";
 For(int i=0; i<m; i++)
```

Cout<<"到顶点"<<g.GetValue(v0)<<"的最短距离为:"<<dist[I]<<end;

Cout<<"从顶点"<<g.GetValue(v0)<<"到其他各顶点的最短路径的前一顶点为:\n";

For(I=0; I<m; I++)

If(path[I]! =-1)

Cout<<"到顶点"<<g.GetValue(i)<<"的前一顶点为:"<<g.GetValue(path[I])<<end;

Cin.get(); cin.get();}

3 运行结果

输入图的总边数:9

输入 9条有向有权边的起点和终点序号及权值!

0 2 5 0 3 30 1 0 2 1 4 8 2 1 15

2 5 7 4 3 4 5 3 10 5 4 18

创建后的邻接矩阵:

$$\begin{pmatrix} 0 & \infty & 5 & 30 & \infty & \infty \\ 2 & 0 & \infty & \infty & 8 & \infty \\ \infty & 15 & 0 & \infty & \infty & 7 \\ \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & \infty & 4 & 0 & \infty \\ \infty & \infty & \infty & 10 & 18 & 0 \end{pmatrix}$$

输出邻接矩阵相应图的每个顶点:

A(0, 2, 5) B(0, 3, 30) C(1, 0, 2) D(1, 4, 8) E(2, 1, 15) F(2, 5, 7)

从顶点 A到其他各顶点的最短距离为:

到顶点 A的最短距离为:0

到顶点 B的最短距离为:20

到顶点 C的最短距离为:5

到顶点 D的最短距离为:22

到顶点 E的最短距离为:28

到顶点 F的最短距离为:12

从顶点 A到其他各顶点的最短路径前一顶点为:

到顶点 B的前一顶点为:C

到顶点 C的前一顶点为:A

到顶点 D的前一顶点为:F

到顶点 E的前一顶点为:B

到顶点 F的前一顶点为:C

4 结束语

综上所述,这个程序的编制和调试是长期理论教学和编程实践的结晶,希望对参与工程决策及学习《数据结构》编程的人们有所帮助。

参考文献:

〔1〕严蔚敏,吴伟民.数据结构〔M〕.北京:清华大学出版社,2002.

〔2〕孙强,沈建华,顾君忠.Dijkstra的一种改进算法〔J〕.计算机工程与应用 2002,38(3):99-101.

〔3〕徐凤生.最短路径的求解算法〔J〕.计算机应用,2004,24(5):88-89.

〔4〕宋丽敏.最短路径的编程实现〔J〕.华北航天工业学院学报,2001,11(4):28-29.

〔5〕侯识忠.数据结构算法〔M〕.北京:中国水利水电出版社,2005.

Realization of the Shortest Path's Algorithm in Data Structure

CAO Yang

(Department of Computer Science, Shaanxi University of Technology, Hanzhong 723000, China)

Abstract: The shortest path's algorithm is in a great variety of which the famous ones including Dijkstra Algorithm, Ford Algorithm, Floyd Algorithm, Moore Algorithm, A* Algorithm and Value of K Algorithm. Even in the same algorithm, the realizing ways are different. This article introduces the design of the shortest path of Dijkstra and the realization of programming the language of Visual C++. All the shortest paths from one node to all other nodes can be derived quickly by using the algorithm.

Key words: Shortest Path; Dijkstra Algorithm; Visual C++ Language Programming