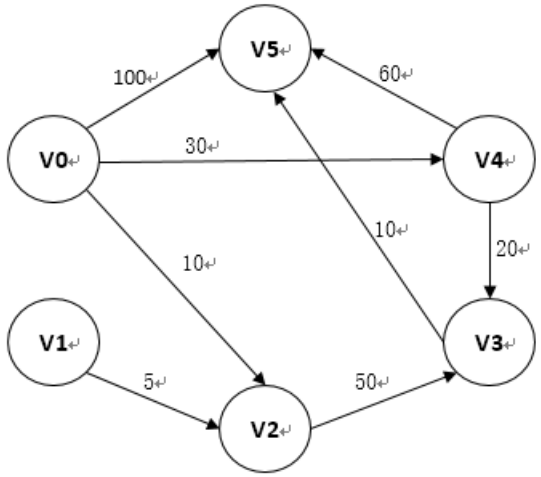
python实现Dijkstra算法的最短路径问题

迪杰斯特拉（Dijkstra）算法主要是针对没有负值的有向图，求解其中的单一起点到其他顶点的最短路径算法。

1 算法原理

迪杰斯特拉（Dijkstra）算法是一个按照路径长度递增的次序产生的最短路径算法。下图为带权值的有向图，作为程序中的实验数据。



其中，带权值的有向图采用邻接矩阵graph来进行存储，在计算中就是采用n*n的二维数组来进行存储，v0-v5表示数组的索引编号0-5，二维数组的值表示节点之间的权值，若两个节点不能通行，比如，v0->v1不能通行，那么graph[0,1]=+∞（采用计算机中最大正整数来进行表示）。那如何求解从v0每个v节点的最短路径长度呢?

$$\begin{bmatrix} graph & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \\ v_0 & \infty & \infty & 10 & \infty & 30 & 100 \\ v_1 & \infty & \infty & 5 & \infty & \infty & \infty \\ v_2 & \infty & \infty & \infty & 50 & \infty & \infty \\ v_3 & \infty & \infty & \infty & \infty & \infty & 10 \\ v_4 & \infty & \infty & \infty & 20 & \infty & 60 \\ v_5 & \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

首先，引进一个辅助数组cost，它的每个值cost[i]表示当前所找到的从起始点v0到终点vi的最短路径的权值（长度花费），该数组的初态为：若从v0到vi有弧，则cost[i]为弧上的权值，否则置cost[i]为+∞。

显然，长度为：cost[j]=Min_i(graph[0,i] | v_i in V)*的路径就是从v0出发的长度最短的一条最短路径。此路径为(v_0,v_j)，那么下次长度次短的路径必定是弧(v_0,v_i)上的权值cost[i](v_i in V)，或者是cost[k](v_k in S)和弧(v_k,v_i)的权值之和。其中V：待求解最短路径的节点集合；S：已求解最短路径的节点集合。*

2 算法流程

根据上面的算法原理分析，下面描述算法的实现流程。

初始化：初始化辅助数组cost，从v0出发到图上其余节点v的初始权值为：cost[i]=graph[0,i] | v_i in V ；初始化待求节点S集合，它的初始状态为始点，V集合，全部节点-始节点。

选择节点v_j ，使得cost[j]=Min（cost[i] | v_i in V -S），v_j 就是当前求的一条从v0出发的最短路径的终点，修改S集合，使得S=S + V_j ，修改集合V = V – V_j。

修改从v0出发到节点V-S上任一顶点 v_k 可达的最短路径，若cost[j]+graph[j,k]<cost[k] ，则修改cost[k]为：cost[k]=cost[j]+graph[j,k] 。

重复操作2，3步骤，直到求解集合V中的所有节点为止。

其中最短路径的存储采用一个path整数数组，path[i]的值记录vi的前一个节点的索引，通过path一直追溯到起点，就可以找到从vi到起始节点的最短路径。比如起始节点索引为0，若path[3]=4, path[4]=0；那么节点v2的最短路径为，v0->v4->v3。

3 算法实现

采用python语言对第2节中的算法流程进行实现，关键代码如下。

3.1 最短路径代码

```
#!/bin/python
# -*- coding:utf-8 -*-

def dijkstra(graph, startIndex, path, cost, max):
"""
求解各节点最短路径，获取path，和cost数组，
path[i] 表示vi节点的前继节点索引，一直追溯到起点。
cost[i] 表示vi节点的花费
"""
lenth = len(graph)
v = [0] * lenth
# 初始化 path，cost，V
for i in range(lenth):
if i == startIndex:
v[startIndex] = 1
else:
cost[i] = graph[startIndex][i] path[i] = (startIndex if (cost[i] < max) else -1)
# print v, cost, path
for i in range(1, lenth):
minCost = max
curNode = -1
for w in range(lenth):
if v[w] == 0 and cost[w] < minCost:
minCost = cost[w] curNode = w
# for 获取最小权值的节点
if curNode == -1: break
# 剩下都是不可通行的节点，跳出循环
v[curNode] = 1
for w in range(lenth):
if v[w] == 0 and (graph[curNode][w] + cost[curNode] < cost[w]):
cost[w] = graph[curNode][w] + cost[curNode] # 更新权值
path[w] = curNode # 更新路径
# for 更新其他节点的权值（距离）和路径
return path

if __name__ == '__main__':
max = 2147483647
graph = [
[max, max, 10, max, 30, 100],
[max, max, 5, max, max, max],
[max, max, max, 50, max, max],
[max, max, max, max, max, 10],
[max, max, max, 20, max, 60],
[max, max, max, max, max, max],
] path = [0] * 6
cost = [0] * 6
print dijkstra(graph, 0, path, cost, max)
```

4 运行结果

```
[0, -1, 0, 4, 0, 3]
```