**期末报告（**选题2**）**

**张峰瑞 M20W0228**

**最短路径的定义：**从某顶点出发，沿图的边到达另一顶点所经过的路径中，各边上权值之和最小的一条路径叫做[最短路径](https://baike.baidu.com/item/%E6%9C%80%E7%9F%AD%E8%B7%AF%E5%BE%84/6334920)。解决最短路的问题有以下算法，Dijkstra算法，Floyd算法，Bellman-Ford算法和SPFA算法等。

**（一）Dijkstra算法**：确定起点的最短路径问题- 即已知起始结点，求最短路径的问题。

**1.Dijkstra算法定义**

Dijkstra(迪杰斯特拉)算法是典型的单源最短路径算法，用于计算一个节点到其他所有节点的最短路径。

主要特点：以起始点为中心向外层层扩展，直到扩展到终点为止。

指定一个点（源点）到其余各个顶点的最短路径，也叫做“单源最短路径”。

**2.算法步骤：**

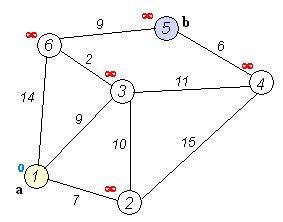
a.初始时，S只包含源点，即S＝{v}，v的距离为0。U包含除v外的其他顶点，即:U={其余顶点}，若v与U中顶点u有边，则<u,v>正常有权值，若u不是v的出边邻接点，则<u,v>权值为∞。

b.从U中选取一个距离v最小的顶点k，把k，加入S中（该选定的距离就是v到k的最短路径长度）。

c.以k为新考虑的中间点，修改U中各顶点的距离；若从源点v到顶点u的距离（经过顶点k）比原来距离（不经过顶点k）短，则修改顶点u的距离值，修改后的距离值的顶点k的距离加上边上的权。

d.重复步骤b和c直到所有顶点都包含在S中。

**3.执行动画如下：**



**代码如下：**

**const int MAXINT = 32767;**

**const int MAXNUM = 10;**

**int dist[MAXNUM];**

**int prev[MAXNUM];**

**int A[MAXUNM][MAXNUM];**

**void Dijkstra(int v0)**

**{**

**bool S[MAXNUM]; // 判断是否已存入该点到S集合中**

**int n=MAXNUM;**

**for(int i=1; i<=n; ++i)**

**{**

**dist[i] = A[v0][i];**

**S[i] = false; // 初始都未用过该点**

**if(dist[i] == MAXINT)**

**prev[i] = -1;**

**else**

**prev[i] = v0;**

**}**

**dist[v0] = 0;**

**S[v0] = true;**

**for(int i=2; i<=n; i++)**

**{**

**int mindist = MAXINT;**

**int u = v0; 　　 // 找出当前未使用的点j的dist[j]最小值**

**for(int j=1; j<=n; ++j)**

**if((!S[j]) && dist[j]<mindist)**

**{**

**u = j; // u保存当前邻接点中距离最小的点的号码**

**mindist = dist[j];**

**}**

**S[u] = true;**

**for(int j=1; j<=n; j++)**

**if((!S[j]) && A[u][j]<MAXINT)**

**{**

**if(dist[u] + A[u][j] < dist[j]) //在通过新加入的u点路径找到离v0点更短的路径**

**{**

**dist[j] = dist[u] + A[u][j]; //更新dist**

**prev[j] = u; //记录前驱顶点**

**}**

**}**

**}**

**}**

**（二）Floyd算法**

1.Floyd算法的定义：**Floyd-Warshall算法**（Floyd-Warshall algorithm）是解决任意两点间的最短路径的一种算法，可以正确处理有向图或负权的最短路径问题，同时也被用于计算有向图的传递闭包。Floyd-Warshall算法的时间复杂度为O(N3)，空间复杂度为O(N2)。

**2.算法过程：**

（1）从任意一条单边路径开始。所有两点之间的距离是边的权，如果两点之间没有边相连，则权为无穷大。

（2）对于每一对顶点 u 和 v，看看是否存在一个顶点 w 使得从 u 到 w 再到 v 比已知的路径更短。

3.[时间复杂度](https://baike.baidu.com/item/%E6%97%B6%E9%97%B4%E5%A4%8D%E6%9D%82%E5%BA%A6):O(n^3)

[空间复杂度](https://baike.baidu.com/item/%E7%A9%BA%E9%97%B4%E5%A4%8D%E6%9D%82%E5%BA%A6):O(n^2)

4．优点：容易理解，可以算出任意两个节点之间的最短距离，代码编写简单。

缺点：时间复杂度比较高，不适合计算大量数据

**5.代码如下：**

1. import numpy as np
2. N = 4
3. M = 100
4. edge = np.mat([[0,2,6,4],[M,0,3,M],[7,M,0,1],[5,M,12,0]])
5. A = edge[:]
6. path = np.zeros((N,N))
7. def Floyd():
8. for i in range(N):
9. for j in range(N):
10. if(edge[i,j] != M and edge[i,j] != 0):
11. path[i][j] = i
13. print 'init:'
14. print A,'\n',path
15. for a in range(N):
16. for b in range(N):
17. for c in range(N):
18. if(A[b,a]+A[a,c]<A[b,c]):
19. A[b,c] = A[b,a] + A[a,c]
20. path[b][c] = path[a][c]
22. print 'result:'
23. print A,'\n',path
25. if \_\_name\_\_ == "\_\_main\_\_":
26. Floyd()

**Dijkstra算法和Floyd算法的区别：**

1.Floyd算法是求任意两点之间的距离，是多源最短路，而Dijkstra(迪杰斯特拉)算法是求一个顶点到其他所有顶点的最短路径，是单源最短路。  
2.Floyd算法可以算带负权的，而Dijkstra(迪杰斯特拉)算法是不可以算带负权的。并且Floyd算法不能算负权回路。  
3.Dijkstra(迪杰斯特拉)算法时间复杂度一般是o(n^2),Floyd算法时间复杂度是o(n^3),Dijkstra(迪杰斯特拉)算法比Floyd算法块。  
4.Floyd算法属于动态规划，Dijkstra(迪杰斯特拉)算法属于贪心算法。

**现实中的应用：“地图导航”**

像 Google 地图、百度地图和高德地图这样的地图软件，生活中我们应该会经常使用。如果从家开到公司，你只需要输入起始地址、结束地址，地图就会给你规划一条最优路线。这里的最优，有很多种定义，比如最短路线、最少用时路线、最少红灯路线等等。

时间最短 高速最多