1）弗洛伊德**[算法](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "_blank)**是求图最短路径的另外一种算法，其适用于求图中任意两节点之间最短路径；

2）其基本思想也是动态规划，时间复杂度是O(N^3),N代表节点个数；

3）动态规划的实现步骤是：a）找出问题的最优子结构；b）根据最优子结构求出递归解；c）以自下而上的方式求出最优解；d）解出最优解的最优路径；其难点往往是在b）环节，解决了b）环节，其它环节都是很好实现的；

4）图在运用动态规划的时候难点在于图的连通性使得迭代过程的值不一定是最优的，Dijsktra算法是通过找出每次循环中源节点到各个节点最小值来确定的，而Floyd算法是通过比较任意两节点间通过N个节点中任一节点的最小值来确定的；

5）时间复杂度是O（N^3）。

具体实现如下：

**[cpp]** [view plain](http://blog.csdn.net/wingofeagle/article/details/13091477) [copy](http://blog.csdn.net/wingofeagle/article/details/13091477)

1. // Floyd\_ShortestPath.cpp : Defines the entry point for the console application.
2. //
4. #include "stdafx.h"
5. #include "stdio.h"
7. #define OK 1
8. #define ERROR 0
9. #define TRUE 1
10. #define FALSE 0
11. #define MAXEDGE 20
12. #define MAXVEX 20
13. #define INFINITY 65535
15. **typedef** **int** Status; /\* Status是函数的类型,其值是函数结果状态代码，如OK等 \*/
17. **typedef** **struct**
18. {
19. **int** vexs[MAXVEX];
20. **int** arc[MAXVEX][MAXVEX];
21. **int** numVertexes, numEdges;
22. }MGraph;
24. **typedef** **int** Patharc[MAXVEX][MAXVEX];
25. **typedef** **int** ShortPathTable[MAXVEX][MAXVEX];
27. /\* 构件图 \*/
28. **void** CreateMGraph(MGraph \*G)
29. {
30. **int** i, j;
32. /\* printf("请输入边数和顶点数:"); \*/
33. G->numEdges=16;
34. G->numVertexes=9;
36. **for** (i = 0; i < G->numVertexes; i++)/\* 初始化图 \*/
37. {
38. G->vexs[i]=i;
39. }
41. **for** (i = 0; i < G->numVertexes; i++)/\* 初始化图 \*/
42. {
43. **for** ( j = 0; j < G->numVertexes; j++)
44. {
45. **if** (i==j)
46. G->arc[i][j]=0;
47. **else**
48. G->arc[i][j] = G->arc[j][i] = INFINITY;
49. }
50. }
52. G->arc[0][1]=1;
53. G->arc[0][2]=5;
54. G->arc[1][2]=3;
55. G->arc[1][3]=7;
56. G->arc[1][4]=5;
58. G->arc[2][4]=1;
59. G->arc[2][5]=7;
60. G->arc[3][4]=2;
61. G->arc[3][6]=3;
62. G->arc[4][5]=3;
64. G->arc[4][6]=6;
65. G->arc[4][7]=9;
66. G->arc[5][7]=5;
67. G->arc[6][7]=2;
68. G->arc[6][8]=7;
70. G->arc[7][8]=4;

73. **for**(i = 0; i < G->numVertexes; i++)
74. {
75. **for**(j = i; j < G->numVertexes; j++)
76. {
77. G->arc[j][i] =G->arc[i][j];
78. }
79. }
81. }
83. /\* Floyd算法，求网图G中各顶点v到其余顶点w的最短路径P[v][w]及带权长度D[v][w]。 \*/
84. **void** ShortestPath\_Floyd(MGraph G, Patharc \*P, ShortPathTable \*D)
85. {
86. **int** v,w,k;
87. **for**(v=0; v<G.numVertexes; ++v) /\* 初始化D与P \*/
88. {
89. **for**(w=0; w<G.numVertexes; ++w)
90. {
91. (\*D)[v][w]=G.arc[v][w]; /\* D[v][w]值即为对应点间的权值 \*/
92. (\*P)[v][w]=w;               /\* 初始化P \*/
93. }
94. }
95. **for**(k=0; k<G.numVertexes; ++k)
96. {
97. **for**(v=0; v<G.numVertexes; ++v)
98. {
99. **for**(w=0; w<G.numVertexes; ++w)
100. {
101. **if** ((\*D)[v][w]>(\*D)[v][k]+(\*D)[k][w])
102. {/\* 如果经过下标为k顶点路径比原两点间路径更短 \*/
103. (\*D)[v][w]=(\*D)[v][k]+(\*D)[k][w];/\* 将当前两点间权值设为更小的一个 \*/
104. (\*P)[v][w]=(\*P)[v][k];/\* 路径设置为经过下标为k的顶点 \*/
105. }
106. }
107. }
108. }
109. }
111. **int** main(**void**)
112. {
113. **int** v,w,k;
114. MGraph G;
116. Patharc P;
117. ShortPathTable D; /\* 求某点到其余各点的最短路径 \*/
119. CreateMGraph(&G);
121. ShortestPath\_Floyd(G,&P,&D);
123. printf("各顶点间最短路径如下:\n");
124. **for**(v=0; v<G.numVertexes; ++v)
125. {
126. **for**(w=v+1; w<G.numVertexes; w++)
127. {
128. printf("v%d-v%d weight: %d ",v,w,D[v][w]);
129. k=P[v][w];              /\* 获得第一个路径顶点下标 \*/
130. printf(" path: %d",v);  /\* 打印源点 \*/
131. **while**(k!=w)             /\* 如果路径顶点下标不是终点 \*/
132. {
133. printf(" -> %d",k);  /\* 打印路径顶点 \*/
134. k=P[k][w];          /\* 获得下一个路径顶点下标 \*/
135. }
136. printf(" -> %d\n",w);    /\* 打印终点 \*/
137. }
138. printf("\n");
139. }
141. printf("最短路径D\n");
142. **for**(v=0; v<G.numVertexes; ++v)
143. {
144. **for**(w=0; w<G.numVertexes; ++w)
145. {
146. printf("%d\t",D[v][w]);
147. }
148. printf("\n");
149. }
150. printf("最短路径P\n");
151. **for**(v=0; v<G.numVertexes; ++v)
152. {
153. **for**(w=0; w<G.numVertexes; ++w)
154. {
155. printf("%d ",P[v][w]);
156. }
157. printf("\n");
158. }
160. **return** 0;
161. }

效果如下：



