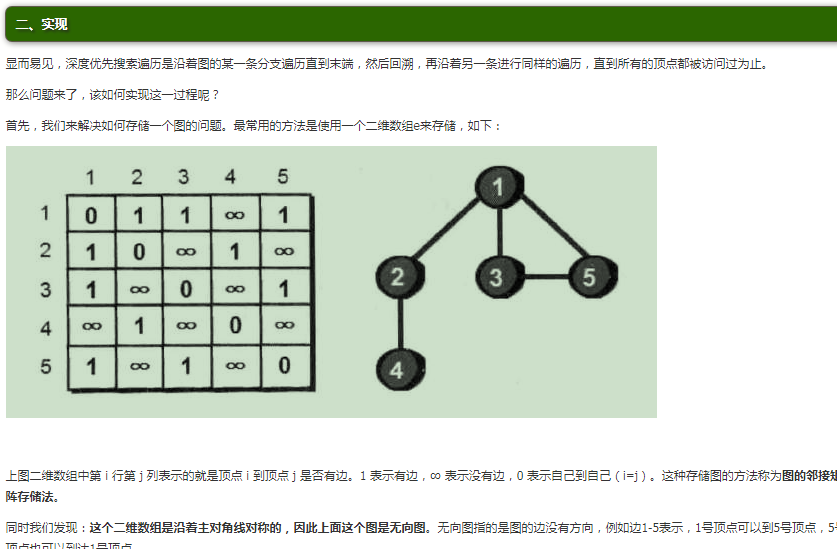
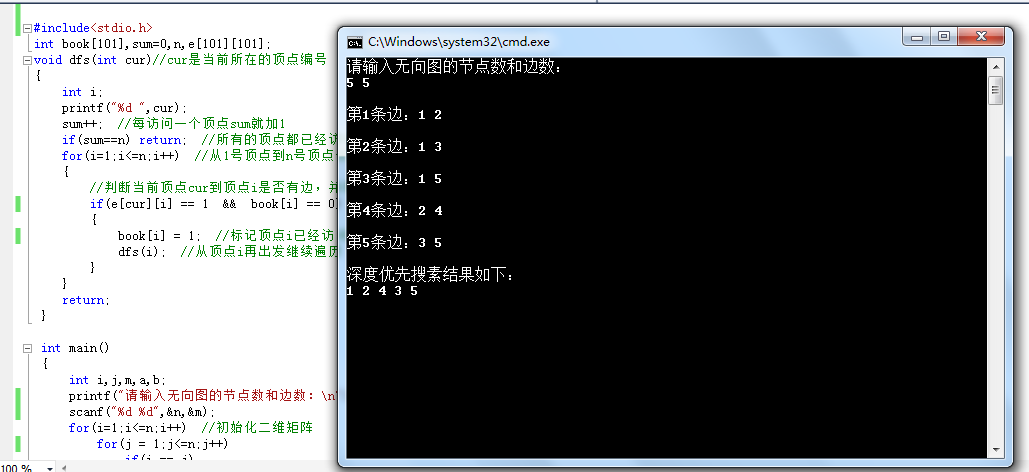
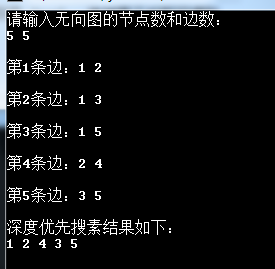
实验题目：深度优先遍历无向图

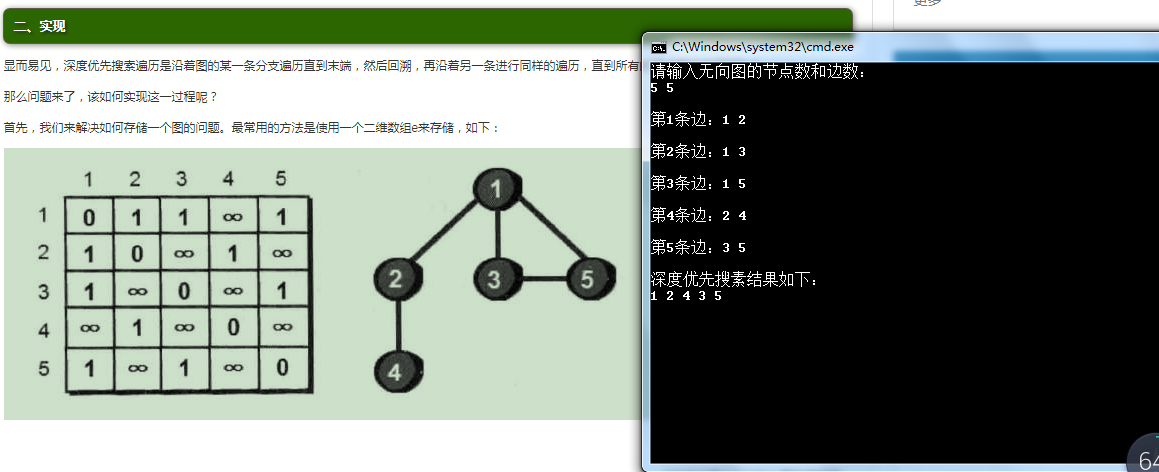
下面是我从网上找到的关于深度优先算法例题:







与题目对比如下



1. 至此本次实验成功完成，通过debug的过程我学习了更多的关于深度优先搜索的知识
2. 实验的源程序如下：
3. #include<stdio.h>
4. int book[101],sum=0,n,e[101][101];
5. void dfs(int cur)//cur是当前所在的顶点编号
6. {
7. int i;
8. printf("%d ",cur);
9. sum++;
10. if(sum==n) return; for(i=1;i<=n;i++) {
11. if(e[cur][i] == 1 && book[i] == 0)
12. {
13. book[i] = 1; //标记顶点i已经访问过
14. dfs(i); // }
15. }
16. return;
17. }
18. int main()
19. {
20. int i,j,m,a,b;
21. printf("请输入无向图的节点数和边数：\n");
22. scanf("%d %d",&n,&m);
23. for(i=1;i<=n;i++) // for(j = 1;j<=n;j++)
24. if(i == j)
25. e[i][j]=0;
26. else
27. e[i][j]=99999999; //
28. //读入顶点之间的边
29. for(i=1;i<=m;i++)
30. {
31. printf("\n第%d条边：",i);
32. scanf("%d %d",&a,&b);
33. e[a][b] = 1;
34. e[b][a] = 1; }
36. //从1号顶点出发
37. book[1]=1; //标记1号顶点已被访问
38. printf("\n深度优先搜素结果如下：\n");
39. dfs(1); //从1号顶点开始遍历
41. getchar();getchar();
42. return 0;
43. }
44. #include <iostream>
46. using namespace std;
48. //
49. int visted[100];
50. //
51. typedef struct EdgeNode {
52. //顶点对应的下标
54. int adjvex;
56. //指向下一个邻接点
58. struct EdgeNode \*next;
59. } edgeNode;
60. //顶点表结点
61. typedef struct VertexNode {
62. //
63. char data;
65. //边表头指针
67. edgeNode \*firstedge;
68. } VertexNode, AdjList[100];
70. typedef struct {
71. AdjList adjList;
72. //顶点数和边数
73. int numVertexes, numEdges;
74. } GraphAdjList;
75. class AdjacencyList {
76. public:
78. void CreateALGraph(GraphAdjList \*G);
80. void ShowALGraph(GraphAdjList \*G);
82. void DFS(GraphAdjList \*G, int i);
84. void DFSTraverse(GraphAdjList \*G);
86. void Test();

89. };
90. void AdjacencyList::CreateALGraph(GraphAdjList \*G) {
91. int i, j, k;
92. edgeNode \*e;
93. cout << "输入顶点数和边数" << endl;
95. cin >> G->numVertexes >> G->numEdges;
96. //读入顶点信息，建立顶点表
97. for (i = 0; i < G->numVertexes; i++)
98. {
99. //输入顶点信息
100. cin >> G->adjList[i].data;
102. G->adjList[i].firstedge = NULL;
104. }
105. //建立边表（头插法）
107. for (k = 0; k < G->numEdges; k++)
108. {
109. cout << "输入边（vi,vj）上的顶点序号" << endl;
110. cin >> i >> j;
111. e = new EdgeNode;
112. e->adjvex = j;
113. e->next = G->adjList[i].firstedge;
114. G->adjList[i].firstedge = e;
116. e = new EdgeNode;
118. e->adjvex = i;
119. e->next = G->adjList[j].firstedge;
120. G->adjList[j].firstedge = e;
122. }
123. }
124. void AdjacencyList::Test() {
125. cout << "ALL IS OK." << endl;
126. }
127. void AdjacencyList::ShowALGraph(GraphAdjList \*G) {
128. for (int i = 0; i < G->numVertexes; i++)
129. {
130. cout << "顶点" << i << ": " << G->adjList[i].data << "--firstedge--";
131. edgeNode \*p = new edgeNode;
132. p = G->adjList[i].firstedge;
133. while (p)
134. {
135. cout << p->adjvex << "--Next--";
136. p = p->next;
138. }
139. cout << "--NULL" << endl;
140. }
142. }
143. void AdjacencyList::DFS(GraphAdjList \*G, int i) {
144. EdgeNode \*p;
145. visted[i] = 1;
146. cout << G->adjList[i].data << "--";
147. p = G->adjList[i].firstedge;
148. while (p)
149. {
150. if (!visted[p->adjvex])
151. {
153. DFS(G, p->adjvex);
155. }
156. p = p->next;
157. }
159. }
160. void AdjacencyList::DFSTraverse(GraphAdjList \*G) {
161. //初始化所有顶点都没有访问过
162. cout<<"深度优先遍历结果为："<<endl;
163. for (int i = 0; i < G->numVertexes; i++)
164. {
165. visted[i] = 0;
166. }
167. for (int i = 0; i < G->numVertexes; i++)
168. {
169. if (visted[i] == 0)
170. {
171. DFS(G, i);
172. }
173. }
174. }
175. int main() {
177. AdjacencyList adjacencyList;
178. GraphAdjList \*GA = new GraphAdjList;
179. adjacencyList.Test();
180. adjacencyList.CreateALGraph(GA);
181. adjacencyList.ShowALGraph(GA);
182. adjacencyList.DFSTraverse(GA);
183. return 0;
184. }