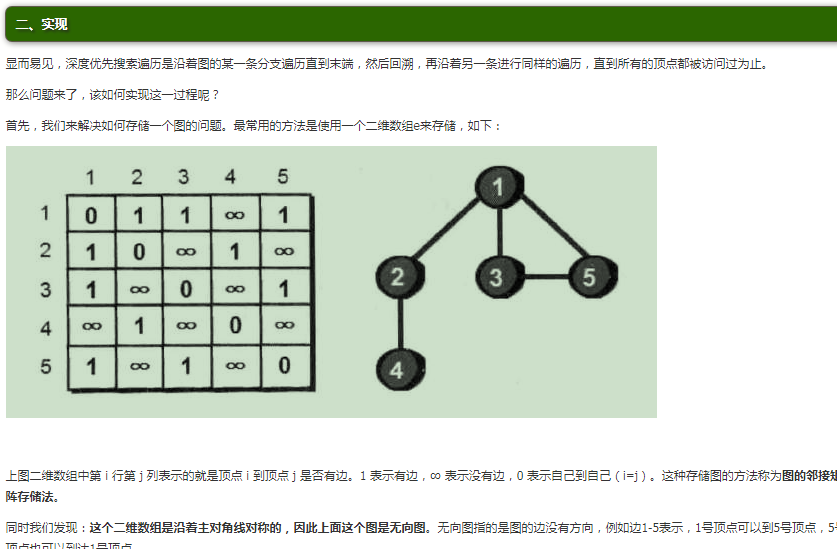
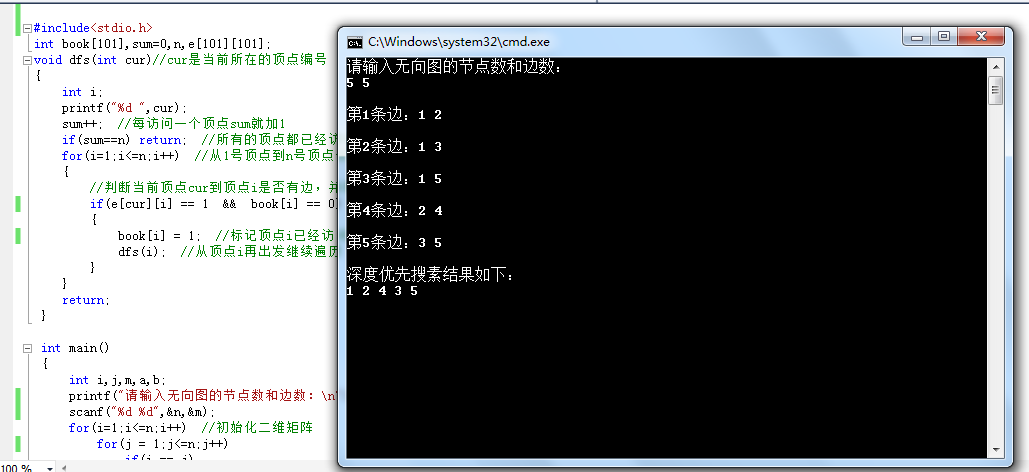
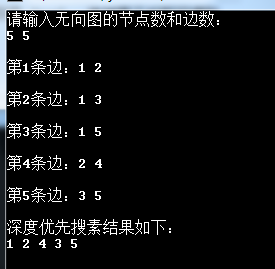
实验题目：深度优先遍历无向图

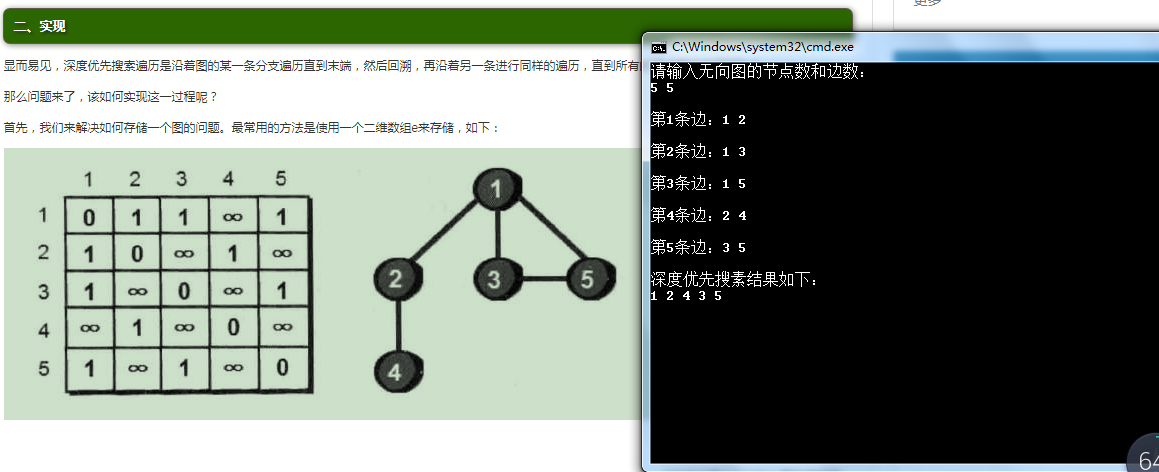
下面是我从网上找到的关于深度优先算法例题:







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 <iostream>
4. using namespace std;
5. int visted[100];
6. typedef struct EdgeNode {
7. int adjvex;//指向下一个邻接点
8. struct EdgeNode \*next;
9. } edgeNode;//顶点表结点
10. typedef struct VertexNode {
11. char data;//边表头指针
12. EdgeNode \*firstedge;
13. } VertexNode, AdjList[100]；
14. typedef struct {
15. // AdjList adjList;
16. //顶点数和边数
17. int numVertexes, numEdges;
18. } GraphAdjList;
19. class AdjacencyList {
20. public:
21. void CreateALGraph(GraphAdjList \*G);
22. void ShowALGraph(GraphAdjList \*G);
23. void DFS(GraphAdjList \*G, int i);
24. void DFSTraverse(GraphAdjList \*G);
25. void Test();
26. };
27. void AdjacencyList::CreateALGraph(GraphAdjList \*G) {
28. int i, j, k;
29. edgeNode \*e;
30. cout << "输入顶点数和边数" << endl;
31. cin >> G->numVertexes >> G->numEdges;
32. //读入顶点信息，建立顶点表
33. for (i = 0; i < G->numVertexes; i++)
34. {
35. cin >> G->adjList[i].data;
36. G->adjList[i].firstedge = NULL;
37. }
38. //建立边表（头插法）
39. for (k = 0; k < G->numEdges; k++)
40. {
41. cout << "输入边（vi,vj）上的顶点序号" << endl;
42. cin >> i >> j;
43. e = new EdgeNode;
44. e->adjvex = j;
45. e->next = G->adjList[i].firstedge;
46. G->adjList[i].firstedge = e;
47. e = new EdgeNode;
48. e->adjvex = i;
49. e->next = G->adjList[j].firstedge;
50. G->adjList[j].firstedge = e;
51. }
52. }
53. void AdjacencyList::Test() {
54. cout << "ALL IS OK." << endl;
55. }
56. void AdjacencyList::ShowALGraph(GraphAdjList \*G) {
57. for (int i = 0; i < G->numVertexes; i++)
58. {
59. cout << "顶点" << i << ": " << G->adjList[i].data << "--firstedge--";
60. edgeNode \*p = new edgeNode;
61. p = G->adjList[i].firstedge;
62. while (p)
63. {
64. cout << p->adjvex << "--Next--";
65. p = p->next;
66. }
67. cout << "--NULL" << endl;
68. }
69. }
70. void AdjacencyList::DFS(GraphAdjList \*G, int i) ;
71. EdgeNode \*p;
72. visted[i] = 1;
73. cout << G->adjList[i].data << "--";
74. p = G->adjList[i].firstedge;
75. while (p)
76. {
77. if (!visted[p->adjvex])
78. {
79. DFS(G, p->adjvex);
80. }
81. p = p->next;
82. }
83. }
84. void AdjacencyList::DFSTraverse(GraphAdjList \*G) {
85. //初始化所有顶点都没有访问
86. cout<<"深度优先遍历结果为："<<endl;
87. for (int i = 0; i < G->numVertexes; i++)
88. {
89. visted[i] = 0;
90. }
91. for (int i = 0; i < G->numVertexes; i++)
92. {
93. if (visted[i] == 0)
94. {
95. DFS(G, i);
96. }
97. }
98. }
99. int main() {
100. AdjacencyList adjacencyList;
101. GraphAdjList \*GA = new GraphAdjList;
102. adjacencyList.Test();
103. adjacencyList.CreateALGraph(GA);
104. adjacencyList.ShowALGraph(GA);
105. adjacencyList.DFSTraverse(GA);
106. return 0;
107. }