1. **自选存储结构，编写一算法判断无向图中任意给定的两个顶点间是否存在一条长度为k的简单路径（即不含回路的路径）。**

#include <iostream>

using namespace std;

#define MAX\_VERTEX\_NUM 100 // 顶点数目最大值

struct ArcNode // 弧结点

{

int adjvex; // 邻接点位置坐标

ArcNode \*next;

};

struct VexNode // 头结点

{

int data;

ArcNode \*firstarc;

};

struct ALGraph // 以邻接表存储的图

{

VexNode vexs[MAX\_VERTEX\_NUM]; // 邻接表adjlist

int vexnum, arcnum;

};

ALGraph \*crt\_graph(void)

{

ALGraph \*g = new ALGraph;

cout << "请输入图的顶点数:";

cin >> g->vexnum;

cout << "请输入图的边数:";

cin >> g->arcnum;

cout << "请输入图的所有顶点的值（空格间隔）:";

for (int i = 1; i <= g->vexnum; i++)

cin >> g->vexs[i].data;

cout << "请输入图的所有边的两端顶点:" << endl;

for (int j = 1; j <= g->arcnum; j++)

{

int vex1, vex2;

cin >> vex1 >> vex2;

ArcNode \*newArc = new ArcNode;

newArc->adjvex = vex2; // 头插法

newArc->next = g->vexs[vex1].firstarc, g->vexs[vex1].firstarc = newArc;

newArc = new ArcNode;

newArc->adjvex = vex1;

newArc->next = g->vexs[vex2].firstarc, g->vexs[vex2].firstarc = newArc;

}

return g;

}

bool dfs(ALGraph G, int start, int end, int length, int visit[], int d)

{

visit[start] = 1; // 访问标记

if (start == end && d == length) // 找到一条路径，返回真

return true;

ArcNode \*p = G.vexs[start].firstarc; // 指针指向第一个邻接点

while (p != NULL)

{

int w = p->adjvex; // 若w结点未访问，递归地访问它

if (visit[w] == 0 && dfs(G, w, end, length, visit, d + 1))

return true;

p = p->next;

}

visit[start] = 0; // 回退操作

return false;

}

int main()

{

ALGraph \*g = crt\_graph();

int start, end, length;

cout << "请输入起始顶点，目标顶点和路径长度（空格间隔）: ";

cin >> start >> end >> length;

int visit[MAX\_VERTEX\_NUM] = {0};

bool result = dfs(\*g, start, end, length, visit, 0);

if (result)

cout << "存在长度为" << length << "的简单路径" << endl;

else

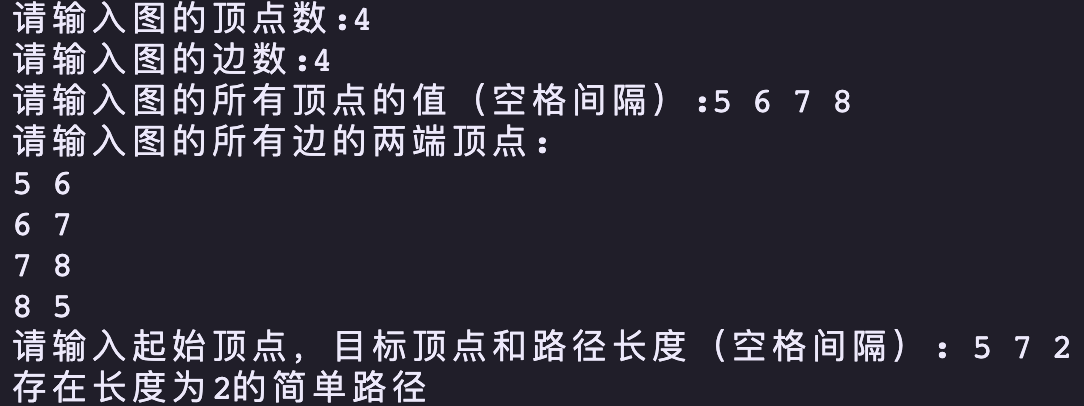
cout << "不存在长度为" << length << "的简单路径" << endl;

delete g;

return 0;

}

运行结果如下：



1. **自选存储结构，试给出求有向图中所有简单回路（即其中不再含有回路的回路）的算法。**

ps：各struct声明和create函数与20题一样，此处省略

bool visit[MAX\_VERTEX\_NUM] = {0};

int path[MAX\_VERTEX\_NUM] = {0};

// 找curVtx的简单回路

void getPath(ALGraph g, int curVtx, int k)

{

visit[curVtx] = true, path[k++] = curVtx;

for (ArcNode \*p = g.vexs[curVtx].firstarc; p != NULL; p = p->next) // 遍历该结点所有邻接点

{

if (visit[p->adjvex] == false) // 有非访问的邻接点

{

if (k > 2 && path[k-2] == p->adjvex) // 路径>2

{

cout << "回路：";

for (int i = k-2; i < k; i++)

cout << g.vexs[path[i]].data << "->";

cout << g.vexs[curVtx].data << endl;

}

else // 继续遍历

getPath(g, p->adjvex, k);

}

}

visit[curVtx] = false;

}

int main()

{

ALGraph \*graph = crt\_graph();

for (int i = 1; i <= graph->vexnum; i++)

getPath(\*graph, i, 0);

return 0;

}