**5-2最小长度电路板排列问题**

#include <iostream>

#include <fstream>

#include <queue>

#include <algorithm>

using namespace std;

const int MAX = 50;

int p[MAX][MAX];

int bestx[MAX];

int n, m; //电路板数，连接块数

class Node

{

public:

int dep; //当前深度

int cd; //当前排列长度

int \*x; //存储当前排列x[1:dep]

int \*low; //电路块中最左边电路板

int \*high; //电路块中最右边电路板

Node()

{

cd = 0;

dep = 0;

high = new int[m+1];

low = new int[m+1];

x = new int[n+1];

}

int len() //计算当前排列最小长度

{

int temp = 0;

for(int j=1; j<=m; j++)

{

if(low[j]<=n && high[j]>0 && temp<high[j]-low[j])

temp = high[j] - low[j];

}

return temp;

}

};

int search()

{

queue<Node> q;

Node enode;

int bestd = n + 1;

int i, j;

for(j=1; j<=m; j++)

{

enode.high[j] = 0;

enode.low[j] = n + 1;

}

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

enode.x[i] = i;

while(true)

{

if(enode.dep == n-1) //仅一个儿子结点，已经排完n-1个电路板，现在排最后一个

{

for(int j=1; j<=m; j++)

if(p[ enode.x[n] ][j]>0 && n>enode.high[j])

enode.high[j] = n;

enode.cd = enode.len();

if(enode.cd < bestd)

{

bestd = enode.cd;

copy(enode.x, enode.x+n+1, bestx);

}

}

else

{

int cur = enode.dep + 1;

for(i=enode.dep+1; i<=n; i++) //产生当前扩展结点的所有儿子结点

{

Node now;

for(int j=1; j<=m; j++)

{

now.low[j] = enode.low[j];

now.high[j] = enode.high[j];

if(p[ enode.x[i] ][j] > 0)

{

if(cur < now.low[j])

now.low[j] = cur;

if(cur > now.high[j])

now.high[j] = cur;

}

}

now.cd = now.len();

if(now.cd < bestd)

{

now.dep = enode.dep + 1;

copy(enode.x, enode.x+n+1, now.x);

now.x[now.dep] = enode.x[i]; //相当于回溯中的swap(x[dep], x[i])

now.x[i] = enode.x[now.dep];

q.push(now);

}

}

}

if(q.empty())

break;

else

{

enode = q.front(); //下一层扩展结点

q.pop();

}

}

return bestd;

}

int main()

{

ifstream fin("电路板.txt");

cout << "输入电路板个数：";

fin >> n; cout << n;

cout << "\n输入连接块个数：";

fin >> m; cout << m;

cout << "\n输入矩阵：\n";

int i, j;

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

{

for(j=1; j<=m; j++)

{

fin >> p[i][j];

cout << p[i][j] << " ";

}

cout << endl;

}

cout << "\n排列的最小长度为：" << search();

cout << "\n最佳排列为：\n" ;

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

cout << bestx[i] << " ";

cout << endl;

cout << endl;

fin.close();

return 0;

}