实验三

实验目的

传教士与野人问题

实验内容

有N个传教士和N个野人来到河边渡河，河岸有一条船，每次至多可供k人乘渡。问传教士为了安全起见，应如何规划摆渡方案，使得任何时刻，河两岸以及船上的野人数目总是不超过传教士的数目(否则不安全，传教士有可能被野人吃掉)。即求解传教士和野人从左岸全部摆渡到右岸的过程中，任何时刻满足M(传教士数)≥C(野人数)和M+C≤k的摆渡方案。

代码：

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

int X, Y;

int k;

struct node

{

int q[3];

};

vector<node> s;

int q[500][3];

//用于存放搜索结点，q[][0]是左岸传教士人数

//q[][1]是左岸野蛮人人数，q[][2]是左岸船的数目

//q[][3]用于搜索中的父亲结点序号。

int ans = 0;

int op\_num = 0;

int go[500][2];

int fx[500][500];

//安全状态：左岸中，传教士都在or都不在or传教士人数等于野人人数

int is\_safe(int state[3])

{

if ((state[0] == 0 || state[0] == X || state[0] == state[1]) && (state[1] >= 0) && (state[1] <= Y))

{

return 1;

}

return 0;

}

//是否到达目标状态

int is\_success(int state[3])

{

if (state[0] == 0 && state[1] == 0)

return 1;

return 0;

}

//该状态是否已经访问过

int vis(int state[3])

{

for (vector<node>::iterator it = s.begin(); it != s.end(); it++)

if ((\*it).q[0] == state[0] && (\*it).q[1] == state[1] && (\*it).q[2] == state[2])

return 1;

return 0;

}

int f1(int state[3])

{

return state[0] + state[1];

}

int f2(int state[3])

{

return state[0] + state[1] - 2 \* state[2];

}

int find\_max(int cur)

{

int max = -1;

int op = -1;

for (int j = 0; j < op\_num; j++)//分别考虑可能的动作

{

if (fx[cur + 1][j] > max)

{

max = fx[cur + 1][j];

op = j;

}

}

if (max == -1)

op = -1;

return op;

}

//过河操作

int search(int cur)

{

if (is\_success(q[cur]))

{

ans = cur;

return 1;

}

int state[3];

int j;

//cout<<"第"<<cur<<"层搜索"<<endl;

//获取当前搜索结点

//cout<<"展开结点"<<cur<<":"<<q[cur][0]<<' '<<q[cur][1]<<' '<<q[cur][2]<<endl;

if (q[cur][2])//船在左边

{

for (j = 0; j < op\_num; j++)//分别考虑可能的动作

{

state[0] = q[cur][0] - go[j][0];

state[1] = q[cur][1] - go[j][1];

state[2] = 0;//船到了右边

fx[cur + 1][j] = f2(state);

}

j = find\_max(cur);

while (j != -1)

{

fx[cur + 1][j] = -1;

state[0] = q[cur][0] - go[j][0];

state[1] = q[cur][1] - go[j][1];

state[2] = 0;//船到了右边

if (is\_safe(state) && !vis(state))//如果是安全状态//判断与之前展开结点是否相同

{

node nd;

nd.q[0] = q[cur + 1][0] = state[0];

nd.q[1] = q[cur + 1][1] = state[1];

nd.q[2] = q[cur + 1][2] = state[2];

s.push\_back(nd);

//cout<<"合法结点:"<<state[0]<<' '<<state[1]<<' '<<state[2]<<endl;

if (search(cur + 1))

return 1;

}

j = find\_max(cur);

}

}

else //船在右边

{

for (j = 0; j < op\_num; j++)//分别考虑可能的动作

{

state[0] = q[cur][0] + go[j][0];

state[1] = q[cur][1] + go[j][1];

state[2] = 1;

fx[cur + 1][j] = f2(state);

}

j = find\_max(cur);

while (j != -1)

{

fx[cur + 1][j] = -1;

state[0] = q[cur][0] + go[j][0];

state[1] = q[cur][1] + go[j][1];

state[2] = 1; //船回到左边

if (is\_safe(state) && !vis(state))//如果是安全状态且与之间状态不同

{

node nd;

nd.q[0] = q[cur + 1][0] = state[0];

nd.q[1] = q[cur + 1][1] = state[1];

nd.q[2] = q[cur + 1][2] = state[2];

s.push\_back(nd);

//cout<<"合法结点:"<<state[0]<<' '<<state[1]<<' '<<state[2]<<endl;

if (search(cur + 1))

return 1;

}

j = find\_max(cur);

}

}

return 0;

}

int main()

{

int n;

cout << "请输入N：";

cin >> n;

cout << "请输入k：";

cin >> k;

X = Y = n;

int state[3];

//初始状态

node nd;

nd.q[0] = state[0] = q[0][0] = X;

nd.q[1] = state[1] = q[0][1] = Y;

nd.q[2] = state[2] = q[0][2] = 1;

s.push\_back(nd);

//初始化操作

cout << "合法的操作组有：" << endl;

for (int i = 1; i <= k; i++)

for (int j = 0; j <= i; j++)

{

if (j >= i - j || j == 0)

{

go[op\_num][0] = j;

go[op\_num][1] = i - j;

cout << go[op\_num][0] << ' ' << go[op\_num][1] << endl;

op\_num++;

}

}

cout << endl;

if (!search(0))

{

cout << "无解" << endl;

return 0;

}

cout << "找到的解为:" << endl;

for (int i = 0; i <= ans; i++)

{

//cout<<q[i][0]<<' '<<q[i][1]<<' '<<q[i][2]<<endl;

if (i > 0)

{

cout << abs(q[i][0] - q[i - 1][0]) << "个传教士和" << abs(q[i][1] - q[i - 1][1]) << "个野人";

if (q[i][2])

cout << "从右岸乘船至左岸" << endl;

else

cout << "从左岸乘船至右岸" << endl;

cout << "左岸有" << q[i][0] << "个传教士和" << q[i][1] << "个野人" << endl;

cout << "右岸有" << n - q[i][0] << "个传教士和" << n - q[i][1] << "个野人" << endl << endl;

}

}

cout << "本次搜索所花费的费用：" << ans << endl;

return 0;

}

实验结果

