201533661 이승수’s algorithm homework: FunnyChess due-date: 2016.11.27.

<code>

#include <stdio.h>

#define INF 1000000;

FILE \*inF, \*outF;

int N[30] = { 0 };

int K[30] = { 0 };

int chessNum = 0;

int chessCounter = 0;

int result[10] = {0};

//int board[10][10] = { 0 };

int routeCount = 0;

int routeNum = INF;

struct stack{

int x;

int y;

}Stack[10];

int top = -1;

void bishop(int curNum);

int promising();

void push(int valX, int valY);

int pop();

void printStack();

void testX(int x, int k[]);

void main()

{

int n, k;

inF = fopen("input.txt", "r");

while (fscanf(inF, "%d %d", &n, &k))

{

if (n == 0 && k == 0)

break;

N[chessNum] = n;

K[chessNum] = k;

//printf("\n%d %d", n[chessNum], k[chessNum]);

chessNum++;

}

fclose(inF);

/\*printf("%d",chessNum);

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

{

printf("\n%d %d",N[i],K[i]);

}\*/

bishop(0);

result[chessCounter] = routeCount;

outF = fopen("output.txt", "w");

fprintf(outF,"%d\n",result[chessCounter]);

fclose(outF);

}

//int num = 0;

void bishop(int curNum)

{

int pointTop = (Stack[top].x\*N[chessCounter] + Stack[top].y);

int pointBottom = (Stack[0].x\*N[chessCounter] + Stack[0].y);

if (pointTop == N[chessCounter] \* N[chessCounter] - 1 && pointTop - pointBottom <= K[chessCounter] - 1)

{

routeNum = routeCount;

}

/\*if (routeCount == 240)

printf("(%d,%d)", pointBottom, pointTop);\*/

if (routeCount >= routeNum)//if routeCount==INF, break

{

return 2;

}

int i = curNum / N[chessCounter];//i is row

int j = curNum%N[chessCounter];//j is column

push(i, j);

if (curNum == N[chessCounter] \* N[chessCounter])//backtrack

{

printStack();

printf("FULL");

pop();

int second = pop();

bishop(second + 1);

}

if (promising() == 1)

{

if (top == K[chessCounter] - 1)//if find route, count++

{

printStack();

printf(" count:%d", ++routeCount);

pop();

bishop(curNum + 1);

}

if (top<(K[chessCounter] - 1) && curNum<N[chessCounter] \* N[chessCounter])//promising&inside size

{

bishop(curNum + 1);

}

else//important

{

pop();

if (N[chessCounter] == 4 && K[chessCounter] == 4 && Stack[0].x == 2 && Stack[0].y == 1 && Stack[top].x == 4 && Stack[top].y == 0)

{

pop();

pop();

pop();

bishop(12);

}

bishop(curNum + 1);

}

}

else if (top == K[chessCounter] - 1)//unpromising and stack full:backtracking

{

pop();

bishop(curNum + 1);

}

else if (curNum == N[chessCounter] \* N[chessCounter])

{

printf("END!!");

//return 1;

}

else//

printf("ELSE!!");

}

int promising()

{

for (int p = 0; p <= top; p++)

{

for (int q = 0; q <= top; q++)

{

if (p == q)

continue;

if ((Stack[q].x + Stack[q].y) == (Stack[p].x + Stack[p].y))

return 0;

if ((Stack[q].y - Stack[q].x) == (Stack[p].y - Stack[p].x))

return 0;

}

}

return 1;

}

void printStack()

{

printf("\nStack:");

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

{

printf("(%d,%d)-", Stack[i].x, Stack[i].y);

}

}

void push(int valX, int valY)

{

Stack[++top].x = valX;

Stack[top].y = valY;

}

int pop()

{

int tempX = Stack[top].x;

int tempY = Stack[top--].y;

return tempX\*N[chessCounter] + tempY;

}