问题描述

　　给定一个n\*n的棋盘，棋盘中有一些位置不能放皇后。现在要向棋盘中放入n个黑皇后和n个白皇后，使任意的两个黑皇后都不在同一行、同一列或同一条对角线上，任意的两个白皇后都不在同一行、同一列或同一条对角线上。问总共有多少种放法？n小于等于8。

输入格式

　　输入的第一行为一个整数n，表示棋盘的大小。  
　　接下来n行，每行n个0或1的整数，如果一个整数为1，表示对应的位置可以放皇后，如果一个整数为0，表示对应的位置不可以放皇后。

输出格式

　　输出一个整数，表示总共有多少种放法。

样例输入

4  
1 1 1 1  
1 1 1 1  
1 1 1 1  
1 1 1 1

样例输出

2

样例输入

4  
1 0 1 1  
1 1 1 1  
1 1 1 1  
1 1 1 1

样例输出

0

锦囊1

搜索算法。

锦囊2

先搜索n皇后的解，在拼凑成2n皇后的解。

import java.util.Scanner;

public class Main {

public static int n = 0;

public static int a[] = new int[10];

public static int isa[][] = new int[10][10];

public static boolean flag = false;

public static int sum = 0;

public static boolean judge(int k) {

for (int j = 1; j < k; j++) {

if (Math.abs(j - k) == Math.abs(a[j] - a[k]) || a[j] == a[k])

return false;

}

return true;

}

public static void place(int t) {

if (t > n && !flag) {

flag = true;

place(1);

flag = false;

} else if (t > n && flag) {

sum++;

} else {

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

if (isa[t][i] == 1) {

int temp = a[t];

a[t] = i;

if (judge(t)) {

isa[t][i] = 2;

place(t + 1);

isa[t][i] = 1;

}

a[t] = temp;

}

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

n = sc.nextInt();

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

for (int j = 1; j <= n; j++) {

isa[i][j] = sc.nextInt();

}

}

place(1);

System.out.println(sum);

}

}

#include<stdio.h>

int a[9][9],n,sum;

int row1[9],diagl1[17],diagr1[17],row2[9],diagl2[17],diagr2[17];

void dfs2(int i)

{

int j;

if(i==n+1)

{

sum++;

return;

}

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

{

if(a[i][j]==1&&row2[j]!=0&&diagl2[n-i+j]!=0&&diagr2[i+j]!=0)

{

a[i][j]=0;

row2[j]=0;

diagl2[n-i+j]=0;

diagr2[i+j]=0;

dfs2(i+1);

row2[j]=1;

diagl2[n-i+j]=1;

diagr2[i+j]=1;

a[i][j]=1;

}

}

}

void dfs1(int i)

{

int j,h,r;

if(i==n+1)

{

for(h=1,r=1;r<=n;r++)

{

if(a[1][r]==1)

{

a[1][r]=0;

row2[r]=0;

diagl2[n-h+r]=0;

diagr2[h+r]=0;

dfs2(h+1);

row2[r]=1;

diagl2[n-h+r]=1;

diagr2[h+r]=1;

a[1][r]=1;

}

}

}

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

{

if(a[i][j]==1&&row1[j]!=0&&diagl1[n-i+j]!=0&&diagr1[i+j]!=0)

{

a[i][j]=0;

row1[j]=0;

diagl1[n-i+j]=0;

diagr1[i+j]=0;

dfs1(i+1);

row1[j]=1;

diagl1[n-i+j]=1;

diagr1[i+j]=1;

a[i][j]=1;

}

}

}

int main()

{

int i,j;

while(scanf("%d",&n)!=EOF)

{

sum=0;

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

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

scanf("%d",&a[i][j]);

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

row1[i]=row2[i]=1;

for(i=1;i<=2\*n;i++)

diagl1[i]=diagr1[i]=diagl2[i]=diagr2[i]=1;

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

{

if(a[1][j]==1)

{

a[1][j]=0;

row1[j]=0;

diagl1[n-i+j]=0;

diagr1[i+j]=0;

dfs1(i+1);

row1[j]=1;

diagl1[n-i+j]=1;

diagr1[i+j]=1;

a[1][j]=1;

}

}

printf("%d\n",sum);

}

return 0;

}

#include<cstdio>

using namespace std;

int n;

int sum;

bool g[9][9];

bool wh[9];

bool wd[17];

bool wu[17];

bool bh[9];

bool bd[17];

bool bu[17];

void white(int h){

if(h==n){

sum++;

}else{

for(int i=0;i<n;i++){

if(!g[h][i])continue;

if(wh[i])continue;

if(wd[i+h])continue;

if(wu[(i-h)+n])continue;

wh[i]=wd[i+h]=wu[(i-h)+n]=1;

white(h+1);

wh[i]=wd[i+h]=wu[(i-h)+n]=0;

}

}

}

void black(int h){

if(h==n){

white(0);

}else{

for(int i=0;i<n;i++){

if(!g[h][i])continue;

if(bh[i])continue;

if(bd[i+h])continue;

if(bu[(i-h)+n])continue;

g[h][i]=0;

bh[i]=bd[i+h]=bu[(i-h)+n]=1;

black(h+1);

g[h][i]=1;

bh[i]=bd[i+h]=bu[(i-h)+n]=0;

}

}

}

int main(){

int i;

int x;

sum=0;

scanf("%d",&n);

for(i=0;i<n;i++){

wh[i]=bh[i]=0;

wd[i]=bd[i]=0;

wu[i]=bu[i]=0;

for(int j=0;j<n;j++){

scanf("%d",&x);

g[i][j]=(bool)x;

}

}

for(;i<2\*n;i++){

wd[i]=bd[i]=0;

wu[i]=bu[i]=0;

}

black(0);

printf("%d\n",sum);

return 0;

}