**金陵十三钗**

问题描述

　　在电影《金陵十三钗》中有十二个秦淮河的女人要自我牺牲代替十二个女学生去赴日本人的死亡宴会。为了不让日本人发现，自然需要一番乔装打扮。但由于天生材质的原因，每个人和每个人之间的相似度是不同的。由于我们这是编程题，因此情况就变成了金陵n钗。给出n个女人和n个学生的相似度矩阵，求她们之间的匹配所能获得的最大相似度。  
　　所谓相似度矩阵是一个n\*n的二维数组like[i][j]。其中i,j分别为女人的编号和学生的编号，皆从0到n-1编号。like[i][j]是一个0到100的整数值，表示第i个女人和第j个学生的相似度，值越大相似度越大，比如0表示完全不相似，100表示百分之百一样。每个女人都需要找一个自己代替的女学生。  
　　最终要使两边一一配对，形成一个匹配。请编程找到一种匹配方案，使各对女人和女学生之间的相似度之和最大。

输入格式

　　第一行一个正整数n表示有n个秦淮河女人和n个女学生  
　　接下来n行给出相似度，每行n个0到100的整数，依次对应二维矩阵的n行n列。

输出格式

　　仅一行，一个整数，表示可获得的最大相似度。

样例输入

4  
97 91 68 14  
8 33 27 92  
36 32 98 53  
73 7 17 82

样例输出

354

数据规模和约定

　　对于70%的数据，n<=10  
　　对于100%的数据，n<=13

样例说明

最大相似度为91+92+93+73=354

C++代码：

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

#define MAXN 15

using namespace std;

const int maxzt=(1<<13); //最大的状态数,

int dp[maxzt];

int like[MAXN][MAXN],n;

int numberOfOne(int num){ //num 二进制中1的个数

int cnt=0;

while(num){

cnt+=(num&1);

num>>=1;

}

return cnt;

}

int lowbit(int x){ //num 二进制中只保留最后一个1 如: num=20 二进制10100 返回 二进制100，也就是4

return x&(-x);

}

int posOfOne(int num){ //num 二进制中最后一个1的位置 如: num=18 二进制10010 返回 2

int pos=0;

while(num){

pos++;

if(num&1)

return pos;

num>>=1;

}

return pos;

}

void work(int x){

int maxstatus=1<<n;

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

int nowstatus=i,t=numberOfOne(nowstatus);

if(t!=x) continue; //第x位女人有x个1，不是就继续找下一个数

while(t--){

int pos=lowbit(nowstatus);

dp[i]=max(dp[i],dp[i-pos]+like[x][posOfOne(pos)]);

nowstatus-=pos;

}

}

}

int main(){

scanf("%d",&n);

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

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

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

memset(dp,0,sizeof(dp));

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

work(i);

printf("%d\n",dp[(1<<n)-1]);

return 0;

}

C代码：

#include <stdio.h>

#include <string.h>

#define max(a, b) a > b ? a : b

int dp[14][1024 \* 8], a[14][14], n;

int f(int x, int fn)

{

if (fn == n)

return 0;

if (dp[fn][x] != -1)

return dp[fn][x];

dp[fn][x] = 0;

int i;

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

if (!(x & (1 << i)))

{

int fx = x | (1 << i);

dp[fn][x] = max(dp[fn][x], f(fx, fn + 1) + a[fn][i]);

}

return dp[fn][x];

}

int main()

{

int i, j, max = 0;

scanf("%d", &n);

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

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

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

memset(dp, -1, sizeof(dp));

f(0, 0);

max = dp[0][0];

for (i = 0; i < n - 1; i++)

if (max < dp[0][1 << i])

max = dp[0][1 << i];

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

return 0;

}

JAVA代码：

import java.util.Scanner;

public class Main

{

final int MAXN=15;

int maxzt=(1<<13);

int dp[]=new int[maxzt];

int like[][]=new int[MAXN][MAXN];

int n;

int numberOfOne(int num)

{

int cnt=0;

while(num>0){

cnt+=(num&1);

num>>=1;

}

return cnt;

}

int lowbit(int x){

return x&(-x);

}

int posOfOne(int num){

int pos=0;

while(num>0){

pos++;

if((num&1)>0)

return pos;

num>>=1;

}

return pos;

}

void work(int x){

int maxstatus=1<<n;

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

int nowstatus=i,t=numberOfOne(nowstatus);

if(t!=x) continue;

while(t>0){

int pos=lowbit(nowstatus);

dp[i]=Math.max(dp[i],dp[i-pos]+like[x][posOfOne(pos)]);

nowstatus-=pos;

t--;

}

}

}

public static void main(String[] args)

{

Main o=new Main();

Scanner sc = new Scanner(System.in);

o.n=sc.nextInt();

for(int i=1;i<=o.n;i++)

for(int j=1;j<=o.n;j++)

o.like[i][j]=sc.nextInt();

for(int i=1;i<=o.n;i++)

o.work(i);

System.out.println(o.dp[(1<<o.n)-1]);

sc.close();

}

}