[INFOMAN](http://infoman.musala.com/welcome/main.html) [брой 16](http://infoman.musala.com/magazine/archive/issue16/index.html)

{ Това са входните данни за програмата:

4

1 1 1 1 1 1 1 1 0

1 0 0 0 0 0 0 0 1

1 0 0 0 1 0 0 0 1

1 0 0 0 0 0 0 0 1

ИГРАТА "ФЛИП"

------------

(задачата е от първи ден на 7-мата балканска олимпиада по информатика)

Дадена е матрица от нули и единици с 9 стълба и M реда (1<=M<=1000). Флип-

ване означава заместване на нулите с единици, а единиците с нули. Играе се

игра, в която са позволени са две операции - флипване на ред и флипване на

стълб. Целта на играта е да се определи най-малкият възможен брой единици,

които могат да се получат в резултат на прилагане на горните операции,като

броят им няма значение. Входът представлава едно число - брой редове в мат-

рицата и матрицата зададена ред по ред - по 9 числа "0" или "1"на ред, раз-

делени с интервали. Изходът е минималният брой единици,които могат да оста-

нат в матрицата след прилагане на поредица от позволени операции.

РЕШЕНИЕ: Трябва да отбележим, че при всички игри подобни на дадената

редът на прилагане на операциите няма значение, защото резултатът в крайна

сметка е еднакъв. Освен това е съвсем ясно, че прилагането на една и съща

операция повече от веднъж е безсмислена,защото двукратното флипване на ред

или стълб не променя матрицата. Като изходим от тези разсъждения, можем да

дадем ефективен алгоритъм за решаване на задачата. Понеже колоните са вина-

ги само 9, то всички възможни поредици от флипвания на колони са всъщност

всички подмножества на 9-елементно множество,които са точно 512. Понеже ре-

дът на операциите няма значение, приемаме, че винаги започваме с флипване

на колони. Да предположим, че сме свършили с вертикалните флипвания (на ко-

лоните). Получили сме някаква матрица с "0" и "1". Можем всеки от редовете

или да го запазим същия или да го флипнем. Очевидно, има полза от флипване

на реда, ако след флипване в него ще останат по-малко единици, т.е. всеки

от редовете трябва да се флипне, само ако в него има 5 или повече единици

и не трябва да се флипва в противен случай. Броят на единиците, които оста-

ват не може да се намали чрез повече флипвания на редове и затова в крайна

сметка колкото единици останат в матрицата, толкова е резултатът при съот-

ветната последователност от флипване на колони. След като изследваме всич-

ките 512 комбинации от флипвания на колоните и изчислим за всяка от тях ре-

зултата, можем да вземем най-малкия и той ще е решението на задачата.

Относно реализацията на алгоритъма можем да споменен, че матрицата се

пази в масив от M елемента - 16 битови цели числа, като всеки бит отразява

една клетка от матрицата на играта.В цикъл се генерират всичките 512 комби-

нации от вертикални флипвания,като всяка една от тях е представена с число

от 0 до 511 - битова маска, в която колоните, които се флипват са единици.

За всяка комбинация от флипвания на колони се намира състоянието на всеки

от редовете, което се получава с операцията "изключващо или" приложена вър-

ху стойността, определяща реда и стойността на битовата маска за колоните.

Това състояние определя битово реда след прилагане на вертикалните флипва-

ния в битовата маска. На това число се преброяват единиците - C1 и към об-

щата сума се добавя Min( C1, 9-C1 ) - минималното от нулите и единиците в

този ред. В крайна сметка получаваме сумарния минималне брой единици,който

остава при всяка от вертикалните битови маски. Минимумът на всичките тези

минимуми е търсения в задачата резултат. За повече подробности вижте реали-

зацията на алгоритъма. Интересно е как се намира най-бързо броя битове "1"

в някакво число N.Очевидният алгоритъм с изместване вдясно извършва винаги

logN операции, а хитрият алгоритъм извършва толкова операции, колкото са

единиците в N, почти винаги по-малко от logN. Разгледайте реализацията на

функцията CountsOfBits1 за повече информация.

Благодарности на Боян Кроснов за идеята за побитово

представяне на данните и съответните бързи алгоритми

Автор на алгоритъма и програмата

--------------------------------

Светлин Наков

септември, 1999

}

{$R-,S-,Q-}

CONST InFileName = 'FLIPGAME.PAS';

OutFileName = '';

MaxM = 16384;

Size = 9;

VAR M: integer;

Game: array[1..MaxM] of word;

MinCount: longint;

Procedure ReadData;

Var F: Text;

i: integer;

ch: char;

Begin

Assign(F,InFileName); Reset(F); ReadLn(F);

FillChar(Game,SizeOf(Game),0);

ReadLn(F,M);

for M:= 1 to M do

begin

Game[M]:=0;

for i:= 0 to Size-1 do

begin

Read(F,ch); while ch=' ' do Read(F,ch);

if ch = '1' then Game[M]:=Game[M] or (1 shl i);

end;

ReadLn(F);

end;

Close(F);

End;

Function CountOfBits1(N:word): word;

Var C: word;

Begin

C:=0;

while N <> 0 do

begin inc(C); N:= N and (N-1); end;

CountOfBits1:=C;

End;

Procedure Solve;

Function Min(a,b:word): word;

Begin if a < b then Min:=a else Min:=b; End;

Var CombinationBitMask, Row, RowBitMask, C1: word;

Count1: longint;

Begin

MinCount:=MaxLongInt;

for CombinationBitMask := 0 to (1 shl Size)-1 do

begin { За всички комбинации от вертикални флипвания намираме }

Count1:=0; { минималния брой оставащи единици Count1 }

for Row:= 1 to M do

begin

RowBitMask:= Game[Row] xor CombinationBitMask;

C1:= CountOfBits1(RowBitMask);

Count1:= Count1 + Min( C1, Size-C1 );

end;

if Count1 < MinCount then

MinCount:=Count1;

end;

End;

Procedure PrintSolution;

Var OutF: Text;

Begin

Assign(OutF,OutFileName); Rewrite(OutF);

WriteLn(OutF,MinCount);

Close(OutF);

End;

BEGIN

ReadData;

Solve;

PrintSolution;

END.