Unitate

Această problemă a fost rezolvată corect de către 43 dintre participanții la cea de-a treia rundă a *Concursului de Programare Agora*. O rezolvare a acestei probleme poate fi găsită în cartea *Proiectarea și implementarea algoritmilor* scrisă de *Mihai Oltean*.

Autorul propune o soluție folosind programarea dinamică. Se construiește o matrice B de dimensiuni $m \times n$ în care elementul b_{ij} reprezintă numărul minim de unități care trebuie adăugate pentru ca toate elementele matricei inițiale care se află pe primele i linii și primele j coloane să aibă valoarea 1.

Cu alte cuvinte, elementul b_{ij} conține numărul de unități care trebuie adăugate pentru ca submatricea care are colțul stânga-sus în poziția (1, 1) și colțul dreapta-jos în poziția (i, j) să conțină numai elemente cu valoarea 1.

Modul de calcul al elementelor matricei B este următorul:

- Elementele de pe prima linie se calculează pe baza formulelor: $b_{11} = 1 - a_{11}$ și $b_{1,i} = 1 - a_{1,i} + c_{1,i-1}$ pentru $i > 1$.
- Elementele de pe prima coloană se calculează pe baza unor formule asemănătoare: $b_{11} = 1 - a_{11}$ și $b_{i,1} = 1 - a_{i,1} + c_{i-1,1}$ pentru $i > 1$.
- Dacă $a_{ij} = 1$, atunci nu mai trebuie adăugată nici o unitate pentru a obține cifra 1 pe poziția (i, j), deci vom avea
$$c_{i,j} = c_{i,j-1} + c_{i-1,j} - c_{i-1,j-1}.$$
- Dacă $a_{ij} = 0$, atunci trebuie adăugate un număr minim de unități pentru a obține cifra 1 pe poziția (i, j). Dacă notăm cu k minimul dintre i și j, vom adăuga valoarea 2^{k-1}, deci formula va deveni
$$c_{i,j} = c_{i,j-1} + c_{i-1,j} + c_{i-1,j-1} + 2^{k-1}.$$

Analizând această soluție observăm că nu se adaugă unități decât în cazul în care întâlnim un element $a_{ij} = 0$, iar numărul de unități adăugate în acest caz este $2^{\min(i,j)-1}$. Astfel algoritmul de programare dinamică poate fi înlocuit cu următorul:

```
algoritm unitate
  total ← 0
  pentru i ← 1, m execută
    pentru i ← 1, n execută
      dacă aij=0 atunci
          total ← total + 2^min(i,j)-1
      sfârșit dacă
    sfârșit pentru
  sfârșit pentru
  returnează total
sfârșit algoritm
```

Datorită faptului că matricea poate avea până la 100 de linii sau coloane, s-ar putea să trebuiască să adunăm valori foarte mari cum ar fi 2^{99}. Datorită acestui fapt va trebui să implementăm operațiile de adunare și înmulțire pentru numere întregi foarte mari.

Pentru a mări timpul de execuție nu vom efectua adunările în momentul în care vom întâlni un element cu valoarea 0, ci vom păstra un vector v care să indice de câte ori va trebui să adunăm valoarea 2^i. O putere a lui 2 poate fi foarte ușor determinată pe baza puterii anterioare și pentru fiecare putere vom înmulți rezultatul cu valoarea corespunzătoare din vectorul v, apoi îl vom aduna la valoarea care va indica în final numărul de unități care trebuie adăugate.

Soluția oficială a acestei probleme este prezentată în continuare:

Listing UNIT.CPP

```
#include <stdio.h>

int  m,n,v[100],
     units[50],dim,
     p2[50],dim2,
     x[50],dimx;

void ReadData(void){
  FILE *f=fopen("UNIT.IN","rt");
  fscanf(f,"%d%d",&m,&n);
  int x;
  for (int i=0;i<m;i++)
    for (int j=0;j<n;j++){
      fscanf(f,"%d",&x);
      if (!x)
        v[((i<j)?i:j)]++;
    }
}

void MultiplyBy2(void){
  int x,t=0;
  for (int i=0;i<dim2;i++){
    x=(p2[i]<<1)+t;
```

```
    p2[i]=x%10;
    t=x/10;
  }
  if (t)
    p2[dim2++]=1;
}

void Multiply(int v){
  int t=0;
  for (int i=0;i<dimx;i++)
    x[i]=0;
  for (i=0;i<dim2;i++){
    x[i]=(p2[i]*v+t)%10;
    t=(p2[i]*v+t)/10;
  }
  dimx=dim2;
  while (t){
    x[dimx++]=t%10;
    t/=10;
  }
}

void Add(void){
  int v,t=0;
  dim=((dim>dimx)?dim:dimx);
  for (int i=0;i<dim;i++){
    v=units[i]+x[i]+t;
    units[i]=v%10;
    t=v/10;
  }
  if (t)
    units[dim++]=1;
}

void BuildSolution(void){
  dim2=dimx=dim=p2[0]=1;
  for (int i=0;i<((m<n)?m:n);i++){
    Multiply(v[i]);
    Add();
    MultiplyBy2();
  }
}

void WriteSolution(void){
  FILE *f=fopen("UNIT.OUT","wt");
  for (int i=dim-1;i>=0;i--)
    fprintf(f,"%d",units[i]);
  fclose(f);
}

void main(void){
  ReadData();
  BuildSolution();
  WriteSolution();
}
```

Au rezolvat corect:

Csaba Andras, Oradea
Marius Andrei, Bucureşti
Mugurel Ionuţ Andreica, Bucureşti
Liviu-Cosmin Andreicuţ, Bucureşti
Victor Asavei, Drobeta Turnu Severin
Cristian Băicoianu Ploieşti
Ciprian Baicu, Drobeta Turnu Severin
Livia Bana, Drobeta Turnu Severin
Valentin Bisa, Lugoj
Denis Bogdănaş, Iaşi
Ciprian Cană, Vaslui
Adrian Cârcu, Bistriţa
Daniel Crişan, Râmnicu Vâlcea
Andrei David, Râmnicu Vâlcea
Laurent Demonet, Franţa
George Drumea, Bucureşti
Octavian-Daniel Dumitran, Bucureşti
Dan Ghinea, Bucureşti
Aurelian Ghiţă, Buzău
Vincent Groenhuis, Olanda
Claudiu Gruia, Bucureşti
Ioana-Maria Ileană, Alba Iulia
Liviu Lalescu, Craiova
Codruţ-Lucian Lazăr, Câmpeni
Bogdan Lucaciu, Drobeta Turnu Severin
Maximilian Machedon, Bucureşti
Andrei Markovits, Satu Mare
Cosmin-Silvestru Negruseri, Bistriţa
Bogdan Nicolae, Sibiu
Sorin Otescu, Braşov
Mihai-Vlad Pantiş, Cluj-Napoca
Flaviu Paşca, Bistriţa
Mihai Pătraşcu, Craiova
Liviu Păunescu, Constanţa
Dan Popovici, Arad
Mohammad Ali Safari Ghahsareh, Iran
Lucian-Raul Silistru, Vaslui
Bogdan Stan, Câmpina
Mihai Stroe, Bucureşti
Radu Ştefan, Braşov
Cristian Toth, Lugoj
Vlad Vâlceanu, Arad
Radu Vătavu, Suceava