Alb și negru

Această problemă a fost rezolvată corect de către 87 dintre cei care au participat la a cincia rundă a ***Concursului de Programare Agora***. Chiar dacă problema nu era foarte ușoară, rezolvarea ei putea fi găsită în numeroase culegeri de probleme.

Metoda de rezolvare poate fi găsită foarte ușor folosind inducția matematică. Pentru început trebuie observat faptul că prima mutare admisă este fie deplasarea unei bile albe cu o poziție spre stânga, fie deplasarea unei bile negre cu o poziție spre dreapta. Vom presupune acum că vom muta bila albă; soluția corespunzătoare mutării unei bile negre la început poate fi găsită efectuând mutările simetric - în locul efectuării unei mutări cu o bilă albă vom efectua mutarea corespunzătoare cu o bilă neagră și invers. Fiecărei alternative îi corespunde o singură soluție.

Vom nota bilele albe cu `A` și bilele negre cu `B`. Configurația inițială este `|A...AAAA BBBB...B|`.

La primul pas vom muta bila albă; astfel vom ajunge în configurația `|A...AAA ABBBB...B|`.

Vom muta acum bila neagră pentru a ajunge în configurația `|A...AAABA BBB...B|`.

Urmează o nouă mutare cu o bilă neagră. Configurația în care se ajunge este `|A...AAABAB BB...B|`.

În acest moment vom efectua două mutări cu bilele albe. Configurația obținută este `|A...AA BABABB...B|`.

Urmează o deplasare cu o poziție a unei bile albe și apoi trei deplasări ale bilelor negre, cu două poziții. Configurația este `|A...ABABABA B...B|`.

Va urma o deplasare a unei bile negre cu o poziție și patru deplasări ale bilelor albe cu două poziții, o deplasare a unei bile albe cu o poziție și cinci deplasări ale bilelor negre cu două poziții etc.

După ce vom ajunge la pasul în care am efectuat n deplasări vom deplasa din nou o bilă cu o poziție și apoi vom efectua n - 1 deplasări cu două poziții etc. În final vom obține configurația cerută `|A...AAABA BBB...B|`.

Vom exemplifica algoritmul descris pentru $n = 4$:

Inițial: `|AAAA BBBB|`;
Pasul 1: `|AAA ABBBB|`;
Pasul 2: `|AAABA BBB|`;
Pasul 3: `|AAABAB BB|`;
Pasul 4: `|AAAB BABB|`;
Pasul 5: `|AA BABABB|`;
Pasul 6: `|A ABABABB|`;
Pasul 7: `|ABA ABABB|`;
Pasul 8: `|ABABA ABB|`;
Pasul 9: `|ABABABA B|`;
Pasul 10: `|ABABABAB |`;
Pasul 11: `|ABABAB BA|`;
Pasul 12: `|ABAB BABA|`;
Pasul 13: `|AB BABABA|`;
Pasul 14: `| BABABABA|`;
Pasul 15: `|B ABABABA|`;
Pasul 16: `|BBA ABABA|`;
Pasul 17: `|BBABA ABA|`;
Pasul 18: `|BBABABA A|`;
Pasul 19: `|BBABAB AA|`;
Pasul 20: `|BBAB BAAA|`;
Pasul 21: `|BB BABAAA|`;
Pasul 22: `|BBB ABAAA|`;
Pasul 23: `|BBBBA AAA|`;
Pasul 24: `|BBBB AAAA|`.

Implementarea este următoarea:

Listing BW.CPP

```
#include <stdio.h>
void main(void){
  int i,j,n;
  fscanf(fopen("BW.IN","rt"),"%d",&n);
  freopen("BW.OUT","wt",stdout);
  for (i=1;i<=n;i++){
    if (i%2){
      putchar('W');
      for (j=i;j--;putchar('b'));
    }
    else {
      putchar('B');
      for (j=i;j--;putchar('w'));
    }
  }
  for (i=n;i;i--){
    if (i%2){
      putchar('W');
      for (j=i;--j;putchar('w'));
    }
    else{
      putchar('B');
      for (j=i;--j;putchar('b'));
    }
  }
}
```

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
Csaba Andras, Oradea
Marius Andrei, București
Mugurel-Ionuț Andreica, București
Liviu-Cosmin Andreicuț, București
Elvis Asaftei, Vaslui
Victor Asavei, Drobeta Turnu Severin
Constantin Asofiei, Târgu Neamț
Ciprian Baicu, Drobeta Turnu Severin
Adrian Balasko, Zalău
Radu Baluta, Târgu Jiu
Livia Bana, Drobeta Turnu Severin
Cristian Băicoianu, Ploiești
Marius-Cosmin Belean, Arad
Andrei Benea, Focșani
Valentin Bișa, Lugoj
Denis Bogdănaș, Iași
Cristian Bojinovici, București
Daniel Bolohan, Iași
Vencel Bors, Oradea
Ralph Bottesch, Sibiu
Ștefan Bucur, București
Nadina Busuioc, Constanța
Ciprian Cană, Vaslui
Adrian Cârcu, Bistrița
Bogdan Chirilă, Botoșani
Horia Ciurdar, Timișoara
Radu Cocieru, Sibiu
Alexandra-Elena Constantin, Ploiești
Horea Coroiu, Cluj-Napoca
Arcadie Crăcan, Iași
Andrei Csibi, Cluj-Napoca
Vlad Dascălu, Bacău
Andrei David, Râmnicu Vâlcea
Laurent Demonet, Franța
George Drumea, București
Octavian-Daniel Dumitran, București
Bogdan Dumitru, Ploiești
Aurel-Mihai Fulger, Constanța
Dan Ghinea, București
Andrei Giurgiu, București
Mihai Gojol, Cluj-Napoca
Vincent Groenhuis, Olanda
Claudiu Gruia, București
Mihai Hărănguș, Cluj-Napoca
Andrei Huțu, Bacău
Alexandru Ivan, București
Silviu Julean, Arad
Artur Kuczapski, Oradea
Liviu Lalescu, Craiova
Codruț-Lucian Lazăr, Câmpeni
Florin Leu, Mediaș
Bogdan Lucaciu, Drobeta Turnu Severin
Maximilian Machedon București
Mihail-Cosmin Piț-Rada, Drobeta Turnu Severin
Alexandru Mosoi, Bacău
Cosmin-Silvestru Negruseri, Bistrița
Bogdan Nicolae, Sibiu
Sorin Otescu, Brașov
Mihai-Vlad Pantiș, Cluj-Napoca
Tiberius Pârcălabu, București
Gabriel Pârvan, Râmnicu Sărat
Flaviu Pașca, Bistrița
Csaba Pătcaș, Oradea
Mihai Pătrașcu, Craiova
Liviu Păunescu, Constanța
Arpad Perini, Sfântu Gheorghe
Bogdan-Milovan Piloga, Timișoara
Valentin Pop, Carei
Mihail Popa, București
Dan Popovici, Arad
Cosmin Răianu, Constanța
Dan-Octavian Savu, București
Lucian-Raul Silistru, Vaslui
Johannes Slotta, Germania
Bogdan Stan, Câmpina
Lucian-Daniel Stanciu, Brăila
Sorin Stoiana, Craiova
Mihai Stoicescu, Boldești-Scăeni
Mihai Stroe, București
Radu Ștefan, Brașov
Kristo Tammeoja, Estonia
Cristian Toth, Lugoj
Silviu-Gabriel Udrea, Târgu Jiu
Vlad Vâlceanu, Arad
Radu Vătavu, Suceava
Victor Vernescu, Constanța

Coeficient

Această problemă a fost rezolvată corect de către 23 dintre cei care au participat la a cincia rundă a ***Concursului de Programare Agora***. Transpusă în termeni de teoria grafurilor problema cerea determinarea drumului minim între două vârfuri ale unui graf, dar costul unui drum nu era dat de suma costurilor muchiilor componente, ci de produsul lor.

Totuși, ea poate fi foarte ușor transformată în problema clasică a drumului minim dacă transformăm operația de înmulțire în operație de adunare. Acest lucru poate fi realizat foarte ușor dacă logaritmăm costurile muchiilor și determinăm logaritmul costului drumului minim, după care calculăm foarte simplu și costul real al acestui drum.

Datorită limitei specificate de 1000 de noduri, era suficient să implementăm versiunea algoritmului lui *Dijkstra* cu complexitatea $O(n^2)$.

O variantă de implementare este următoarea:

Listing COEF.CPP

```
#include <stdio.h>
#include <math.h>
#include <alloc.h>

int n;
int len[1000];
int t[1000];
int sol[5000],
int far *links[1000];
int far *c[1000];
double far *clog[1000];
double d[1000];

void ReadData(void){
  int i,j,m,x,y,cost;
  double cost_log;
  FILE *f=fopen("COEF.IN","rt");
  fscanf(f,"%d%d",&n,&m);
  for (i=0;i<n;len[i++]=0);
  for (i=0;i<m;i++){
    fscanf(f,"%d%d%*d",&x,&y);
    len[--x]++;
    len[--y]++;
  }
  fclose(f);
```

Au rezolvat corect:

Csaba Andras, Oradea
Mugurel Ionuț Andreica, București
Ciprian Baicu, Dr. Tr. Severin
Adrian Balasko, Zalău
Vencel Bors, Oradea
Adrian Cârcu, Bistrița
Arcadie Crăcan, Iași
George Drumea, București
Daniel Dumitran, București
Bogdan Dumitru, Ploiești
Dan Ghinea, București
Artur Kuczapski, Oradea
Liviu Lalescu, Craiova
Alexandru Mosoi, Bacău
Silvestru Negruseri, Bistrița
Mihai-Vlad Pantiș, Cluj-Napoca
Tiberius Pârcălabu, București
Mihai Pătrașcu, Craiova
Liviu Păunescu, Constanța
Sorin Stoiana, Craiova
Mihai Stroe, București
Radu Ștefan, Brașov
Vlad Vâlceanu, Arad

```
  for (i=0;i<n;i++){
    links[i]=(int far*)
             farmalloc(len[i]*sizeof(int));
    c[i]=(int far*)
             farmalloc(len[i]*sizeof(int));
    clog[i]=(double far*)
           farmalloc(len[i]*sizeof(double));
    t[i]=0;
  }
  f=fopen("COEF.IN","rt");
  fscanf(f,"%*d%*d");
  for (i=0;i<m;i++){
    fscanf(f,"%d%d%d",&x,&y,&cost);
    j=t[--x];
    links[x][j]=--y;
    c[x][j]=cost;
    clog[x][j]=log(cost);
    t[x]++;
    j=t[y];
    links[y][j]=x;
    c[y][j]=cost;
    clog[y][j]=log(cost);
    t[y]++;
  }
  fclose(f);
}

void Multiply(unsigned long a){
  unsigned long k;
  int i;
  k=0;
  for (i=1;i<=sol[0];i++){
    k+=a*sol[i];
    sol[i]=k%100;
    k/=100;
  }
  while (k){
    sol[++sol[0]]=k%100;
    k/=100;
  }
}

void FindShortestPath(void){
  int i,j,imin,k;
  double min;
  char s[1000];
  for (i=0;i<n;i++){
    d[i]=1.0/0.0;
    s[i]=0;
    t[i]=-1;
  }
  for (i=0;i<len[0];i++){
    d[links[0][i]]=clog[0][i];
    t[links[0][i]]=0;
  }
  k=0;
  d[0]=0;
  s[0]=1;
  t[0]=-1;
  while (1){
    k++;
    imin=-1;
    min=1000000.0f;
    for (i=0;i<n;i++)
      if (!s[i] && d[i]<min){
        min=d[i];
        imin=i;
      }
    if (imin==-1)
      break;
    s[imin]=1;
    for (i=0;i<len[imin];i++){
      j=links[imin][i];
      if (!s[j] &&
                d[imin]+clog[imin][i]<d[j]){
        d[j]=d[imin]+clog[imin][i];
        t[j]=imin;
      }
    }
  }
}

void WriteSolution(){
  int i,j;
  FILE *f=fopen("COEF.OUT","wt");
  sol[0]=sol[1]=1;
  for (i=n-1;i;i=t[i]){
    for (j=0;j<len[i];j++)
      if (links[i][j]==t[i]){
        Multiply(c[i][j]);
        break;
      }
  }
  fprintf(f,"%d",sol[sol[0]]);
  for (i=sol[0]-1;i;i--)
    fprintf(f,"%02d",sol[i]);
  fprintf(f,"\n");
  fclose(f);
}

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

Notă

Această implementare îi aparține lui George Drumea *din București, concurentul clasat pe primul loc înaintea ultimei etape a* Concursului de Programare Agora *organizat de* Gazeta de Informatică.

Matrice

Această problemă a fost rezolvată corect de 59 dintre cei care au participat la a cincia rundă a *Concursului de Programare Agora*. Problema a fost preluată din cartea *Proiectarea și implementarea algoritmilor*, scrisă de *Mihai Oltean*.

Este evident că efectuarea a două operații asupra aceleiași linii sau a aceleiași coloane nu are nici un efect.

O valoare `-1` poate părea pe poziția (i, j) dacă s- efectuat o operație asupra liniei i și nu s-a efectuat o operație asupra linie j sau invers.

Dacă notăm cu x numărul operațiilor efectuate asupra liniilor și cu y al celor efectuate asupra coloanelor, atunci numărul total de valori `-1` va fi $x \cdot (m - y) + y \cdot (n - x)$. Această valoare trebuie să fie k, deci va trebui să rezolvăm ecuația $x \cdot (m - y) + y \cdot (n - x) = k$.

Vom rezolva ecuația considerând toate perechiile (x, y) posibile. În cazul în care aceasta are soluție, putem alege oricare x linii și oricare y coloane.

Algoritmul descris poate fi implementat astfel:

Listing MATRIX.CPP

```
#include <stdio.h>
void main(void){
  long x,y,m,n,k;
  FILE *f=fopen("MATRIX.IN","rt");
  fscanf(f,"%ld%ld%ld",&m,&n,&k);
  fclose(f);
  f=fopen("matrix.out","wt");
  for (x=0;x<=m;x++)
    for (y=0;y<=n;y++)
      if (x*(n-y)+y*(m-x)==k){
        while (x) fprintf(f,"L%d\n",x--);
        while (y) fprintf(f,"C%d\n",y--);
        fclose(f); return;
      }
  fprintf(f,"0\n");
  fclose(f);
}
```

Au rezolvat corect:

Cristian Alexandrescu, Botoşani
Csaba Andras, Oradea
Marius Andrei, Bucureşti
Mugurel Andreica, Bucureşti
Elvis Asaftei, Vaslui
Victor Asavei, Dr. Tr. Severin
Constantin Asofiei, Târgu Neamţ
Ciprian Baicu, Dr. Tr. Severin
Adrian Balasko, Zalău
Livia Bana, Dr. Tr. Severin
Andrei Benea, Focşani
Valentin Bisa, Lugoj
Denis Bogdănaş, Iaşi
Cristian Bojinovici, Bucureşti
Daniel Bolohan, Iaşi
Ciprian Cană, Vaslui
Adrian Cârcu, Bistriţa
Bogdan Chirilă, Botoşani
Horia Ciurdar, Timişoara
Alexandra Constantin, Ploieşti
Andrei Csibi, Cluj-Napoca
George Drumea, Bucureşti
Daniel Dumitran, Bucureşti
Bogdan Dumitru, Ploieşti
Dan Ghinea, Bucureşti
Andrei Giurgiu, Bucureşti
Vincent Groenhuis, Olanda
Mihai Hărănguş, Cluj-Napoca
Alexandru Ivan, Bucureşti
Artur Kuczapski, Oradea
Liviu Lalescu, Craiova
Lucian Lazăr, Câmpeni
Bogdan Lucaciu, Dr. Tr. Severin
Maximilian Machedon, Bucureşti
Mihail-Cosmin Piţ-Rada, Dr. Tr. Severin
Alexandru Mosoi, Bacău
Silvestru Negruseri, Bistriţa
Sorin Otescu, Braşov
Mihai Pantiş, Cluj-Napoca
Tiberius Pârcălabu, Bucureşti
Gabriel Pârvan, Râmnicu Sărat
Flaviu Paşca, Bistriţa
Mihai Pătraşcu, Craiova
Liviu Păunescu, Constanţa
Arpad Perini, Sfântu Gheorghe
Bogdan Piloga, Timişoara
Valentin Pop, Carei
Mohammad Ali Safari Ghahsareh, Iran
Lucian-Raul Silistru, Vaslui
Sorin Stoiana, Craiova
Mihai Stoicescu, Boldeşti-Scăeni
Mihai Stroe, Bucureşti
Radu Ştefan, Braşov
Kristo Tammeoja, Estonia
Cristian Toth, Lugoj
Vlad Vâlceanu, Arad
Radu Vătavu, Suceava
Victor Vernescu, Constanţa
Violeta Voinescu, Târgu Jiu