# Bursele AGORA

**A patra etapă a concursului de programare Bursele Agora a luat sfârșit. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, precum și clasamentul general înaintea desfășurării ultimei etape.**

## A patra rundă

Problemele celei de-a patra runde au fost mai puțin dificile decât cele de la etapele anterioare. O dovadă în acest sens este faptul că patru dintre concurenți au reușit să obțină punctajul maxim: ***Mugurel-Ionuț Andreica*** din *București*, ***Cosmin-Silvestru Negrușeri*** din *Bistrița*, ***Claudiu Raicu*** din *Pitești* și ***Paul Rusu*** din *Cluj-Napoca*.

Chiar dacă doar cei patru au obținut punctajul maxim la problema ***Arbori***, celelalte două probleme au fost rezolvate de un număr destul de mare de concurenți, deci puteau fi obținute destul de ușor punctaje de peste 200 de puncte.

Din nefericire s-au pierdut din nou puncte datorită unor erori banale cum ar fi folosirea *unit*-ului `CRT`, citirea datelor de intrare de la tastatură, afișarea pe ecran a datelor de ieșire etc. Unii concurenți au afișat diferite informații pe ecran ceea ce a dus la expirarea timpului de execuție în unele cazuri. Primii 20 clasați la această rundă sunt:

| | |
|---|---|
| *1. Mugurel-Ionuț Andreica, București* | *300* |
| *1. Cosmin-Suilvestru Negrușeri, Bistrița* | *300* |
| *1. Claudiu Raicu, Pitești* | *300* |
| *1. Paul Rusu, Cluj-Napoca* | *300* |
| *5. Radu Dondera, București* | *288* |
| *5. Maximilian Machedon, București* | *288* |
| *7. Radu Berinde, București* | *261* |
| *7. Tiberiu Dăneț, București* | *261* |
| *7. Octavian-Daniel Dumitran, București* | *261* |
| *10. Csaba Pătcaș, Oradea* | *234* |
| *11. Victor-Marius Costan, București* | *209* |
| *12. Marius-Dorin Moraru, Buftea* | *192* |
| *13. Andrei Homescu, Târgu-Jiu* | *184* |
| *14. Liviu-Cosmin Andreicuț, Bacău* | *183* |
| *15. Johannes Slotta, Germania* | *182* |
| *16. Radu-Andrei Ștefan, Brașov* | *172* |
| *17. Andrei David, Bacău* | *168* |
| *18. Andrei Vancea, Cluj-Napoca* | *164* |
| *19. Victoria Tarasova, Ucraina* | *158* |
| *20. Flaviu Roman, Cluj-Napoca* | *154* |

Datele de test folosite pentru testare, precum și variante ale ieșirilor corecte, pot fi descărcate folosind *link*-urile de pe paginile dedicate celor trei probleme. Adresele acestor pagini sunt:

- www.ginfo.ro/concurs/arbori.shtml;
- www.ginfo.ro/concurs/atlantis.shtml;
- www.ginfo.ro/concurs/celule.shtml.

Tot pe *site* veți putea găsi clasamanetul integral al rundei a patra (www.ginfo.ro/concurs/runda4/top.shtml), precum și clasamentul general (www.ginfo.ro/concurs/top.shtml).

Problemele ***Atlantis*** și ***Celule*** sunt destul de cunoscute, ele fiind prezente la diferite concusuri care s-au desfășurat în ultimii ani. Acesta este unul dintre motivele pentru care punctajele obținute de concurenți la aceste două probleme au fost relativ mari.

## Sistemul de evaluare

Anunțăm faptul că problemele ultimei runde vor fi corectate folosind un calculator dotat cu un procesor *AMD Athlon XP* 1600+. Celelalte caracteristici ale sistemului de evaluare rămân neschimbate.

## La mulți ani!

Vlad Ilie, Craiova - 4 IV 1984
Alexandra Constantin, Ploiești - 6 IV 1984
Gheorghe Fan, Ștefești - 7 IV 1984
Andrei Benea, Focșani - 10 IV 1983
Florin Ghețu, Focșani - 11 IV 1982
Silviu-Georgian Aprozeanu, Botoșani - 14 IV 1983
Mihai Stoicoi, Orăștie - 14 IV 1984
Omar Choudary, București - 15 IV 1984
Ioana-Violeta Brutaru, Hunedoara - 20 IV 1983
Radu-Teodor Băluță, Târgu-Jiu - 22 IV 1983
Bogdan Nicolae, Sibiu - 23 IV 1983
Alexandru Roșiu, București - 28 IV 1982

*Runda 4, problema 1*

# ARBORI

Această problemă a fost rezolvată corect doar de patru dintre participanții la a patra rundă a concursului *Bursele Agora*, ea fiind cea mai dificilă problemă propusă spre rezolvare la această etapă.

Algoritmul care trebuia aplicat pentru rezolvarea problemei constă din doi pași. La început se va determina costul transformării unui arbore din curte în fiecare dintre arborii din poză. Acest algoritm este destul de simplu, el neridicând dificultăți deosebite. La al doilea pas se va aplica un algoritm de determinare a unui cuplaj maxim de cost minim pentru un graf construit pe baza costurilor determinate la primul pas. Acest algoritm este mai dificil, dar el este prezentat în majoritatea lucrărilor care prezintă teoria grafurilor. Așadar, chiar dacă algoritmul nu era cunoscut, nu era foarte dificil de găsit și de implementat.

Primul pas care trebuie realizat în rezolvarea problemei este calcularea, pentru fiecare pereche de forma *arbore_din_curte - arbore_din_poză*, a costului (numărul de crengi tăiate) necesar transformării.

Pentru aceasta se poate folosi o funcție recursivă care returnează valoarea -1 dacă transformarea nu este posibilă (arborele din poză conține crengi în anumite poziții în care arborele din grădină nu conține crengi) sau costul cerut, calculat în funcție de costurile necesare transformării subarborilor stâng și drept (dacă oricare dintre ei este vid pentru arborele din poză și nevid pentru cel din curte, se adaugă valoarea 1 la costul corespunzător celuilalt subarbore și nu se mai efectuează apel recursiv pe ramura respectivă).

Este esențial să începem cu această operație, deoarece nu se dorește recalcularea acestor costuri de fiecare dată când avem nevoie de ele.

Rezultatul va fi o matrice *C*; aceasta poate fi privită ca fiind un graf bipartit; partițiile nodurilor sunt "*arborii din curte*" și "*arborii din poză*". Problema cere să asociem în mod biunivoc cât mai multor noduri din prima partiție, câte un nod din a doua partiție, astfel încât costul total să fie minim.

Așadar, problema se reduce la aplicarea unui algoritm pentru determinarea unui ***cuplaj maximal de cost minim***, care va fi rezolvată conform unor algoritmi foarte cunoscuți.

Graful se poate transforma într-o rețea de transport prin introducerea unui *nod sursă* (conectat cu arborii din curte) și a unui *nod destinație* (conectat cu arborii din poză).

**Au rezolvat corect:**

Mugurel-Ionuț Andreica, București
Cosmin-Silvestru Negruşeri, Bistrița
Claudiu Raicu, Pitești
Paul Rusu, Cluj-Napoca

Toate arcele din graful bipartit inițial au capacitatea 1, iar sensul este dinspre sursă spre destinație. Costul arcelor introduse de nodurile sursă și destinație este 0.

Trebuie să determinăm fluxul maxim de cost minim în această rețea. După cum se știe, fluxul maxim se determină găsind repetat *drumuri de creștere*, de la sursă la destinație. Un drum de creștere poate să conțină un arc (*i*, *j*), dacă nu există flux pe el, sau dacă există flux pe arcul (*j*, *i*) - deci se parcurge arcul (*j*, *i*) în sens invers.

Se caută un astfel de drum de la sursă la destinație, se actualizează fluxul pe arcele care formează drumul (devine 1 dacă valoarea a fost 0 și 0 dacă valoarea a fost 1), se caută un nou drum de creștere etc.

Operația se repetă până în momentul în care nu mai există nici un drum de creștere. Inițial se va considera că fluxurile corespunzătoare tuturor arcelor din graf au valoarea 0.

Pot exista mai multe drumuri de creștere la un moment dat. În această situație va trebui ales *drumul de creștere de cost minim*.

Costul unui drum de creștere este dat de suma costurilor arcelor componente.

Costul unui arc este pozitiv dacă valoarea fluxului de pe acel arc este 0 și negativ în cazul în care fluxul de pe acel arc are valoarea 1. Valoarea absolută a acestui flux este dată de costul determinat la pasul anterior.

În continuare se va construi graful format din arcele corespunzătoare (cele pentru care valoarea fluxului este 1 sunt inversate și vor avea cost negativ; celelalte arce sunt păstrate fără a fi modificate). În acest graf, pe baza algoritmului *Belmann-Ford*, se va determina un astfel de drum de cost minim.

Algoritmul *Belmann-Ford* va funcționa corect deoarece graful nu va conține cicluri cu cost negativ. În final, se va obține un flux maxim de cost minim; arcele pe care fluxul are valoarea 1 constituie soluția problemei.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/arbori.shtml**.



# ATLANTIS

Această problemă a fost rezolvată corect de 15 dintre concurenții care au participat la runda a patra a concursului *Bursele Agora*.

În principiu, problema ar putea fi enunțată astfel: *dându-se un număr întreg să se determine cel mai mic multiplu al său care este format doar din anumite cifre date*.

Soluția acestei probleme este destul de simplă și se bazează pe câteva observații matematice.

Pentru început vom demonstra că, dacă există un multiplu al numărului $N$ care este format doar din cifrele $x_1$, $x_2$, ..., $x_m$, atunci cel mai mic astfel de multiplu are cel mult $N$ cifre.

Vom presupune, prin absurd, că cel mai mic multiplu $M$ are $N + 1$ cifre. Eliminând ultima cifră, vom obține un număr care, împărțit la $N$, duce la obținerea unui rest $R_1$. Repetând procesul de eliminare a cifrelor, se obțin resturile $R_2$, ..., $R_N$.

Dacă cele $N$ resturi sunt distincte, atunci unul dintre ele este 0, deci există un multiplu care conține mai puțin de $N + 1$ cifre (cel mult $N$ cifre).

Dacă resturile nu sunt distincte, atunci cel puțin două dintre ele (fie acestea $R_i$ și $R_j$, $i < j$) sunt egale. Multiplul $M$ are forma $m_1...m_i...m_j...m_{N+1}$; datorită faptului că resturile $R_i$ și $R_j$ sunt egale, înseamnă că prin eliminarea cifrelor $m_{i+1}$, ..., $m_j$ se obține un număr care este, la rândul său, multiplu al numărului $N$.

Acest număr are forma $m_1...m_i m_{j+1}...m_{N+1}$, deci are cel mult $N$ cifre.

Putem trage concluzia că ipoteza inițială este falsă; așadar, dacă există un multiplu al numărului $N$ care să fie format din anumite cifre date, atunci acest multiplu conține cel mult $N$ cifre.

Ipoteza potrivit căreia multiplul ar fi format din $N + 1$ cifre nu reduce generalitatea deoarece, dacă presupunem că numărul ar fi format din $N + k$ cifre, folosind același procedeu putem obține, pe rând, multiplii formați din cel mult $N + k - 1$ cifre, $N + k - 2$ cifre și așa mai departe până se ajunge la un multiplu format din cel mult $N + 1$ cifre.

În continuare vom descrie modul în care poate fi determinat un multiplu format din cel mult $N$ cifre.

Vom crea o coadă care va conține numere prin împărțirea cărora la $N$ se obțin resturi distincte. Corespunzător fiecărui rest care poate fi obținut, în coadă se va afla cel mai mic număr care duce la obținerea restului respectiv.

Inițial vom insera în coadă numere formate dintr-o singură cifră (cifrele care le avem la dispoziție), ordonate crescător. La fiecare pas, vom alege primul element din coadă și vom încerca să adăugăm la sfârșitul său, pe rând, una dintre cifrele disponibile. Pentru a obține cele mai mici numere, cifrele vor fi considerate în ordine crescătoare.

În cazul în care un număr astfel determinat duce, prin împărțirea la $N$, la obținerea unui rest care nu a mai fost obținut anterior, atunci numărul determinat este adăugat în coadă.

După considerarea tuturor cifrelor, elementul curent este eliminat din coadă. Datorită acestor eliminări trebuie păstrat un vector de valori *booleene* care să indice dacă un rest a fost sau nu obținut anterior.

În momentul în care se obține restul 0, cel mai mic multiplu este determinat și execuția algoritmului se oprește.

Noile resturi pot fi determinate foarte repede folosind o altă observație matematică: dacă restul împărțirii unui număr $a$ la $N$ este $r$, atunci restul împărțirii la $N$ a numărului obținut prin adăugarea cifrei $x$ la sfârșitul numărului $a$ este ***rest***$[(r \cdot 10 + x) / N]$.

În cazul în care nu am reușit să obținem restul 0 considerând numere formate din cel mult $N$ cifre, atunci suntem siguri că nu există nici un multiplu format doar din cifrele date deoarece, dacă acesta ar fi existat, atunci el ar fi avut cel mult $N$ cifre.

Codul sursă al programului care implementează algoritmul descris este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/atlantis.shtml**.

**Au rezolvat corect:**

Mugurel-Ionuț Andreica, București
Radu Berinde, București
Victor-Marius Costan, București
Tiberiu Dănet, București
Radu Dondera, București
Octavian-Daniel Dumitran, București
Maximilian Machedon, București
Silvestru-Cosmin Negruțeri, Bistrița
Bogdan Nicolae, Sibiu
Andrei Paponiu, Drobeta-Turnu Severin
Csaba Pătcaș, Oradea
Claudiu Raicu, Pitești
Flaviu Roman, Cluj-Napoca
Paul Rusu, Cluj-Napoca
Johannes Slotta, Germania

*Runda 4, problema 3*

# CELULE

Aceasta a fost cea mai simplă problemă a rundei a patra, 18 dintre participanți obținând punctajul maxim.

Pentru început, vom studia modul în care se propagă semnalul în cazul în care șirul are o lungime infinită și, inițial, avem o singură celulă excitată. Vom considera că celula excitată se află pe poziția 0.

```
-9 -8 -7 -6 -5 -4 -3 -2 -1  0 +1 +2 +3 +4 +5 +6 +7 +8 +9
 0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0
 0  0  0  0  0  0  1  0  1  0  1  0  1  0  0  0  0  0  0
 0  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0
 0  0  0  0  1  0  1  0  0  0  0  0  1  0  1  0  0  0  0
 0  0  0  1  0  0  0  1  0  0  0  1  0  0  0  1  0  0  0
 0  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  1  0  0
 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
 1  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1
```

Se observă că, dacă momentul inițial este considerat a fi $t = 0$, atunci la momentele de timp de forma $t = 2^k$ vom avea celule excitate doar pe pozițiile $2^{k+1}$ și $-2^{k+1}$.

Vom considera acum un șir de celule excitate și liniștite; prima celulă a șirului se va afla pe poziția 0, a doua pe poziția 1 etc. Se observă că, la fiecare pas, configurația se obține aplicând operatorul ***xor*** (*sau exclusiv*) între configurațiile care ar fi fost obținute pentru fiecare celulă în parte luând în considerare poziția inițială a acesteia. Astfel, dacă inițial celula excitată se află pe poziția $n$, atunci la momente de timp de forma $t = 2^k$ vom avea celule excitate doar pe pozițiile $n + 2^{k+1}$ și $n - 2^{k+1}$.

Observăm că, la momente de timp de forma $t = 2^k$ vom obține configurația inițială pe șirurile de celule care încep din pozițiile $2^{k+1}$ și $-2^{k+1}$.

De exemplu, pentru șirul de celule `111` vom obține următoarele configurații:

```
-9 -8 -7 -6 -5 -4 -3 -2 -1  0 +1 +2 +3 +4 +5 +6 +7 +8 +9
 0  0  0  0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  1  1  0  1  1  0  0  0  0  0  0
 0  0  0  0  0  0  0  1  1  1  0  1  1  1  0  0  0  0  0
 0  0  0  0  0  0  1  1  0  1  0  1  0  1  1  0  0  0  0
 0  0  0  0  0  1  1  1  0  0  0  0  0  1  1  1  0  0  0
 0  0  0  0  1  1  0  1  1  0  0  0  1  1  0  1  1  0  0
```

Datorită faptului că lungimea $L$ a șirului de celule este finită, rezultă că o anumită configurație dată va ajunge la momente de timp de forma $t = 2^k$ pe poziții de forma **rest**$[2^{k+1} / L]$ și **rest**$[-2^{k+1} / L]$.

Evident, pentru ca celulele să nu se liniștească trebuie să existe o perioadă de repetare a configurațiilor și aceasta nu poate fi mai lungă decât $2^L$.

Așadar, este suficient să verificăm dacă la momentul de timp $t = 2^L$ mai există celule excitate.

## Au rezolvat corect:

Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Liviu-Cosmin Andreicuț, Bacău
Radu Berinde, București
Ștefan Bucur, București
Mihai-Daniel Cilidariu, Botoșani
Andrei David, Râmnicu Vâlcea
Tiberiu Dănet, București
Radu Dondera, București
Octavian-Daniel Dumitran, București
Andrei Homescu, Târgu-Jiu
Maximilian Machedon, București
Marius-Dorin Moraru, Buftea
Silvestru-Cosmin Negruțeri, Bistrița
Csaba Pătcaș, Oradea
Claudiu Raicu, Pitești
Paul Rusu, Cluj-Napoca
Victoria Tarasova, Ucraina
Andrei Vancea, Cluj-Napoca

La acest moment de timp, șirul inițial se va regăsi în poziții de forma **rest**$[2^{L+1} / L]$ și **rest**$[-2^{L+1} / L]$. Aplicând operatorul ***xor*** asupra celor două configurații vom obține un șir care ar putea să conțină celule excitate.

În cazul în care avem celule excitate rezultă că excitația se va menține la infinit. Dacă celulele s-au liniștit atunci este evident faptul că nu se va menține la infimit excitația.

O altă modalitate (mai puțin eficientă) de abordare a problemei era generarea configurațiilor la fiecare pas. În cazul în care celulele se liniștesc este evident că excitația nu se menține. Astfel, există șanse destul de mari ca, dacă celulele nu se liniștesc după un anumit timp, excitația să se păstreze la infinit. O altă abordare ar fi fost căutarea perioadei după care se repetă configurațiile. Astfel, pentru fiecare configurație nouă se va verifica dacă există o configurație identică obținută anterior. În caz afirmativ, se poate trage concluzia că excitația se menține la infinit.

Nici una dintre aceste două abordări nu ar fi dus la obținerea unor punctaje mulțumitoare.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/celule.shtml**.