Bursele AGORA

S-a încheiat prima etapă a concursului Bursele Agora. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, precum și cele trei probleme ale rundei a patra.

Prima rundă

Problemele primei runde a concursului nostru au fost destul de dificile. Prima dintre ele, ***Matrice***, a fost rezolvată corect doar de trei concurenți, în timp ce la problema ***Warcraft*** nici unul dintre participanți nu a obținut punctajul maxim. Problema ***Dune*** a fost relativ simplă, ea putând fi rezolvată cu ajutorul algoritmului lui *Lee*, un algoritm clasic de programare dinamică.

Este îmbucurător faptul că nu au mai apărut erori banale cum ar fi folosirea *unit*-ului CRT de către programatorii în *Pascal*, așteptarea reacțiilor de la tastatură etc. Aceste erori nu au dispărut complet, dar numărul celor care au pierdut puncte datorită lor este nesemnificativ.

Pentru a studia datele de test folosite în cadrul evaluării puteți accesa paginile *Web* dedicate celor trei probleme ale primei runde. Acestea sunt:

- www.ginfo.ro/concurs/matrice.shtml;
- www.ginfo.ro/concurs/dune.shtml;
- www.ginfo.ro/concurs/warcraft.shtml.

Pe aceste pagini veți putea găsi și prezentările soluțiilor oficiale ale problemelor, precum și legături spre fișierele sursă ale rezolvărilor date de autori. Vă reamintim faptul că toate fișierele disponibile pe *site*-ul www.ginfo.ro sunt proprietatea editurii *Agora Media* și nu pot fi folosite în scopuri comerciale fără acordul scris al acesteia.

De asemenea, puteți obține informații referitoare la evaluarea problemelor. Clasamentul primei etape este disponibil la adresa www.ginfo.ro/concurs/runda1/top.shtml.

Clasamentul general al concursului este disponibil la adresa www.ginfo.ro/concurs/top.shtml

Această pagină conține lista tuturor celor care au trimis soluții pentru această rundă. În dreptul numelor apar punctajele obținute la cele trei probleme. Aceste punctaje sunt, de fapt, legături spre paginile personalizate ale concurenților. Un *click* pe unul dintre punctaje va duce la apariția unei pagini care oferă informații detaliate despre evaluarea problemei și modul în care a fost obținut punctajul respectiv.

Runda a patra

Cele trei probleme ale rundei a patra sunt *Arbori*, *Atlantis* și *Celule*.

Prima dintre ele a fost propusă de ***Mihai Stroe***, student în anul IV la *Universitatea Politehnica* din București, fost olimpic internațional, câștigător al primei ediții a concursului nostru și clasat pe locul al doilea la ediția de anul trecut.

Cea de-a doua problemă a fost propusă de ***Mihai Scorțaru***, student în anul IV la *Universitatea Tehnică* din Cluj, redactor-șef adjunct al revistei noastre și organizator principal al concursului.

Ultima problemă a fost propusă de ***d-na lect. univ. Clara Ionescu***, cadru didactic în cadrul *Universității Babeș-Bolyai* din Cluj și redactor-șef al ***GInfo***.

Rezolvările celor trei probleme pot fi trimise până la data de ***28 februarie 2002***. Rezultatele oficiale ale rundei a patra vor deveni publice pe data de ***15 martie 2002***.

Pentru a evita întârzierile, vă recomandăm să trimiteți soluțiile cu câteva ore sau zile înaintea expirării termenului limită de trimitere a acestora.

La mulți ani!

Șerban Gîlvitu, Hunedoara - 5 I 1986
Paul-Andrei Tuțu, Arad - 9 I 1984
Mihai Stoicescu, Boldești-Scăeni - 10 I 1984
Ștefan Filip, Bacău - 12 I 1985
Nicolae Tanase-Nicola, Slatina - 13 I 1985
Constantin Asofiei, Târgu Neamț - 14 I 1984
Mugurel-Ionuț Andreica, București - 16 I 1983
Radu-Victor Sărmășag, Târgoviște - 17 I 1986
Romulus Ghiorghiaș, Brașov - 18 I 1980
Romulus-Nicolae Apolzan, Timișoara - 18 I 1982
Adrian-Gheorghe Ceaușu, Alba Iulia - 18 I 1986
Cristian Băicoianu, Ploiești - 20 I 1984
Andrei Gheorghe, București - 27 I 1983
Mircea-Șerban Savin, Iași - 30 I 1984
Andrei Ionescu, Arad - 31 I 1985

Runda 1, problema 1

MATRICE

Au rezolvat corect:

Adrian-Nicolae Cârcu, Bistrița
Radu Berinde, București
Csaba András, Oradea

Această problemă a fost rezolvată corect doar de către trei dintre cei care au participat la prima rundă a ediției 2001/2002 a concursului de programare ***Bursele Agora***. Aceasta dovedește faptul că problema a fost dificilă, dar rezolvabilă.

Metoda folosită de către autor pentru rezolvarea problemei este ***programarea dinamică***.

Considerăm matricea de intrare *X*. Vom începe cu câteva observații ajutătoare. Pentru început, numărul mic al coloanelor ar trebui să ne pună pe gânduri. Apare în mod evident întrebarea dacă această limitare ne poate ajuta la rezolvarea problemei.

Se observă că există cel mult 64 de modalități de a plasa pătrate (mai exact colțurile din stânga-sus ale acestora) pe o linie a matricei. De ce 64? Pentru că pătratele trebuie să fie incluse complet în matrice, deci colțul din stânga-sus al unui pătrat poate fi plasat numai pe una dintre primele șase coloane. De fapt, acest număr poate fi redus la 37, deoarece unele posibilități de plasare sunt inutile.

De exemplu, se observă că patru pătrate sunt suficiente pentru a acoperi întreaga linie, deci nu are sens să plasăm cinci pătrate pe o linie, deoarece cel puțin unul dintre ele acoperă poziții acoperite și de alte pătrate, deci poate fi eliminat, suma elementelor acoperite rămânând aceeași. De asemenea, nu are rost să plasăm colțurile din stânga-sus a trei pătrate în trei coloane consecutive, deoarece pozițiile acoperite de pătratul din mijloc, sunt acoperite de celelalte două pătrate.

Presupunem că știm, pentru o anumită linie, sumele maxime care pot fi obținute plasând 0, 1, 2, ..., *K* pătrate pe primele *i* linii folosind pe linia *i* o anumită configurație. Apare întrebarea dacă putem determina aceleași informații pentru linia *i* + 1.

Răspunsul este afirmativ și ne oferă rezolvarea problemei. Construim (cel puțin teoretic) tabloul tridimensional `M[1..N-1,0..K,1..37]`. Un element `M[i,j,l]` reprezintă suma maximă obținută plasând `j` pătrate pe primele `i` linii folosind configurația `l` pe linia `i`.

Elementele matricei `M[i+1]` pot fi determinate folosind matricea `M[i]`. Pentru a calcula un element al matricei `M[i+1]` avem nevoie de următoarele informații:

- suma elementelor de pe linia `i+2` care sunt acoperite de pătratele care vor fi plasate pe linia `i+1`, folosind configurația `l`;
- numărul `nr[l]` al pătratelor care sunt folosite pentru obținerea configurației `l`;
- suma elementelor de pe linia `i+1` care sunt acoperite folosind configurația `l`, dar nu și folosind o configurație `c` (determinăm valorile sumei pentru orice configurație `c`);
- valoarea maximă din `M` pentru fiecare configurație `c` plasată pe linia `i` care conține `j-nr[l]` pătrate.

Modul în care determinăm aceste informații poate fi înțeles studiind fișierul sursă.

Programul este construit astfel:

- configurațiile posibile de pe o linie sunt păstrate în matricea de constante `p`, iar numărul de pătrate folosit pentru obținerea fiecărei configurații este păstrat în vectorul `nr`;
- tabloul tridimensional `M` este simulat cu ajutorul a două matrice `A` și `B` (adică `A=M[i]`, se calculează `B=M[i+1]`, apoi `A` devine `B` etc.)
- pentru fiecare rând unde se pot plasa pătrate
  - se calculează suma adăugată pe rândul respectiv dacă plasăm o anumită configurație, în funcție de configurația plasată pe rândul anterior;
  - se calculează suma adăugată pe rândul ***următor*** dacă plasăm o anumită configurație pe rândul curent;
  - se inițializează matricea `B`;
  - se realizează calculul valorilor din `B` (datorită faptului că operațiile din această zonă sunt executate de foarte multe ori, codul trebuie optimizat la maximum; programul prezentat aici a fost optimizat după obținerea unei versiuni funcționale, dar mai lente; sunt posibile și alte optimizări, dar am preferat să ne limităm la acestea pentru a nu afecta lizibilitatea codului; versiunea pe care v-o punem la dispoziție se încadrează în limita de o secundă pentru orice set de date dacă se folosește cu o configurație echivalentă celei specificate în regulamentul concursului *Bursele Agora*);
  - `A` devine `B`.

Soluția este dată de elementul maxim obținut în final în matricea `A`.

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

Runda 1, problema 2

DUNE

Această problemă a fost rezolvată corect de 49 dintre participanții la prima rundă a concursului *Bursele Agora*, ceea ce demonstrează faptul că rezolvarea sa nu ridică dificultăți deosebite. Problema poate fi rezolvată foarte ușor folosind algoritmul lui *Lee*. La fiecare pas vom memora toate pozițiile în care poate ajunge *Muad'Dib*.

Inițial *Muad'Dib* se află în poziția marcată cu `'*'`. Vom marca cu `'#'` toate regiunile în care se poate ajunge din poziția inițială mergând în direcția indicată de primul element al șirului de caractere dat.

La fiecare dintre următorii pași vom efectua următoarele operații:

- vom marca cu `'.'` toate pozițiile care nu sunt marcate cu `'+'` sau cu `'#'`;
- vom marca cu `'*'` toate pozițiile care sunt marcate cu `'#'`;
- vom marca cu `'+'` poziția în care se află *sietch*-ul de plecare:
- pentru fiecare poziție marcată cu `'*'`:
  - ♦ vom marca cu `'#'` toate regiunile în care se poate ajunge din poziția considerată mergând în direcția indicată de elementul curent al șirului de caractere dat;
  - ♦ există posibilitatea să marcăm cu `'#'` poziții care au fost marcate cu `'*'`; acest lucru nu afectează corectitudinea algoritmului, deoarece din pozițiile care au fost marcate cu `'*'` nu se poate ajunge în alte poziții decât cele deja marcate cu `'#'`.

În final, soluția va fi dată de numărul pozițiilor marcate cu `'#'`.

Aceasta este o prezentare teoretică a rezolvării. Practic, pentru simplitate, nu se mai face diferență între primul și ceilalți pași ai algoritmului. Astfel, se iau în considerare toate pozițiile marcate cu `'*'`, se marchează cu `'#'` toate pozițiile în care se poate ajunge, se marchează cu `'.'` toate pozițiile marcate cu `'*'`, se marchează cu `'+'` poziția *sietch*-ului de pornire și se repetă acești pași până când nu mai are loc nici o schimbare de direcție. În final, se numără pozițiile marcate cu `'*'`.

Mulți dintre concurenți nu au luat în considerare anunțul general care indica faptul că *sietch*-ul de pornire reprezintă o regiune stâncoasă deoarece s-au înscris după publicarea acestuia și nu au mai primit mesajul în care se preciza că acest anunț poate fi citit pe *site*-ul oficial al concursului. Am decis ca rezolvările care nu iau în considerare acest anunț să fie considerate corecte.

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

Au rezolvat corect:

Cristian Alexandrescu, Botoşani
Csaba András, Oradea
Mugurel-Ionuţ Andreica, Bucureşti
Liviu-Cosmin Andreicuţ, Bacău
Adrian Balaşko, Zalău
Andrei Benea, Focşani
Adrian-Nicolae Cârcu, Bistriţa
Mihai-Daniel Cilidariu, Botoşani
Phuong Dao The, Vietnam
Vlad Dascălu, Bacău
Lorena-Iulia Dăian, Cluj-Napoca
Tiberiu Dăneţ, Bucureşti
Radu Dondera, Bucureşti
George-Adrian Drumea, Bucureşti
Dorin-Daniel Dudău, Târgu-Jiu
Octavian-Daniel Dumitran, Bucureşti
Ştefăniţă Fechete, Suceava
Ştefan Gheorghe, Bucureşti
Andrei Giurgiu, Bucureşti
Ioan-Claudiu Gruia, Bucureşti
Andrei Homescu, Târgu-Jiu
Liviu-Laurenţiu Iacob, Bucureşti
Alexandru Ivan, Bucureşti
Slotta Johannes, Germania
Andrew Kirilenko, Belarus
Cosmin Lehene, Zalău
Maximilian Machedon, Bucureşti
Aurelian Marinescu, Braşov
Andrei Markovits, Satu Mare
Marius-Dorin Moraru, Buftea
Alexandru Moşoi, Bacău
Silvestru-Cosmin Negruşeri, Bistriţa
Bogdan Nicolae, Sibiu
Alexandru Oprea, Sighişoara
Sorin Otescu, Braşov
Alexandru Palade, Târgu Mureş
Mihai Pantelimon, Bucureşti
Csaba Pătcaş, Oradea
Alpár-Ferenc Perini, Sfântu Gheorghe
Alexandru Radu, Bârlad
Claudiu Raicu, Piteşti
Constantin Răuţu, Botoşani
Lucian-Iulian Saucă, Braşov
Dan-Octavian Savu, Bucureşti
Radu Andrei Ştefan, Braşov
Eszter Tasi, Sfântu Gheorghe
Silviu-Gabriel Udrea, Târgu-Jiu
Andrei Vancea, Cluj-Napoca
Mihai Voinescu, Târgu-Jiu

Runda 1, problema 3

WARCRAFT

Nici unul dintre participanți nu a reușit să obțină toate cele 100 de puncte corepunzătoare acestei probleme.

Cele mai mari punctaje au fost obținute de ***Mugurel-Ionuț Andreica*** din *București* (62 de puncte), ***Constantin Asofiei*** din *Târgu Neamț* (53 de puncte) și ***Ștefăniță Fechete*** din *Suceava* (50 de puncte).

Dintre cei 86 de concurenți care au trimis o soluție pentru această problemă, doar nouă au reușit să obțină cel puțin 20 de puncte.

Până în acest moment nu a fost găsit un algoritm polinomial pentru rezolvarea acestei probleme, deci o putem considera ca fiind *NP*-completă chiar dacă nu avem la dispoziție o demonstrație matematică a acestui fapt. Lăsând la o parte metoda *backtracking*, rămâne să căutăm o rezolvare bazată pe metode euristice sau cu ajutorul unui algoritm probabilistic.

În cele ce urmează vom prezenta un algoritm evolutiv care poate fi folosit pentru rezolvarea acestei probleme.

Reamintim că algoritmii evolutivi constau în alegerea unei populații inițiale și apoi efectuarea unor mutații asupra acestora cu scopul de a îmbunătăți soluțiile obținute. În final, este aleasă cea mai bună soluție obținută. Evident, nu se poate garanta că această soluție este întotdeuana cea corectă, dar practica arată că, în majoritatea situațiilor, soluția descrisă este acceptabilă.

Mai întâi va trebui să alegem structura unui membru al populației. Alegerea nu este foarte dificilă, un membru poate fi reprezentat de o matrice cu elemente `'+'` și `'.'`, unde fiecare simbol reprezintă teritoriile uneia dintre națiuni.

O soluție va reprezenta configurația hărții după efectuarea schimburilor de teritorii. Evident, numărul elementelor cu valoarea `'+'` trebuie să fie identic cu numărul elementelor cu valoarea `'+'` citite la intrare. Analog, numărul elementelor cu valoarea `'.'` trebuie să fie identic cu numărul elementelor cu valoarea `'.'` citite la intrare.

De asemenea, trebuie să existe doar două zone disjuncte compacte: una ale cărei elemente au valoarea `'+'` și una ale cărei elemente au valoarea `'.'`.

Trebuie să găsim o măsură a performanței unui individ. Aceasta poate fi găsită foarte ușor: cu cât există mai puține diferențe între configurația individului și harta inițială, cu atât individul este mai performant. Evident, indivizii pentru care există mai mult de două zone disjuncte, trebuie penalizați. Vom aplica câte o penalizare pentru fiecare zonă suplimentară găsită.

Conceptul de algoritm evolutiv implică folosirea unui singur operator pentru îmbunătățirea soluțiilor și anume ***mutația***.

Prin mutație se efectuează o mică modificare asupra configurației corespunzătoare unui individ. În cazul nostru cea mai mică modificare este reprezentată de un schimb de teritorii. Pentru aceasta vom alege aleator o poziție căreia îi corespunde un element cu valoarea `'+'` și o poziție căreia îi corespunde un element cu valoarea `'.'` și vom interschimba valorile celor două elemente. După efectuarea mutației vom studia performanța noului individ și în cazul în care elementul este mai performant decât elementul asupra căruia se efectuează mutația, noul elemenet este inclus în populație, iar vechiul element este eliminat.

În final vom alege cel mai performant individ care, în majoritatea cazurilor, reprezintă soluția problemei. Vom număra pozițiile în care elementul din configurația corespunzătoare individului diferă de elementul din configurația inițială (citită din fișierul de intrare). Numărul schimburilor de teritorii este obținut prin înjumătățirea acestei valori și va fi scris în fișierul de ieșire. Înjumătățirea se datorează faptului că printr-un schimb de teritorii apar diferențe în două poziții, deci numărul diferențelor este egal cu dublul schimburilor de teritorii.

În cazul acestei probleme, un rol foarte important îl are generarea populației inițiale. Algoritmul standard impune generarea unor populații aleatoare, dar această regulă nu se aplică în acest caz.

Este indicat ca, pentru generarea populației inițiale, să folosiți câteva metode euristice de găsire a soluției. În unele cazuri aceste metode vor duce chiar la găsirea soluției dar, în majoritatea cazurilor vor duce doar la o configurație mai mult sau mai puțin apropiată de soluție. Totuși, numărul mutațiilor care trebuie efectuate asupra configurațiilor obținute prin metode euristice este mult mai mic și există șanse mult mai mari de obținere a soluțiilor.

Datorită faptului că nici un concurent nu a rezolvat integral problema, am decis să nu prezentăm metodele euristice pe care le-am folosit pentru generarea populației inițiale.

Va trebui să vă folosiți imaginația pentru a găsi astfel de metode care să ducă la obținerea de soluții cât mai apropiate de cea corectă.

Schema codului sursă al programului este disponibilă pentru *download* la **www.ginfo.ro/concurs/warcraft.shtml**.