Bursele AGORA

A doua etapă a concursului de programare Bursele Agora s-a încheiat. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, clasamentul general după două etape, precum și cele trei probleme ale rundei a patra.

A doua rundă

Problemele celei de-a doua runde au fost mai puțin dificile decât cele propuse pentru rezolvare în cadrul primei etape. Chiar dacă pentru problema ***Funcție*** nici un concurent nu a obținut punctajul maxim, celelalte două probleme au fost rezolvate corect de foarte mulți concurenți.

Vă prezentăm în continuare punctajele primilor 20 clasați în clasamentul acestei runde:

1. Adrian-Nicolae Cârcu, Bistrița	*293*
2. Mugurel-Ionuț Andreica, București	*292*
3. Adrian Balașko, Zalău	*288*
4. Tiberiu Dăneț, București	*272*
4. Silvestru-Cosmin Negruțeri, Bistrița	*272*
6. Kovychev Ruslan Alexanderovich, Rusia	*268*
7. Paul Rusu, Cluj-Napoca	*241*
8. Radu-Andrei Ștefan, Brașov	*238*
9. Andrei Homescu, Târgu-Jiu	*234*
10. Csaba András, Oradea	*232*
11. Radu Berinde, București	*230*
11. Emilian Miron, Bacău	*230*
13. Octavian-Daniel Dumitran, București	*229*
14. Andrei Vancea, Cluj-Napoca	*227*
15. Maximilian Machedon, București	*226*
16. Bogdan-Vasile Hârjoc, Cugir	*213*
17. Bogdan Nicolae, Sibiu	*208*
18. Cornel Mihăilă, Brașov	*201*
19. Victor-Marius Costan, București	*200*
19. Csaba Pătcaș, Oradea	*200*

Runda a cincea

Cele trei probleme ale rundei a cincea sunt *Fibonacci*, *Atlantis* și *Tic-Tac-Toe*.

Prima dintre ele a fost propusă de ***Claudiu Soroiu***, student în anul II la *Universitatea Babeș-Bolyai* din Cluj și redactor al revistei noastre.

A doua problemă 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.

Ultima problemă a fost propusă de ***Mihai Oltean***, doctorand și master al *Universității Babeș-Bolyai* din Cluj, autor a mai multor cărți de specialitate printre care și câteva culegeri de probleme.

Rezolvările celor trei probleme pot fi trimise până la data de ***31 martie 2002***. Rezultatele oficiale ale rundei a cincea vor deveni publice pe data de ***15 aprilie 2002***.

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

La mulți ani!

Dorin-Daniel Dudău, Târgu-Jiu - 1 II 1984
Andrei Markovits, Satu Mare - 2 II 1983
Theodor Drăgoi, Pitești - 2 II 1985
Ion Ionașcu, București - 4 II 1983
Mihai-Silviu Chivu, Brăila - 7 II 1984
Dorin-Cătălin Stoicescu, Buzău - 7 II 1984
Grigore-Mihai Dobra, Baia Mare - 10 II 1985
Mihai Pantelimon, București - 13 II 1984
Ștefan Conțiu, Reghin - 15 II 1984
Ionuț Stan, Aiud - 15 II 1987
Andrei-Daniel Turi, Dr. Turnu Severin - 16 II 1984
Adrian-Nicolae Cârcu, Bistrița - 17 II 1981
Radu Vatavu, Suceava - 17 II 1981
Cătălin Bujdei, Brașov - 17 II 1982
Cornel Mihăilă, Brașov - 17 II 1982
Andrei Homescu, Târgu-Jiu - 19 II 1985
Andrei Blaj, Suceava - 20 II 1985
Ștefan Panțiru, Bicaz-Chei - 22 II 1984
Cătălin Bogda, Buzău - 24 II 1985
Eduard Dumitru, Pitești - 25 II 1985
Ramona Cazacu, Vaslui - 26 II 1985
Marius Anton, Alba Iulia - 29 II 1984

Runda 2, problema 1

MATRICE

La această problemă, 29 dintre participanți au obținut punctajul maxim, deci problema nu a fost foarte dificilă. Metoda folosită de către autor pentru rezolvarea problemei este ***programarea dinamică***.

Evident, determinarea fiecărui element din matrice consumă prea mult timp, deci nu este o soluție viabilă. Rezolvarea propusă se bazează pe o tehnică recursivă.

Vom încerca să determinam, pentru fiecare linie din M_7, numărul de elemente cu valoarea 1 aflate pe linia respectivă. Folosind această informație, linia pe care se află al X-lea element poate fi se găsită imediat (se parcurg liniile matricei M_7, se însumează valorile corespunzătoare primelor K linii și, în cazul în care această sumă este mai mică decât X se trece la linia K+1; dacă suma este mai mare sau egală cu X, atunci al X-lea element se află pe linia K).

Se pune problema determinării numărului de elemente cu valoarea 1 de pe o linie a matricei M_7. Linia respectivă este obținută prin expandarea unei linii a matricei M_6 (de fapt prin această expandare se obțin N linii, linia respectivă fiind una dintre ele). Dacă se cunoaște numărul de elemente cu valoarea 1 de pe linia corespunzătoare a matricei M_6, rezultatul cerut se determină ușor.

Pentru simplificarea formulelor vom numerota liniile și coloanele începând cu 0; în final se vor incrementa cu 1 valorile obținute în urma calculelor.

Așadar, dorim să determinăm numărul elementelor cu valoarea 1 de pe o linie L a matricei M_7. Pentru aceasta, trebuie să cunoaștem numărul elementelor cu valoarea 1 de pe linia [L / N] a matricei M_6. Fiecare dintre aceste elemente se transformă în matricea A și "contribuie" la construirea liniei L din M_7 prin intermediul liniei *rest*[L / N] a matricei A. Similar, fiecare element cu valoarea 0 din M_6 se transformă în matricea B. Așadar, rezultatul cerut este:

numărul de elemente cu valoarea 1 *de pe linia* [L / N] *a matricei* M_6 ***** *numărul de elemente cu valoarea* 1 *de pe linia* rest[L / N] *din matricea* A **+** *numărul de elemente cu valoarea* 0 *de pe linia* [L / N] *a matricei* M_6 ***** *numărul de elemente cu valoarea* 1 *de pe linia* rest[L / N] *din matricea B.*

Pentru a determina numărul de elemente cu valoarea 1 de pe linia [L / N] a matricei M_6 vom aplica recursiv același procedeu, până ajungem la $M_1 = A$. Din nefericire, această metodă duce la repetarea unor calcule de foarte multe ori, prin apelarea funcției recursive cu aceiași parametri.

Pentru a rezolva acest impediment se memorează rezultatele într-o matrice NR de dimensiuni $6 \times n^6$ în care elementul NR_{ij} reprezintă numărul de elemente cu valoarea 1 de pe linia j a matricei M_i.

Au rezolvat corect:

András Csaba, Oradea
Mugurel-Ionuț Andreica, București
Adrian Balașko, Zalău
Radu Berinde, București
Vencel-Istvan Bors, Oradea
Adrian-Nicolae Cârcu, Bistrița
Mihai-Daniel Cilidariu, Botoșani
Victor-Marius Costan, București
Tiberiu Dănet, București
Radu Dondera, București
Octavian-Daniel Dumitran, București
Ștefan Gheorghe, București
Ioan-Claudiu Gruia, București
Andrei Homescu, Târgu-Jiu
Liviu Lalescu, Craiova
Maximilian Machedon, București
Cornel Mihăilă, Brașov
Emilian Miron, Bacău
Silvestru-Cosmin Negrușeri, Bistrița
Mihai Pantelimon, București
Csaba Pătcaș, Oradea
Mihail Popa, București
Claudiu Raicu, Pitești
Constantin Răuțu, Botoșani
Kovychev Ruslan Alexanderovich, Rusia
Paul Rusu, Cluj-Napoca
Radu Andrei Ștefan, Brașov
Victoria Tarasova, Ucraina
Andrei Vancea, Cluj-Napoca

Rezultatele pentru M_7 nu sunt memorate, deoarece fiecare dintre acestea sunt calculate o singură dată. Fără această optimizare, matricea NR nu ar fi putut fi memorată în memoria convențională.

După determinarea liniei pe care se află al X-lea element cu valoarea 1, determinarea coloanei este simplă și poate fi realizată fie într-o manieră similară, fie pur și simplu determinând pe rând fiecare element de pe linia respectivă a matricei M_7. Pe linia respectivă se află cel mult $5^7 = 78125$ elemente, deci numărul iterațiilor nu ar fi foarte mare.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/matrice2.shtml**.

Runda 2, problema 2

VÂNZĂRI

Această problemă a fost rezolvată corect de 18 dintre participanții la a doua rundă a concursului *Bursele Agora*, ceea ce demonstrează faptul că nu este greu de rezolvat.

Ea este doar o particularizare a problemei ***Telefoane mobile*** propusă spre rezolvare celor care au participat la ediția din anul 2001 a *Olimpiadei Internaționale de Informatică*.

În numărul 11/7 (noiembrie 2001) a apărut prezentarea soluției oficiale a problemei de la *IOI*, atât pentru cazul unidimensional (care corespunde problemei ***Vânzări***), cât și în cazul bidimensional (care corespunde problemei *Telefoane mobile*).

Prima idee de rezolvare a problemei ***Vânzări*** este de a păstra un șir *a* ale cărui elemente a_i conțin valoarea vânzărilor către magazinele cuprinse între pozițiile 1 și *i*.

Astfel, suma vânzărilor către magazinele cuprinse între pozițiile *i* și *j* poate fi obținută printr-o simplă diferență de tipul $a_j - a_{i-1}$. Pentru a nu apărea dificultăți în cazul în care trebuie determinată suma vânzărilor către magazinele cuprinse între 1 și *i*, vom inițializa elementul a_0 cu valoarea 0. Vom avea $a_i - a_{1-1} = a_i - a_0 = a_i - 0 = a_i$.

În cazul în care are loc o vânzare către un magazin *i*, atunci trebuie actualizate primele *i* elemente ale vectorului *a*. Așadar, în cazul cel mai defavorabil, ordinul de complexitate al unei actualizări este $O(N)$. Pentru aflarea valorii totale a vânzărilor către magazinele cuprinse între pozițiile *i* și *j*, se efectuează o simplă scădere, deci ordinul de complexitate este $O(1)$. Vom avea maximum *K* actualizări, deci ordinul de complexitate al algoritmului este $O(N \cdot K)$.

Se observă că un algoritm cu acest ordin de complexitate nu se încadrează în limita de timp admisă pentru cazul cel mai defavorabil. Pentru a găsi un algoritm eficient trebuie găsită o modalitate de păstrare a datelor care să permită realizarea mai rapidă a actualizărilor.

Structura de date propusă poartă denumirea de ***arbore indexat binar***. Aceasta a fost prezentată în numărul 11/7 al *GInfo*. Articolul este disponibil pentru *download* la adresa **www.ginfo.ro/revista/11_7/index.shtml**.

Folosind această structură, actualizarea datelor este mult mai rapidă, ordinul de complexitate al algoritmului fiind, în cazul cel mai defavorabil, $O(\log N)$.

Așa cum s-a arătat în articolul amintit anterior, suma corespunzătoare magazinelor cuprinse între pozițiile *i* și *j* nu mai poate fi determinată în timp constant, ordinul de complexitate al acestei operații fiind, în cazul cel mai defavorabil, $O(\log N)$.

Au rezolvat corect:

Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Adrian Balașko, Zalău
Andrei Benea, Focșani
Radu Berinde, București
Adrian-Nicolae Cârcu, Bistrița
Victor-Marius Costan, București
Tiberiu Dăneț, București
Octavian-Daniel Dumitran, București
Ștefăniță Fechete, Suceava
Ștefan Gedo, București
Bogdan-Vasile Hârjoc, Cugir
Andrei Homescu, Târgu-Jiu
Liviu-Laurențiu Iacob, București
Emilian Miron, Bacău
Silvestru-Cosmin Negrușeri, Bistrița
Bogdan Nicolae, Sibiu
Kovychev Ruslan Alexanderovich, Rusia

Se observă că primul algoritm prezentat este mai lent la actualizarea datelor, dar este mai rapid la regăsirea acestora. Din aceste motive, în cazul în care se efectuează puține vânzări și se cer multe statistici, primul algoritm ar fi mai rapid. În cazul cel mai favorabil (când nu se efectuează nici o actualizare) ordinul de complexitate al algoritmului este $O(N)$.

Totuși, dacă se folosește al doilea algoritm, operațiile de actualizare și regăsire a datelor necesită doar un timp logaritmic. Deoarece au loc *K* astfel de operații, ordinul de complexitate este $O(K \cdot \log N)$. Acesta nu depinde de tipul operațiilor (vânzări sau cereri de statistici) pentru că timpul necesar efectuării acestora are același ordin de complexitate. Așadar, ordinul de complexitate al algoritmului nu variază, indiferent de structura datelor de intrare.

În medie, numărul vânzărilor este aproximativ egal cu cel al statisticilor cerute, deci vor fi aproximativ $K / 2$ actualizări. Ordinul de complexitate al primului algoritm va fi, în cazul mediu, $O(K \cdot N)$. Ca urmare, și pentru cazul mediu, al doilea algoritm este mai performant.

Putem trage concluzia că este mai indicată folosirea structurii de *arbore binar indexat* chiar dacă, în anumite situații, folosirea vectorului de sume ar putea duce la obținerea unor performanțe superioare.

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

Runda 2, problema 3

FUNCȚIE

Se știe că rezolvarea ecuațiilor este o problemă dificilă a matematicii. O dovadă în plus în acest sens este faptul că 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 ***Adrian-Nicolae Cârcu*** din *Bistrița* (93 de puncte), ***Mugurel-Ionuț Andreica*** din *București* (92 de puncte) și ***Adrian Balașko*** din *Zalău* (88 de puncte).

Nu se știe dacă există metode sigure pentru determinarea soluțiilor unei ecuații în cazul general. Pentru anumite cazuri particulare există algoritmi siguri care determină exact toate soluțiile. De exemplu, există metode sigure pentru determinarea soluției ecuațiilor polinomiale de gradul unu, doi, trei sau patru. Însă, pentru ecuații polinomiale de grad mai mare s-a demonstrat că nu există nici o metodă exactă pentru determinarea tuturor soluțiilor. De asemenea, precizia cu care este determinată o soluție este o problemă dificil de rezolvat.

Din aceste motive, pentru rezolvarea acestei probleme trebuie folosit un algoritm euristic. Acesta nu va garanta întotdeauna obținerea tuturor soluțiilor, dar folosirea lui în condiții optime ar duce la obținerea unui număr mare de soluții, cu o precizie bună.

Unul dintre cei mai simpli algoritmi aproximativi folosiți pentru rezolvarea unei ecuații este ***metoda înjumătățirii intervalului***. Acest algoritm are dezavantajul că pe intervalul pe care este aplicat trebuie ca ecuația să aibă exact o soluție. În caz contrar, aplicarea lui poate duce la obținerea de rezultate imprevizibile. Dacă există mai multe rădăcini ale ecuației în intervalul dat, atunci ar trebui să împărțim intervalul în subintervale în interiorul cărora există o singură rădăcină, iar apoi să aplicăm metoda înjumătățirii intervalului pentru fiecare dintre aceste subintervale. Această metodă presupune însă împărțirea unui interval în subintervale care conțin exact o soluție a ecuației, ceea ce, din punct de vedere al gradului de dificultate, echivalează cu rezolvarea problemei.

O altă idee de rezolvare este parcurgerea întregului interval cu pasul 10^{-5} și afișarea valorilor care ar fi putut constitui soluția problemei. Această abordare este eficientă doar în cazul în care lungimea intervalului este mică, dar ineficientă pentru intervale de lungime mare.

O altă idee este aplicarea unei ***strategii evolutive***. Se pornește cu o populatie inițială care va evolua doar cu ajutorul operatorului de *mutație*. Fiecare individ va fi selectat pentru mutație, iar fiul rezultat în urma aplicării operatorului va fi comparat direct cu părintele său. Cel mai performant dintre cei doi indivizi (părinte și fiu) va fi inclus în noua populație. Calitatea (*fitness*-ul) unui individ x se stabilește în funcție de distanța la care se află $f(x)$ față de axa Ox. Cu alte cuvinte, calitatea unui individ x este chiar $|f(x)|$. Prin mutație, se înțelege adăugarea sau scăderea unei valori mici la (de la) valoarea individului. În final sunt afișate soluțiile aflate la o distanță mai mare decât precizia dată unele de altele. Această soluție are dezavantajul unei execuții lente cauzată de faptul că unii indivizi ajung mai repede decât alții la soluție și, cu toate acestea, ei vor fi supuși din nou mutației. Eliminarea celor care au ajuns deja la soluții acceptabile ar însemna efectuarea unor operații suplimentare, deci o creștere a complexității algoritmului.

Din aceste motive, se va aplica o variantă modificată a algoritmului de mai sus. Se pornește cu o populație inițială generată aleator. Fiecare individ este selectat și asupra lui se aplică operatorul de mutație până în momentul în care individul ajunge să fie soluție a ecuației. Pasul de mutație este 10^{-5}. Această valoare este exact precizia cu care trebuie determinate soluțiile. Din nefericire, această valoare este destul de mică pentru unele cazuri.

Din acest motiv vom modifica din nou algoritmul. Pasul de mutație nu va mai fi fix, ci va descrește treptat. La început, pasul de mutație va fi mai mare (de exemplu 10^{-3}). Acest pas de mutație va face ca soluțiile să efectueze salturi mai mari spre soluție. În momentul în care nu se vor mai putea efectua mutații cu acest pas, acesta va fi micșorat (noua valoare ar putea fi 10^{-4}) și algoritmul va continua. Acest tip de strategie de căutare se numește ***călire simulată***. Pasul de mutație este scăzut pe măsura ce crește numărul generațiilor.

Nici acest algoritm nu este sigur. Datorită faptului că pasul de mutație este, inițial, mai mare decât precizia cu care trebuie determinate soluțiile, s-ar putea ca unele soluții să fie pierdute. O îmbunătățire ar putea fi setarea pasului inițial al mutației în funcție de lungimea intervalului pe care se caută soluțiile ecuației. Astfel, pentru lungimi mari ale intervalului se va începe cu un pas mare de mutație, iar pentru lungimi mici ale intervalului se va începe cu pași mici de mutație. Pentru mărirea șanselor de a găsi toate soluțiile vom genera o populație inițială cât mai mare.

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