Olimpiada JUDEȚEANĂ de INFORMATICĂ 2005

În cadrul acestui articol vă vom prezenta soluțiile celor 14 probleme propuse spre rezolvare la faza județeană a olimpiadei de informatică. Enunțurile acestor probleme au fost publicate în numărul anterior al revistei noastre.

Clasa a V-a

P050201: Ultima cifră

Pentru fiecare număr *i* cuprins între 1 și *n* vom determina ultima cifră a valorii i^i, iar apoi vom determina ultima cifră a sumei cifrelor obținute.

Ultima cifră a unei valori i^i este obținută astfel:

- se determină ultima cifră a valorii *i* (această este dată de restul împărțirii la 10 a valorii *i*);
- se pornește de la valoarea 1 și, de *i* ori, valoarea curentă este înmulțită cu *i*, dar se păstrează doar ultima cifră (prin determinarea restului împărțirii la 10);

Pe măsură ce determinăm aceste ultime cifre, le vom însuma, dar și de această dată vom păstra, la fiecare pas, doar ultima cifră a sumei (tot prin determinarea restului împărțirii la 10).

Prezentăm în continuare versiunea în pseudocod a algoritmului descris:

```
algoritm UltimaCifră
  citește n
  rezultat ← 0
  pentru i ← 1, n execută
    cifra ← 1
    pentru i ← 1, n execută
      cifra ← rest[cifra * i
                        / 10]
    sfârșit pentru
    rezultat ← rest[(rezultat +
                    cifra) / 10]
  sfârșit pentru
  scrie rezultat
sfârșit algoritm
```

P050202: Mulțimi

Pentru a determina intersecția mulțimilor date va trebui să luăm în considerare doar extremitățile acestora. Mulțimea de intersecție va fi și ea descrisă prin extremitățile ei.

Intersecția mulțimilor poate fi obținută prin parcurgerea mai multor pași. Vom porni de la prima mulțime și, la fiecare pas *i*, vom determina intersecția dintre mulțimea curentă și cea de-a *i*-a mulțime.

Intersecția a două mulțimi $\{a, ..., b\}$ și $\{c, ..., d\}$ va fi întotdeauna:

$$\{\max(a, c), ..., \min(b, d)\}$$

Practic, extremitatea stângă a intersecției va fi dată de cea mai mare dintre extremitățile stângi ale mulțimilor date, iar extremitatea dreaptă a intersecției va fi dată de cea mai mică dintre extremitățile drepte ale mulțimilor date.

În cazul în care maximul extremităților stângi este mai mare decât minimul extremităților drepte, mulțimea de intersecție va fi vidă.

Varianta în pseudocod a algoritmului este prezentată în continuare.

```
algoritm Intersecție
  scrie "n="
  citește n
  // se citește prima mulțime
  citește min, max
  // se citesc celelalte mulțimi și
  // se determină maximul extre-
  // mităților stângi și minimul
  // extremităților drepte
  pentru i ← 2, n execută
    citește st, dr
    dacă max < st atunci
      max ← st
    sfârșit dacă
    dacă min > dr atunci
      min ← dr
    sfârșit dacă
  sfârșit pentru
  // se scrie intersecția
  dacă max > min atunci
    scrie "Mulțime vidă"
  altfel
    pentru i ← max, min execută
      scrie i
    sfârșit pentru
  sfârșit dacă
sfârșit algoritm
```

Clasa a VI-a

P050203: Numere

După citirea valorilor *a* și *b*, se determină reprezentarea binară a acestora și se creează doi vectori care păstrează cifrele corespnzătoare celor două reprezentări.

După construirea vectorilor, încep transformările. Se verifică dacă primul element al primului vector este egal cu ultimul element al celui deal doilea.

Dacă cele două elemente sunt egale, atunci aceste elemente sunt eliminate din cei doi vectori. Transformările continuă până în momentul în care primul element al primului vector este diferit de ultimul element al celui de-al doilea.

În continuare, pe baza elementelor rămase în cei doi vectori, se determină valorile *c* și *d* și se afișează valoarea $c + d$.

P050204: Șir

Datorită faptului că elementele tabloului conțin doar cel mult două cifre, acesta va avea cel mult 100 de elemente.

Vom utiliza o variabilă care va indica numărul elementelor generate; aceasta va fi inițializată cu 0.

Primul element al șirului este obținut prin împărțirea la 10 a restului împărțirii la 1000 a numărului $k \cdot k$.

În continuare, fiecare element va fi dat de câtul împărțirii la 10 a restului împărțirii la 1000 a pătratului elementului anterior.

Practic restul împățirii la 1000 duce la obținerea ultimelor trei cifre ale unui număr. Dacă acest rest este împărțit la 10, ultima cifră este eliminată, deci rezultatul este un număr format din cifra sutelor, urmată de cifra zecilor numărului inițial.

După generarea unui nou element, se verifică dacă acesta a mai fost generat prin parcurgerea elementelor generate anterior.

În cazul în care se identifică un astfel de număr procesul se oprește.

În caz contrar, se adaugă la sfârșitul vectorului elementul nou generat și se incrementează variabila care conține numărul elementelor vectorului.

După determinarea elementelor vectorului, urmează afișarea acestora.

În continuare trebuie să realizăm sortarea în funcție de cea mai din stânga cifră. Pentru aceasta, vom crea un alt vector care va conține cea mai din stânga cifră.

Un element al acestui al doilea vector va avea valoarea elementului corespunzător din primul vector dacă acest element este mai mic decât 10 sau valoarea obținută prin împărțirea la 10 a elementului corespunzător din primul vector dacă acest element este cel puțin egal cu 10.

Urmează ordonarea elementelor acestui al doilea vector, operație care poate fi realizată folosind diverse metode.

În momentul în care sunt interschimbate două elemente în al doilea vector, vor fi interschimbate și elementele corespunzătoare ale primului vector.

În final, se vor afișa elementele primului vector, în ordinea obținută după această ordonare.

Clasa a VII-a

P050205: OCR

Coordonatele centrului de greutate vor fi determinate în doi pași. Vom determina mai întâi linia pe care se va afla centrul de greutate și apoi coloana.

Pentru a determina linia pe care se află centrul de greutate vom determina, mai întâi, suma elementelor de pe fiecare linie.

Ulterior, pentru fiecare linie, putem determina foarte ușor suma elementelor de pe liniile anterioare, precum și suma elementelor de pe liniile care urmează.

Pentru aceasta vom parcurge pe rând liniile și vom considera, inițial, că suma *A* a elementelor de deasupra liniei curente este 0, iar suma *B* a elementelor de sub linie curentă este dată de suma tuturor elementelor matricei (acaestă sumă poate fi determinată simultan cu determinarea sumelor corespunzătoare liniilor) din care se scade suma elementelor de pe prima linie.

La fiecare pas, suma *A* va crește cu suma elementelor de pe linia anterioară, iar suma *B* va scădea cu suma elementelor de pe linia curentă.

Dintre liniile parcurse, va fi aleasă cea pentru care diferența dintre sumelor *A* și *B* este minimă. Astfel am reușit să determinăm linia pe care se află centrul de greutate al imaginii date.

Aplicăm un procedeu similar pentru a determina coloana pe care se află centrul de greutate. Trebuie menționat faptul că sumele elementelor de pe coloane pot fi determinate simultan cu sumelor elementelor de pe linii.

După efectuarea acestor doi pași cunoaștem ambele coordonate ale centrului de greutate al imaginii considerate.

P050206: Tabel

Este evident faptul că, în cazul în care de pe o linie lipsește un singur număr, atunci valoarea sa poate fi determinată.

Dacă elementul care lipsește este ultimul al linie, atunci valoarea sa va fi dată de suma celorlalte elemente.

În caz contrar, valoarea sa va fi dată de diferența dintre valoarea ultimului elemente și suma celorlalte elemente.

Evident, același procedeu poate fi aplicat și în cazul în care lipsește un singur element de pe o coloană.

De asemenea, este evident faptul că, dacă de pe o linie sau de pe o coloană lipsesc două elemente, atunci valorile acestor elemente nu pot fi determinate.

Inițial vom considera că nici un element nu este cunoscut, deci numărul elementelor necunoscute de pe fiecare linie va fi *m*, iar numărul elementelor necunoscute de pe fiecare coloană va fi *n*.

Pe măsură ce sunt citite informațiile despre valorile cunoscute, pentru fiecare astfel de valoare, numărul elementelelor necunoscute de pe linia și coloana corespunzătoare va scădea cu 1.

După citirea datelor, vom ști cu exactitate câte elemente lipsesc de pe fiecare linie și de pe fiecare coloană.

Datorită faptului că știm că tabelul va putea fi reconstituit întotdeauna știm că există cel puțin o linie sau o coloană de pe care lipsește un singur element.

Practic, la fiecare pas, vom determina o linie sau o coloană de pe care lipsește un singur element.

După reconstituirea elementului care lipsește de pe linia sau coloana aleasă, va scădea cu 1 numărul elementelor lipsă de pe linia și coloana corespunzătoare.

Din nou, datorită faptului că tabelul poate fi reconstituit, putem fi siguri că vom avea din nou o linie sau o coloană de pe care lipsește un singur element sau am reconstituit toate elementele.

Așadar, ne vom opri în momentul în care nu mai există nici o coloană și nici o linie de pe care lipsesc elemente. În acest moment vom putea afișa rezultatul.

Clasa a VIII-a

P050207: Muzică

Vom efectua succesiv operațiile descrise în enunț până în momentul în care obținem pentru o valoare care a mai fost obținută anterior.

Datorită faptului că la fiecare pas efectuăm aceleași operații, începând din acest moment secvența se va repeta cu siguranță.

Dacă notăm cu i poziția valorii care se repetă și cu j poziția pe care a apărut prima dată, atunci vom scrie toate valorile obținute până la poziția i - 1, iar lungimea refrenului melodiei va fi i - j.

P050208: Volei

Vom ordona pentru început băieții în ordine crescătoare, în funcție de înălțimea lor.

După ordonare, vom parcurge în ordine șirul ordonat și, pentru fiecare băiat, vom verifica dacă există două fete vecine care sunt ambele mai scunde decât el și între care nu joacă deja un alt băiat.

În final vom obține numărul maxim al băieților care vor juca, precum și pozițiile lor în cerc.

Clasa a IX-a

P050209: Numere

Pentru rezolvarea acestei problemă este suficient să utilizăm un șir de biți care vor indica dacă un număr apare sau nu în fișier.

Vom avea nevoie întotdeauna de cel mult 250.000 de biți, deci 31250 de octeți.

Inițial toți biții vor avea valoarea 0, iar pe măsură ce sunt citite numerele care formează matricea, valoarea bitului corespunzător unui număr citit devine 1.

Șirul de biți va conține în final o secvență de zerouri care va reprezenta soluția problemei.

Există și posibilitatea de a utiliza *heap*-ul pentru a păstra 250.000 de valori booleene sau întregi care vor permite și ele determinarea secvenței care lipsește.

P050210: MaxD

Pentru fiecare număr în intervalul dat, vom determina numărul divizorilor acestuia.

Pentru aceasta vom determina descompunerea în factori primi a fiecărui număr.

Pentru un număr n, descompunerea în factori primi are forma:

$$n = f_1^{e_1} \cdot f_2^{e_2} \cdot f_3^{e_3} \cdot \ldots \cdot f_k^{e_k},$$

iar numărul divizorilor va fi dat de valoarea $(e_1 + 1) \cdot (e_2 + 1) \cdot (e_3 + 1) \cdot \ldots \cdot (e_k + 1)$.

Pentru fiecare valoare x din intervalul $[a, b]$ se determină numărul său de divizori, se reține acea valoare care are număr maxim de divizori; în cazul identificării unor valori cu același număr maxim de divizori, se incrementează contorul care reține numărul acestor valori.

În cazul identificării unei valori cu număr mai mare de divizori, valoarea contorului devine 1 și se actualizează variabila care conține numărul cu cei mai mulți divizori.

În final se va afișa prima valoare x care are cel mai mare număr de divizori, precum și valoarea contorului.

Clasa a X-a

P050211: Lăcusta

Pentru rezolvarea acestei probleme vom utiliza metoda programării dinamice.

Vom nota prin A matricea dată și vom construi o matrice B ale cărei elemente b_{ij} vor conține sumele minime necesare pentru a ajunge în celula (i, j) pornind din celula $(i - 1, j)$.

Vom completa inițial elementele de pe a doua linie a matricei B.

Valoarea b_{21} va fi ∞ deoarece în această celulă nu se poate ajunge.

Valoarea celorlalte elemente b_{2i} vor fi calculate pe baza formulei:

$$b_{2i} = a_{11} + a_{1i} + a_{2i}.$$

Pentru celelalte linii, valorile b_{ij} vor fi calculate pe baza formulei:

$$b_{ij} = a_{ij} + a_{i-1,j} + \mathbf{min}(b_{i-1,k}),$$

unde k variază între 1 și n.

Evident, relația nu este valabilă pentru elementul de pe coloana k care corespunde minmului, deoarece nu se poate coborî direct, ci trebuie efectuat un salt orizontal. În această situație vom alege al doilea minim de pe linia anterioară.

În final alegem minimul valorilor de pe ultima linie a matricei B (fără a lua în considerare elementul de pe ultima coloană a acestei linii) la care adaugăm valoarea a_{mn}.

P050212: Scara

Pentru rezolvarea acestei probleme vom utiliza metoda *backtracking*.

Pentru aceasta vom construi un vector a care va conține înălțimile scărilor, precum și un vector *sol* care va conține, în fiecare moment, cea mai bună soluție.

De asemenea, vom construi un vector b care va conține efortul care trebuie depus pentru a urca scările (al i-lea element al acestui vector indică efortul necesar pentru a urca primele i scări).

Pentru a ne asigura că înălțimile sunt distincte vom păstra un vector de valori logice în care se vor marca înălțimile folosite în fiecare moment.

De asemenea, la fiecare pas, va fi păstrată suma totală a înălțimilor scărilor consruite la pașii anteriori.

În timpul algoritmului, trebuie să ne asigurăm că valorile vectorului *b* sunt cuprinse între 1 și *M* și sunt distincte. De asemenea, suma lor trebuie să fie egală cu *H*.

La fiecare pas *i*, valoarea b_i va fi calculată pe baza formulei:

$$b_i = \min(b_{i-1} + a_i, \min(b_j + p + (a_{j+1} + a_{j+2} + ... + a_i) / (i - j)))$$

unde *j* variază între 1 și *i* - 2, iar sumele de forma $a_{j+1} + a_{j+2} + ... + a_i$ sunt luate în considerare numai dacă sunt cel mult egale cu *M*.

Dacă am reușit să ajungem la ultimul pas, vom verifica dacă soluția curentă este mai bună decât cea considerată anterior a fi cea mai bună.

În acest caz atât vectorul *sol*, precum și efortul total minim, sunt actualizate.

În final, vom afișa efortul minim obținut, precum și elementele vectorului *sol*.

Clasele a XI-a și a XII-a

P050213: Lanț

Pentru început vom determina cuvintele distincte din textul dat și le vom păstra într-un vector *c*.

Numărul elementelor acestui vector va fi notat prin *n*, iar cuvintele vectorului *c* sunt numerotate de la 0 la *n* - 1 în ordinea în care apar pentru prima dată în text.

Vom construi un graf orientat ale cărui noduri reprezintă cuvintele distincte (elementele vectorului construit *c*).

Va exista un arc de la un nod *i* la un nod *j* dacă numărul operațiilor `insert` și `delete` necesare transformării cuvântului c_i în cuvântul c_j este cel mult egal cu *k*.

În plus, un astfel de arc poate exista numai dacă avem $i < j$; datorită acestui fapt graful construit este întotdeauna aciclic.

Pentru a determina dacă un graf există sau nu va trebui să rezolvăm problema determinării costului de transformare a unui cuvânt *x* într-un cuvânt *y*.

Aceasta este o problemă clasică pentru a cărei rezolvare este aplicată metoda programării dinamice.

Vom construi o matrice *D* ale cărei elemente d_{ij} vor reprezenta costurile transformării cuvântului obținut prin eliminarea primelor *i* - 2 litere ale cuvântului *x* în cuvântul obținut prin eliminarea primelor *j* - 2 litere ale cuvântului *x*.

Valorile *i* variază între 0 și *p* - 1, iar valorile *j* variază între 0 și *q* - 1, unde *p* este numărul literelor din care este format cuvântul *x*, iar *q* este numărul literelor din care este format cuvântul *y*.

Vom completa ultima linie a matricei *D* folosind formula $d_{nj} = m - j$ și ultima coloană folosind formula $d_{mi} = n - i$.

Celelalte elemente ale matricei vor fi determinate pe baza formulei:

$d_{ij} = \min(e_{i+1,j+1}, d_{i+1,j} + 1, d_{i,j+1} + 1)$, unde e_{ij} este d_{ij} dacă $x_i = y_j$ sau ∞ în caz contrar.

Costul transformării va fi dat de valoarea d_{00}. Dacă aceasta este cel mult egală cu *k*, atunci arcul corespunzător va fi creat.

Numărul lanțurilor căutate poate fi determinat foarte simplu. Vom construi un vector *s* ale cărui valori s_i vor conține numărul lanțurilor care încep cu al *i*-lea cuvânt din vectorul *c*.

Valorile corespunzătoare tuturor nodurilor terminale vor fi 1.

Pentru celelalte noduri, valoarea s_i este calculată însumând toate valorile s_j pentru care există un arc de la *i* la *j*.

Soluția problemei va fi dată de maximul valorilor s_i.

P050214: Scara

Pentru rezolvarea acestei probleme vom construi un graf orientat cu *n* noduri (corespunzătoare celor *n* trepte).

Costurile arcelor acestui graf trebuie să respecte următoarele condiții:

- costul unui arc care unește două noduri trebuie să fie cu atât mai mic, cu cât treptele corespunzătoare acestora sunt mai îndepărtate;
- costurile arcelor corespunzătoare salturilor efectuate după ce se bea băutura energizantă trebuie să fie mai mare decât costul salturilor efectuate după ce se bea apă între aceleași noduri, dar acest cost să nu depășească costul saltului între două noduri mai apropiate.

Există o mulțime de posibilități de a alege aceste costuri. Evident, vor exista arce doar de la nodurile cu numere de ordine mai mici spre noduri cu numere de ordine mai mari.

Un exemplu de alegere a acestor costuri este (matricea costurilor este notată prin *C*):

- $c_{i,i+1} = 999000$;
- $c_{ij} = 1000000 - (j - i)$ pentru arcele corespunzătoare salturilor care pot fi efectuate după ce se bea apă.
- $c_{ij} = 1000000 - (j - i) + q$ pentru arcele corespunzătoare salturilor care pot fi efectuate după ce se bea băutura energizantă.

După construirea acestui graf poate fi aplicat algoritmului lui *Dijkstra* pentru determinarea soluției.

Notă...

Descrierile prezentate în cadrul acestui articol sunt bazate pe soluțiile oficiale puse la dispoziție de către autorii problemelor. Prezentările au fost modificate de către membrii redacției GInfo pentru a îmbunătăți claritatea.