AMINTIRI

Prezentăm acum câteva detalii referitoare la modul în care pot fi rezolvate problemele propuse în numărul 12/7 (noiembrie 2002) al Gazetei de Informatică.

P070201: Influențe

O modalitate simplă de verificare a existenței a două persoane care se pot influența reciproc (fără a folosi cunoștințe avansate de programare) se bazează pe construirea unei liste (pentru fiecare persoană) care conține numerele de ordine ale persoanelor influențate direct sau indirect de o anumită persoană.

Putem spune că va exista o influență reciprocă în momentul în care o persoană se influențează indirect pe ea însăși (numărul ei de ordine apare în propria listă).

Pentru a construi mai ușor aceste liste le vom păstra sub forma unor mulțimi deoarece o persoană poate apărea o singură dată într-o astfel de listă.

Pentru construirea listelor vom avea în vedere faptul că o persoană *i* care influențează o persoană *j* va influența indirect toate persoanele influențate de *j*.

Vom parcurge lista perechilor (*i*, *j*) cu semnificația "*persoana i este influențată de persoana j*" și pentru fiecare astfel de pereche vom adăuga în lista (mulțimea) corespunzătoare persoanei *j* toate persoanele din lista (mulțimea) corespunzătoare persoanei *i*. Se observă că această operație constă într-o simplă reuniune de mulțimi.

După parcurgerea listei vom repeta acest pas până în momentul în care nu vom mai efectua nici o modificare a mulțimilor (nici o reuniune nu duce la mărirea numărului de elemente a vreunei mulțimi).

În acest moment vom cunoaște numerele de ordine ale tuturor persoanelor influențate de fiecare persoană *i*.

Dacă în mulțimea corespunzătoare unei persoane *i* apare și numărul *i*, atunci cu siguranță există o influență reciprocă și algoritmul se încheie.

Dacă nu a fost detectată nici o influență reciprocă, atunci cea mai influentă persoană este cea căreia îi corespunde mulțimea cu cel mai mare număr de elemente.

În continuare vom descrie modul în care poate fi determinat cel mai lung lanț al slăbiciunilor. Pentru început trebuie să observăm faptul că este posibil ca acest lanț să nu înceapă cu cea mai influentă persoană.

Pentru fiecare persoană vom construi cel mai lung lanț care începe cu acea persoană și apoi vom alege cel mai lung lanț dintre lanțurile construite.

Inițial, vom ști doar că lanțurile corespunzătoare persoanelor care nu influențează nici o altă persoană au lungimea 1. Știm cu siguranță că există astfel de persoane deoarece nu există influențe reciproce.

În continuare, la fiecare pas, dacă observăm că o persoană *i* influențează o persoană *j* al cărei lanț a fost determinat, atunci putem adăuga persoana *i* în fața acestui lanț, lungimea lui crescând cu o unitate. Evident, s-ar putea ca pentru persoana *i* să existe deja un lanț mai lung sau egal. În această situație, noul lanț este ignorat.

Vom parcurge de mai multe ori lista influențelor, până în momentul în care o parcurgere care nu duce la determinarea nici unui lanț care nu este ignorat.

După terminarea parcurgerilor vom alege cel mai lung lanț și îl vom afișa.

Observație

Există multe alte soluții mai performante, dar care necesită cunoștințe de algoritmică ce depășesc nivelul clasei a IX-a.

De exemplu, se poate construi un graf al influențelor; graful este orientat, fiecare nod reprezintă o persoană, iar pentru fiecare pereche vom avea câte un arc. Arcele vor fi de la nodul corespunzător persoanei care influențează la nodul corespunzător unei persoane pe care o influențează direct.

Vor exista influențe reciproce dacă graful este ciclic, cea mai influentă persoană este cea corespunzătoare nodului din care se poate ajunge în cele mai multe noduri urmând arcele grafului, iar cel mai lung lanț al slăbiciunilor este dat de cel mai lung traseu dintre două noduri ale grafului.

P070202: Păsărească

Dacă trebuie să traducem din română în păsărească, atunci vom parcurge cuvintele caracter cu caracter. În cazul în care caracterul curent nu este o vocală, atunci acesta este scris în fișierul de ieșire. În caz contrar (caracterul curent este o vocală *v*), în fișierul de ieșire vor fi scrise trei caractere (vocala *v*, caracterul `'p'` și din nou vocala *v*).

Dacă trebuie să traducem din păsărească în română, vom parcurge și în acest caz cuvintele caracter cu caracter.

În cazul în care caracterul curent nu este o vocală, atunci acesta este scris în fișierul de ieșire. În cazul în care caracterul curent este o vocală *v*, atunci aceasta este scrisă în fișierul de ieșire, iar următoarele două caractere sunt ignorate (acestea sunt, obligatoriu, litera `'p'` și din nou vocala *v*).

P070203: Piramida

Vom determina elementele șirului până în momentul în care vom ajunge sau vom trece de poziția *q*.

Pentru aceasta vom păstra întotdeauna poziția în șir la care începe ultima linie a piramidei și vom determina următoarele elemente (pe baza regulii din enunț). Vom repeta procedeul până în momentul în care este determinat elementul de pe poziția *q*.

În continuare vom considera numărul în baza 4 format din cifrele de pe pozițiile cuprinse între *p* și *q* și îl vom converti în baza 10.

P070204: Numere

Pentru a verifica dacă un număr are aspect de munte vom parcurge cifrele sale atât timp cât ele se află în ordine crescătoare. Dacă următoarea cifră este mai mică decât cifra curentă, atunci vom continua parcurgerea cifrelor atât timp cât ele se află în ordine descrescătoare. Dacă ajungem la ultima cifră și aceasta respectă condiția cerută (este mai mică decât predecesoarea sa), atunci numărul are aspect de munte.

Verificarea faptului că un număr are aspect de vale se realizează similar, singura diferență fiind că cifrele cu care începe numărul trebuie să fie în ordine descrescătoare, iar celelalte în ordine crescătoare.

Evident, la începutul numărului trebuie să fie cel puțin două cifre în ordine crescătoare/descrescătoare, iar la sfârșit cel puțin două în ordine descrescătoare/crescătoare. Vom construi un șir care conține numerele cu aspect de munte din intervalul [*a*, *b*] și unul care conține numerele cu aspect de vale din acest interval.

În continuare vom considera toate perechile de numere din același șir și vom verifica dacă perechea curentă respectă condiția din enunț (oglinditul sumei numerelor este suma oglinditelor lor).

În momentul în care detectăm prima astfel de pereche, numerele vor fi scrise în fișierul de ieșire și vom opri execuția programului.

P070205: Cel mai mare divizor comun

Pentru a determina cel mai mare divizor comun a două numere vom folosi algoritmul lui *Euclid*.

Vom determina inițial cel mai mare divizor comun d_2 al primelor două numere din șir (a_1 și a_2), apoi vom determina cel mai mare divizor comun d_3 al numerelor d_2 și a_3. Vom continua determinarea celor mai mari divizori comuni d_i ai numerelor d_{i-1} și a_i.

Se observă că d_i este cel mai mare divizor comun al primelor *i* numere din șir.

Cel mai mare divizor comun al celor *n* numere din șir va fi d_n.

Pentru a determina divizorii primi ai acestui număr vom aplica următorul algoritm:

```
i ← 2
cât timp i ≤ d_n execută
  dacă rest[d_n / i] = 0 atunci
    scrie i
    cât timp rest[d_n / i] = 0 execută
      d_n ← d_n / i
    sfârşit cât timp
  sfârşit dacă
  i ← i + 1
sfârşit cât timp
```

Algoritmul anterior determină cel mai mic divizor *i* prim al valorii d_n și apoi împarte valoarea d_n la *i* până în momentul în care *i* nu mai este divizor al lui d_n. Astfel, noua valoare d_n nu va avea nici un divizor mai mic decât *i*.

În continuare se va determina următorul divizor (mai mare decât *i*) al noii valori. Acesta va fi cu siguranță prim deoarece dacă ar fi de forma $p \cdot q$ atunci *p* și *q* ar fi fost determinați ca divizori anterior și valoarea d_n ar fi fost împărțită până în momentul în care nu i-ar mai fi avut ca divizori.

Se continuă același procedeu până în momentul în care valoarea divizorului testat depășește valoarea curentă d_n.

Evident, există și posibilitatea de a testa toate numerele cuprinse între 2 și valoarea inițială d_n, dar algoritmul este mult mai lent. Pentru fiecare divizor detectat al lui d_n se verifică dacă acesta este prim.

Acest algoritm poate fi îmbunătățit dacă se consideră doar numerele din intervalul cuprins între 1 și rădăcina pătrată a valorii d_n.

În acest caz, pentru fiecare divizor *i* din interval se va testa atât primalitatea sa, cât și primalitatea valorii d_n / i.

Algoritmul va funcționa corect chiar dacă d_n este un număr prim deoarece d_n va fi detectat ca divizor chiar la început (la primul pas) când se va testa primalitatea numărului 1 și cea a numărului $d_n / 1 = d_n$.

P070206: Numere deosebite

Pentru a verifica dacă un număr *n* este deosebit vom parcurge numerele *n* - 1, *n* - 2, etc. Pentru fiecare astfel de număr vom verifica dacă prin operația de adunare dintre număr și suma cifrelor sale se obține exact valoarea *n*.

Dacă valoarea *n* nu se obține când ajungem la numărul 0, atunci numărul *n* nu este deosebit.

Datorită faptului că valoarea *n* este cuprinsă între 1 și 1000, suma cifrelor numerelor considerate este cel mult 26 (pentru $m = 999$ avem $m + S(m) = 999 + 27 = 1026 > 1000$; pentru $m = 899$ avem $m + S(m) = 899 + 26 = 925 < 1000$). Așadar putem lua în considerare numerele cuprinse între **max**(*n* - 26, 0) și *n*. Evident, nu are rost să testăm și numere negative.