Bursele Agora

Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la etapele #05-#08 ale ediției 2002/2003 a concursului de programare Bursele Agora.

P080215: Perechi

Se consideră un număr natural n și trebuie formate cât mai multe perechi din care trebuie să facă parte numere naturale cuprinse între 1 și $2 \cdot n$ astfel încât suma pătratelor celor două numere din oricare dintre perechi să fie număr prim. Fiecare număr poate face parte din cel mult o pereche.

Date de intrare

Fișierul de intrare `PAIRS.IN` conține o singură linie pe care se află valoarea n.

Date de ieșire

Prima linie a fișierului de ieșire `PAIRS.OUT` va conține numărul k al perechilor formate. Fiecare dintre următoarele k linii va conține câte două numere cuprinse între 1 și $2 \cdot n$, astfel încât suma pătratelor celor două numere este număr prim. Perechile pot fi scrise în orice ordine.

Restricție

- $1 \le n \le 2500$.

Exemplu

`PAIRS.IN`

```
7
```

`PAIRS.OUT`

```
7
1 4
2 3
5 6
7 8
9 10
11 14
12 13
```

Modalitatea de acordare a punctajului

Dacă pentru un test se pot obține cel mult X puncte, punctajul acordat va fi $X / 2^{n-k}$, unde n este numărul din fișierul de intrare, iar k este numărul perechilor formate. Valoarea va fi rotunjită la două zecimale exacte. Dacă nu se obține nici o pereche sau există erori în fișierul de ieșire nu se va acorda nici un punct.

Timp de execuție: 1 secundă/test

P080216: Numere prime în 3D

Se consideră un tablou tridimensional A de dimensiune $M \times N \times P$ ale cărui elemente trebuie completate astfel încât orice element are valoarea egală cu numărul elementelor vecine care sunt prime.

Două elemente A_{ijk} și $A_{i'j'k'}$ sunt vecine dacă și numai dacă $\max(|i - i'|, |j - j'|, |k - k'|) = 1$, $1 \le i, i' \le M$, $1 \le j, j' \le N$, $1 \le k, k' \le P$.

Valoarea `0` poate fi folosită ca ***joker***, în sensul că unui element i se poate atribui această valoare indiferent de numărul elementelor vecine care sunt prime.

Punctajul unui tablou este dat de suma elementelor sale.

Va trebui să completați tabloul astfel încât punctajul său să fie cât mai mare posibil.

Date de intrare

Fișierul de intrare `3D.IN` conține valorile M, N și P, separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire `3D.OUT` va avea $M \times N \times P$ linii. Fiecare trebuie să conțină câte un număr reprezentând valoarea unui element al tabloului. Elementele tabloului vor fi ordonate crescător în funcție de primul index i, pentru valori egale ale primului index în funcție de al doilea index j, iar pentru valori egale ale primilor doi indecși în funcție de al treilea index k. De exemplu, pentru $M = N = P = 2$, ordinea va fi A_{111}, A_{112}, A_{121}, A_{122}, A_{211}, A_{212}, A_{221}, A_{222}.

Restricții

- $1 \le M, N, P \le 30$.

Exemplu

`3D.IN`

```
2 2 2
```

3D.OUT

```
7
7
7
7
7
7
7
7
```

Modalitatea de acordare a punctajului

Vom considera că, pentru fiecare test, se vor putea obține cel mult *X* puncte.

Concurenții care vor obține cea mai mare valoare *PtMax* pentru punctajul tabloului vor primi *X* puncte pentru testul respectiv.

Ceilalți concurenți, care au completat corect tabloul *A* și au obținut un punctaj *Pt*, vor obține $X \cdot Pt / PtMax$ puncte pentru testul respectiv. Această valoare va fi aproximată cu două zecimale exacte.

Punctajul final va fi obținut prin adunarea punctajelor de la fiecare test și rotunjirea acestuia la cel mai apropiat număr întreg.

Dacă tabloul nu este completat corect, concurenții nu vor primi nici un punct pentru testul respectiv.

De exemplu, dacă pentru un test se pot obține cel mult **5** puncte, cel mai bun rezultat obținut de un concurent constă într-un tablou cu punctajul **56**, iar un alt concurent obține o soluție cu punctajul **25**, atunci, pentru testul respectiv, punctajul concurentului va fi de $5 \cdot 25 / 56 = 2.23$ puncte.

Timp de execuție: 1 secundă/test

P080217: Factorizare

Se consideră un număr natural *N* și trebuie determinată o pereche de numere naturale strict mai mari decât 1, al căror produs să fie egal cu *N*.

Date de intrare

La această problemă fișierele de intrare sunt deschise. În pagina http://www.ginfo.ro/concurs/runda07/index.shtml sunt disponibile *link*-uri spre douăzeci de fișiere de intrare.

Numele acestora au forma **FACTORXX.IN**, unde **XX** ia valori între **01** și **20** și reprezintă numărul testului. Fiecare dintre cele 20 de fișiere de intrare conține o singură linie pe care se află valoarea *N*.

Date de ieșire

Pentru această problemă nu va trebui să trimiteți un program care să o rezolve, ci doar cele 20 de fișiere de ieșire corecte. Acestea vor fi denumite **FACTORXX.OUT**, unde **XX** ia valori între **01** și **20** și reprezintă numărul testului.

Prima linie a fișierului de ieșire va conține textul `FACTORS - TEST #XX`, unde `XX` reprezintă numărul testului.

Următoarele două linii vor conține câte un număr strict mai mare decât **1**. Produsul acestor două numere trebuie să fie egal cu numărul *N* din fișierul de intrare corespunzător.

Dacă nu există două astfel de numere înseamnă că valoarea *N* este un număr prim, iar fișierul de ieșire va conține doar două linii; pe prima se va afla textul `FACTORS - TEST #XX`, unde `XX` reprezintă numărul testului, iar pe a doua se va afla textul `PRIME NUMBER`.

Exemplu

Vom considera că acest exemplu reprezintă testul **00**.

FACTOR00.IN

```
221
```

FACTOR00.OUT

```
FACTORS - TEST #00
13
17
```

Modalitatea de acordare a punctajului

Pentru fiecare fișier de ieșire corect veți obține un anumit număr de puncte. Punctajul maxim care poate fi obținut (dacă toate cele 20 de fișiere de ieșire sunt corecte) este de 100 de punte.

Pentru primele patru teste (**01 - 04**) se vor acorda câte 3 puncte, pentru următoarele patru (**05 - 08**) câte 4 puncte, pentru următoarele patru (**09 - 12**) câte 5 puncte, pentru următoarele patru (**13 - 16**) câte 6 puncte, iar pentru ultimele patru (**17 - 20**) câte 7 puncte.

Trimiterea fișierelor de ieșire

Arhiva pe care o veți trimite se va numi **YYYYYR07.ZIP**, **YYYYYR07.RAR** sau **YYYYYR07.ACE** (în funcție de programul de arhivare pe care îl folosiți), unde **YYYYY** este codul dumneavoastră de identificare (ID).

Arhiva nu va conține fișiere batch (**YYYYYR07.BAT**) sau fișiere sursă (**YYYYYR07.PAS**, **YYYYYR07.CPP** sau **YYYYYR07.C**). În arhivă se vor afla cele 20 de fișiere de ieșire corecte, denumirea lor fiind **FACTORXX.OUT**, unde **XX** reprezintă numărul testului (cuprins între **01** și **20**).

P080218: Graf

Se consideră un graf neorientat cu *N* noduri și *N* muchii în care fiecare nod are exact ***două*** muchii adiacente (fiecare nod are gradul 2).

Costurile muchiilor sunt numere prime, dar sunt necunoscute. Costul unui nod este dat de produsul costurilor muchiilor adiacente nodului respectiv. Nodurile sunt identificate prin numere cuprinse între 1 și *N*, iar costurile lor sunt cunoscute.

Va trebui să determinați costurile muchiilor folosind datele furnizate de o bibliotecă externă. Biblioteca vă va pune la dipoziție rutine pentru:

- inițializare;
- determinarea numărului de noduri ale grafului;

- determinarea costului unui nod;
- verificarea existenței unei muchii între două noduri;
- verificarea costului unei muchii;
- furnizarea ca rezultat a costului unei muchii;
- determinarea numărului maxim de verificări ale costurilor muchiilor.

Programul vostru va trebui să determine costurile muchiilor folosind un număr limitat de apeluri ale funcției de verificare a costului unei muchii. Programul va trebui să furnizeze rezultate pentru toate cele *N* muchii ale grafului, deci vor exista *N* apeluri ale rutinei de furnizare a costului unui muchii.

Bibliotecile Pascal și C/C++

Pentru a putea folosi biblioteca va trebui să includeți în programul dumneavoastră instrucțiunea **uses** `graph`; pentru limbajul *Pascal* și `#include "graph.h"` pentru limbajul *C/C++*. În continuare sunt prezentate sintaxele funcțiilor și procedurilor incluse în biblioteci:

```
procedure Init;
void Init()
```

- trebuie apelată la începutul programului și este folosită de către bibliotecă pentru inițializare;
- întrerupe execuția programului dacă este apelată a doua oară.

```
function GetNumberOfNodes:Integer;
int GetNumberOfNodes()
```

- returnează numărul nodurilor (și al muchiilor) din graf.

```
function IsEdgeBetween(x,y:Integer):Boolean;
int IsEdgeBetween(int x,int y)
```

- returnează `true` (valoare nenulă) dacă există o muchie între nodurile `x` și `y` și `false` (0) dacă nu există o astfel de muchie;
- întrerupe execuția programului dacă cel puțin una dintre valorile `x` și `y` nu reprezintă un nod al grafului.

```
function CheckEdgeWeight(x,y,c:Integer):
                                        Boolean;
int CheckEdgeWeight(int x,int y,int c)
```

- returnează `true` (valoare nenulă) dacă muchia dintre nodurile `x` și `y` are costul `c` și `false` (0) dacă muchia are costul diferit de `c`;
- întrerupe execuția programului dacă cel puțin una dintre valorile `x` și `y` nu reprezintă un nod al grafului, dacă muchia determinată de aceste noduri nu face parte din graf sau dacă a fost depășit numărul maxim admis de apeluri ale acestei funcții.

```
function GetNodeWeight(x:Integer):Longint;
long GetNodeWeight(int x)
```

- returnează costul nodului `x`;
- întrerupe execuția programului dacă `x` nu reprezintă un nod al grafului.

```
function GetNumberOfCallsLeft:Integer;
int GetNumberOfCallsLeft()
```

- returnează un număr întreg care reprezintă numărul de apeluri ale funcției `CheckEdgeWeight` pe care le mai aveți la dispoziție.

```
function Result(x,y,c:Integer):Boolean;
int Result(int x,int y,int c)
```

- acceptă ca rezultat faptul că muchia dintre nodurile `x` și `y` are costul `c`;
- întrerupe execuția programului dacă:
  - cel puțin una dintre valorile `x` și `y` nu reprezintă un nod al grafului;
  - muchia determinată de aceste noduri nu face parte din graf;
  - costul muchiei dintre nodurile `x` și `y` nu este `c`;
  - costul muchiei dintre nodurile `x` și `y` este furnizat a doua oară;
  - au fost furnizate costurile tuturor celor *N* muchii ale grafului (*Acesta este singurul caz în care sunt acordate puncte!*).

Toate funcțiile și procedurile (excepție: `Init`) întrerup execuția programului dacă sunt apelate înaintea inițializării.

Restricții și precizări

- Programul vostru nu va citi date din nici un fișier de intrare și nu va scrie date în nici un fișier de ieșire.
- $3 \le N \le 1000$.
- Costurile muchiilor sunt numere prime cuprinse între 2 și 29989.

Exemplu de utilizare a bibliotecii

Subprogram apelat	*Valoare returnată*
`Init`	-
`GetNumberOfNodes`	4
`IsEdgeBetween(1,2)`	true (1)
`IsEdgeBetween(1,3)`	false (0)
`IsEdgeBetween(1,4)`	true (1)
`IsEdgeBetween(2,3)`	true (1)
`IsEdgeBetween(2,4)`	false (0)
`IsEdgeBetween(3,4)`	true (1)
`GetNumberOfCallsLeft`	2
`GetNodeWeight(1)`	14
`GetNodeWeight(2)`	6
`GetNodeWeight(3)`	15
`GetNodeWeight(4)`	35
`CheckEdgeWeight(1,2,7)`	false (0)
`CheckEdgeWeight(1,2,2)`	true (1)
`Result(1,2,2)`	-
`Result(2,3,3)`	-
`Result(3,4,5)`	-
`Result(4,1,7)`	-

Timp de execuție: 5 secunde/test