Descompunerea în factori primi

Autor: Ovidiu Marian Marcu
Colegiul Național ”Ștefan cel Mare” Suceava

Problemă

Se dă un număr natural **n.** Afișați, pe câte un rând al ecranului, factorii din descompunerea lui **n** în factori primi și puterile la care aceștia apar.

Exemplu: pentru **n** = 40 se va afișa pe ecran:

2 3
5 1

Deoarece **n**=2^3*5^1.

Cum se descompune în factori primi un număr natural?

Se consideră o valoare inițială a factorului **f=2.** Se împarte **n** la **f** cât timp se poate și se contorizează numărul de repetiții; acesta fiind puterea **p** la care va apărea factorul **f** în descompunere. Apoi se incrementează valoarea factorului și se continuă algoritmul până când **n=1.**

n=40	**f=2**
n=20	**f=2**
n=10	**f=2, p=3**
n=5	**f=5, p=1**
n=1	

```
#include <iostream>
using namespace std;
int main()
{
   int n,f,p;
```

```
    cin>>n;
    f=2;
    while(n>1)
    {
       p=0;
       while(n%f==0)
       {
          n=n/f;
          p++;
       }
       if(p>0)
          cout<<f<<" "<<p<<"\n";
       f++;
    }
    return 0;
}
```

Probleme rezolvate

1. Se dau două numere naturale nenule **a** și **b**. Să se determine numărul din intervalul [**a**,**b**] care are număr maxim de divizori. Dacă există mai multe asemenea numere, se va afișa cel mai mare dintre ele.
   Exemplu: pentru **a=4** și **b=19** se va afișa **18** deoarece este numărul cu cel mai mare număr de divizori (12 și 18 au câte 6 divizori, dar 18 este mai mare).

```
#include <iostream>
#include <cmath>
using namespace std;
int numar_divizori(int n)
{
   int p=0,f=2,nr=1;
```

```
    while(n>1)
    {
        p=0;
        while(n%f==0)
        {
            p++;
            n=n/f;
        }
        if(p>0) nr=nr*(p+1);
        f++;
    }
    return nr;
}
int main()
{
    int a,b,x,maxx=0,val,nr;
    cin>>a>>b;
    for(x=a; x<=b; x++)
    {
        nr=numar_divizori(x);
        if(nr>=maxx)
        {
            maxx=nr;
            val=x;
        }
    }
    cout<<val;
    return 0;
}
```

2. Problema **arma –** Olimpiada Județeană de Informatică, clasa a VIII-a, 2016

Enunț

În anul 2214 a izbucnit primul război interstelar. Pământul a fost atacat de către n civilizaţii extraterestre, pe care le vom numerota pentru simplicitate de la 1 la n.

Pentru a se apăra, pământenii au inventat o armă specială ce poate fi încărcată cu proiectile de diferite greutăţi, fabricate dintr-un material special denumit narun. Dacă arma este programată la nivelul p, atunci un proiectil de greutate k va ajunge exact la distanţa kp km (k la puterea p) faţă de Pământ şi dacă în acel punct se află cartierul general al unui atacator, acesta va fi distrus. De exemplu, dacă arma este programată la nivelul 2, un proiectil de greutate 10 va distruge cartierul general al extratereştrilor situat la distanţa 102 = 100 km de Pământ.

Arma poate fi încărcată cu proiectile de diferite greutăţi, dar cum narunul este un material foarte rar şi foarte scump, pământenii vor să folosească proiectile cât mai uşoare pentru a distruge cartierele generale inamice.

Cerință

Cunoscându-se **n**, numărul atacatorilor, precum şi cele **n** distanţe până la cartierele generale ale acestora, să se scrie un program care determină:

1. cantitatea minimă de narun necesară pentru a distruge toate cartierele generale inamice;
2. nivelurile la care trebuie programată arma, pentru a distruge fiecare cartier general inamic cu o cantitate minimă de narun.

Date de intrare

Fișierul de intrare **arma.in** conține pe prima linie un număr natural **c** reprezentând cerinţa care trebuie să fie rezolvată (1 sau 2). Pe cea de a doua linie se află numărul natural **n**, reprezentând numărul atacatorilor. Pe următoarele **n** linii se află **n** numere naturale, câte un număr pe o linie; pe cea de a i-a linie dintre cele **n** ($1 \leq i \leq n$) se află distanţa faţă de Pământ a cartierului general al celei de a i-a civilizaţii extraterestre.

Date de ieșire

Dacă cerinţa c=1, atunci pe prima linie a fişierului **arma.out** va fi scris un număr natural reprezentând cantitatea minimă de narun necesară distrugerii tuturor cartierelor generale inamice. Dacă cerinţa este c=2, atunci fişierul de ieşire **arma.out** va conţine n linii. Pe a i-a linie **(1≤i≤n)** se va scrie nivelul la care trebuie programată arma pentru a distruge cartierul general al celei de a i-a civilizaţii extraterestre.

Restricții și precizări

- **1 ≤ n ≤ 10 000**
- Distanţele până la cartierele generale inamice sunt numere naturale nenule ≤ 2.000.000.000.
- Pentru 50% dintre teste cerinţa este 1.

Exemple

arma.in	**arma.out**	**arma.in**	**arma.out**	**Explicaţíe**
1 5 100 97 625 40353607 81	122	2 5 100 97 625 40353607 81	2 1 4 9 4	Primul cartier general se poate distruge cu un proiectil de greutate 10, programat la nivelul 2, al doilea obiectiv cu un proiectil de greutate 97 programat la nivelul 1, al treilea cu un proiectil de greutate 5 programat la nivelul 4, al patrulea cu un proiectil de greutate 7 programat la nivelul 9, iar ultimul cu un proiectil de greutate 3 programat la nivelul 4. Cantitatea minimă de narun necesară este 10+97+5+7+3=122. Nivelurile sunt în ordine: 2 1 4 9 4

Timp maxim de execuţie/test: 0.6 secunde

Memorie totală disponibilă 4 MB, din care 2 MB pentru stivă

Dimensiunea maximă a sursei: 10 KB

Descrierea soluției:

Folosind ciurul lui Eratostene, se generează numerele prime până la 48.000 cu ajutorul vectorului **a,** totodată se memorează numerele prime în vectorul **prime.**

Se descompune în factori primi fiecare număr citit, folosind pentru alegerea factorilor vectorul **prime**. Se memorează, pe un rând nou în matricea **v,** pe prima coloană factorul și pe a doua puterea la care apare. Se efectuează apoi cmmdc dintre valorile memorate pe a doua coloană a matricei.

Implementarea în C++:

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream fin("arma.in");
ofstream fout("arma.out");
int a[500001], prime[100001], nr;
int eratostene()
{
   int i, j;
   for(i=4; i<=48000; i=i+2)
      a[i]=1;
   nr++;
   prime[nr]=2;
   for(i=3; i<=48000; i=i+2)
   {
      if(a[i]==0)
        for(j=2; j<=48000/i; j++)
            a[i*j]=1;
      if(a[i]==0)
      {
         nr++;
         prime[nr]=i;
```

```
      }
   }
}
int v[100][3];
int main()
{
   eratostene();
   long  i, p, a, n, j, num, b, c, r;
   long long T=0;
   fin >> p;
   fin >> n;
   if(p==1)
   {
      for(i=1; i<=n; i++)
      {
         fin >> a;
         num=0;
         j=1;
         int copie=a;
         while(a>1 && j<=nr && prime[j]<=a)
         {
            if(a%prime[j]==0)
            {
               int aux=0;
               while(a%prime[j]==0)
               {
                  aux++;
                  a=a/prime[j];
               }
               num++;
               v[num][1]=prime[j];
```

```
            v[num][2]=aux;
        }
        j++;
    }
    if(num==0 || (num==1 && v[num][2]==1) || a>1)
        T=T+copie;
    else
    {
        if(num==1)   T=T+v[num][1];
        else
        {
            b=v[1][2];
            for(j=2; j<=num; j++)
            {
                c=v[j][2];
                r=b%c;
                while(r)
                {
                    b=c;
                    c=r;
                    r=b%c;
                }
                b=c;
            }
            if(b==1) T=T+copie;
            else
            {
                long long  pr=1;
                for(j=1; j<=num; j++)
                    for(int k=1; k<=v[j][2]/b; k++)
                        pr=pr*v[j][1];
```

```
                T=T+pr;
            }
        }
    }
    fout << T;
}
if(p==2)
 {
    for(i=1; i<=n; i++)
    {
        fin >> a;
        num=0;
        j=1;
        int copie=a;
        while(a>1 && j<=nr && prime[j]<=a)
        {
            if(a%prime[j]==0)
            {
                int aux=0;
                while(a%prime[j]==0)
                {
                    aux++;
                    a=a/prime[j];
                }
                num++;
                v[num][1]=prime[j];
                v[num][2]=aux;
            }
            j++;
        }
```

```
        if(num==0 || (num==1 && v[num][2]==1) || a>1)
           fout << 1 << "\n";
        else
        {
           if(num==1)  fout << v[1][2] << "\n";
           else
           {
              b=v[1][2];
              for(j=2; j<=num; j++)
              {
                 c=v[j][2];
                 r=b%c;
                 while(r)
                 {
                    b=c;
                    c=r;
                    r=b%c;
                 }
                 b=c;
              }
              fout << b << "\n";
           }
        }
     }
  }
  return 0;
}
```

Aplicații ale descompunerii în factori primi

Un număr natural nenul **n** mai mare decât **1** se poate scrie astfel:

$$n = f_1^{p1} * f_2^{p2} * ... * f_k^{pk}$$

Exemplu: **n=20** se scrie ca $\mathbf{2^2*5^1}$**.** Deci **n** are **k=2** factori în descompunerea în factori primi.

1. **Numărul de divizori**

   Numărul de divizori ai numărului **n** se poate afla după formula:

$$nr = (p_1 + 1) * (p_2 + 1) * ... * (p_k + 1)$$

   Exemplu: **n=20** are (2+1)*(1+1) divizori, adică **6**: 1, 2, 4, 5, 10, 20

   Programul C++:

```
#include <iostream>
using namespace std;
int main()
{
   int n,f,p=0,nr=1;
   cin>>n;
   f=2;
   while(n%f==0)
   {
      n=n/f;
      p++;
   }
   if(p>0)
      nr=nr*(p+1);
   f=3;
```

```
    while(n>1)
    {
       p=0;
       while(n%f==0)
       {
          n=n/f;
          p++;
       }
       if(p>0)
          nr=nr*(p+1);
       f=f+2;
    }
    cout<<nr;
    return 0;
}
```

2. Suma divizorilor

Suma divizorilor numărului **n** se poate afla după formula:

$$S = \frac{f_1^{p1+1} - 1}{f_1 - 1} * \frac{f_2^{p2+1} - 1}{f_2 - 1} * \dots * \frac{f_k^{pk+1} - 1}{f_k - 1}$$

Exemplu: Pentru **n=20** suma divizorilor este $S = \frac{2^3-1}{2-1} * \frac{5^2-1}{5-1} = 42$

Programul C++:

```
#include <iostream>
using namespace std;
int main()
{
```

```
int n,f,p=0,S=1,nr;
cin>>n;
f=2;
while(n>1)
{
   p=1;nr=0;
   while(n%f==0)
   {
      n=n/f;
      p=p*f;
                nr++;
   }
   if(nr>0)
      S=S*(p*f-1)/(f-1);
   f=f+1;
}
cout<<S;
return 0;
}
```

3. Indicatorul lui Euler

Indicatorul lui Euler sau **totient** reprezintă numărul de numere prime cu **n,** mai mici sau egale cu **n.** Acesta se calculează după formula:

$$\varphi(n) = n * \left(1 - \frac{1}{f_1}\right) * \left(1 - \frac{1}{f_2}\right) * \ldots * \left(1 - \frac{1}{f_k}\right)$$

$$sau$$

$$\varphi(n) = (f_1 - 1) * f_1^{p1-1} * (f_2 - 1) * f_2^{p2-1} * \ldots * (f_k - 1) * f_k^{pk-1}$$

Exemplu: Pentru **n=20** $\varphi(n) = 20 * \left(1 - \frac{1}{2}\right) * \left(1 - \frac{1}{5}\right) = 8$

Programul C++:

```
#include <iostream>
using namespace std;
int main()
{
   int n,f,p=0,fi=1,nr;
   cin>>n;
   f=2;
   while(n>1)
   {
      p=1;nr=0;
      while(n%f==0)
      {
         n=n/f;
         p=p*f;
         nr++;
      }
      if(nr>0)
      {
         p=p/f;
         fi=fi*(f-1)*p;
      }
      f=f+1;
   }
   cout<<fi;
   return 0;
}
```