ECN versenyfelkészítő

MÁRTON Gyöngyvér

Sapientia Egyetem, Matematika és Informatika Tanszék,
Marosvásárhely, Románia
mgyongyi@ms.sapientia.ro
http://www.ms.sapientia.ro/~mgyongyi

2013, november

Könyvészet:
Skiena, Programming Challenge
Web oldalak:

- https://github.com/lbv/pc-code
- http://www.codechef.com/
- http://www.spoj.com/problems/

1. feladat (Skiena, Programming Challenge)

- Az $n!$ faktoriális függvényt, bármely nem negatív egész szám esetében a következőképpen definiálhatjuk:

$$0! = 1, \text{ és } n! = n \cdot (n-1)!, \text{ ha } (n > 0).$$

- A program bemenete több tesztesetből áll, mindegyik teszteset két számot tartalmaz, n-t és m-t, ahol $n, m < 2^{31}$.
- A program kimenete, mindegyik bemenet esetében meg kell vizsgálja, hogy m osztja-e $n!$-t, az eredményt a lenti példánál feltüntetett formában megadva.

Példa:

BE.IN
6 9
6 27
20 10000
20 100000
1000 1009

Standard kimenet
9 divides 6!
27 does not divide 6!
10000 divides 20!
100000 does not divide 20!
1009 does not divide 1000!

- meg kell vizsgálni, hogy az m prímfaktorizációjában szereplő prímek legnagyobb hatvány értéke kisebb-e, mint az $n!$ prímfaktorizációjában a legnagyobb prímhatvány,
- a 2^{16}-nál kisebb prímek meghatározása,
- az m prímfaktorizációjának meghatározása,
- nem kell az $n!$ prímfaktorizációja,
- meg kell vizsgálni, hogy a p prím legnagyobb e hatványa, azaz a p^e osztja-e $n!$-t.

1. módszer: meghatározza 2^{16}-ig a prímeket. A *primes* tömbben, ha az *i* pozíción 1 van, akkor az *i* prímszám, ha 0 van, akkor az *i* összetett szám. A *primes* tömb tárigénye: 65536 byte.

```
const int MAX = 1 << 16;
const int LMT = 1 << 8;

void prime_sieve(){
    bool primes[MAX] = {0};

    for (int i = 3; i < LMT; i += 2) {
        if (primes[i] == 0) {
            for (int j = i * i; j < MAX; j += i+i) primes[j] = 1;
            //beallitja a megfelelo bitet 1-re, ami azt
            //jelenteni, hogy a j osszetett szam
        }
    }

    cout << "2 ";
    for(int i = 3; i < MAX; i += 2)
        if(primes[i] == 0)
            cout << i << " ";
}
```

2. módszer, bit műveletekkel, sokkal kisebb tárkapacitás esetén. A *primes* tömb tárigénye 4100 byte.

```
#define IsComp(n) (primes[ n >> 6 ] & ( 1 << (( n >> 1 ) & 31 )))
//megnezi, hogy az n-nek megfelelo bit 1-e?
#define SetComp(n) (primes[ n >> 6 ] |= ( 1 << (( n >> 1) & 31 )))
//beallitja az n-nek megfelelo bitet 1-re

int primes[( MAX >> 6 ) + 1 ] = {0};

void prime_sieve_1() {
    for (int i = 3; i <= LMT; i += 2)
        if (!IsComp(i)) {
            for (int j = i*i; j <= MAX; j += i+i) SetComp(j);
        }

    cout << "2 ";
    for (int i = 3; i <= MAX; i += 2) if (!IsComp(i)) {
        cout << i << " ";
    }
}
```

- meghatározza, hogy a p hatványai, milyen legnagyobb e többszörösel szerepelnek az n-ben, összegezve a többszörösöket.

```
int pow_div_fact(int n, int p)
{
    int e = 0;
    for (int d = p; d <= n; d *= p)
        e += n/d;
    return e;
}
```

2. feladat (http://www.spoj.com/problems/)

- Határozzuk meg az összes 10^8-nál kisebb prímszámot.
- A program kimenetként meg kell határozza az első, a 101., a 201., 301., 401., ... prímszámokat.

Példa:

Standard kimenet
2
547
1229
...
99995257
99996931
99998953

3. feladat: Határozzuk meg $n!$ értékét.

1. módszer

- feltételezzük, hogy n páros szám:

$$\begin{aligned} 20! &= (1 \cdot 20) \cdot (2 \cdot 19) \cdot (3 \cdot 18) \cdot \ldots \\ 20! &= 20 \cdot (20 + 18) \cdot (20 + 18 + 16) \cdot (20 + 18 + 16 + 14) \cdot \ldots \\ n! &= n \cdot (n + n - 2) \cdot (n + n - 2 + n - 4) \cdot \ldots \end{aligned}$$

- Rekurzív formában:

$$\begin{aligned} fakt\ n\ 0 &= 1 \\ fakt\ n\ m &= (n + m) \cdot fakt\ (n + m)\ (m - 2) \end{aligned}$$

2. módszer

- Peter B. Borwein, módszere (On the Complexity of Calculating Fatorials, Journal of Algorithms, 1985)
- meghatározzuk $n!$ prímtényezős felbontását,
- a gyorshatványozás algoritmusával meghatározzuk a hatványértékeket, majd ezeket összeszorozzuk,

3. módszer

- feltételezzük, hogy n páros szám,
- divide et impera módszert alkalmazunk,

$$\begin{aligned}(2n)! &= 1 \cdot 2 \cdot 3 \cdot \dots \cdot 2n \\ &= n! \cdot 2^n \cdot 3 \cdot 5 \cdot \dots \cdot (2n-1)\end{aligned}$$

További módszerek:
http://www.luschny.de/math/factorial/FastFactorialFunctions.htm