

Resumen de algoritmos para torneos de programación

Andrés Mejía

22 de agosto de 2008

Índice

1. Teoría de números	1
1.1. Big mod	1
1.2. Criba de Eratóstenes	2
1.3. Divisores de un número	2
2. Programación dinámica	4
2.1. Longest common subsequence	4

Listings

1. Big mod	1
2. Criba de Eratóstenes	2
3. Divisores	3
4. Longest common subsequence	4

1. Teoría de números

1.1. Big mod

Listing 1: Big mod

```
1 //retorna (b^p)mod(m)
2 // 0 <= b, p <= 2147483647
3 // 1 <= m <= 46340
4 long f(long b, long p, long m){
5     long mask = 1;
6     long pow2 = b %m;
7     long r = 1;
8
9     while (mask){
10        if (p & mask)
11            r = (r * pow2) %m;
12        pow2 = (pow2*pow2) %m;
13        mask <<= 1;
```

```

    }
15  return r;
    }

```

1.2. Criba de Eratóstenes

Marca los números primos en un arreglo. Algunos tiempos de ejecución:

SIZE	Tiempo (s)
100000	0.004
1000000	0.078
10000000	1.550
100000000	14.319

Listing 2: Criba de Eratóstenes

```

#include <iostream>
2
const int SIZE = 1000000;
4
//criba[i] = false si i es primo
6 bool criba[SIZE+1];
8 void buildCriba(){
    memset(criba, false, sizeof(criba));
10
    criba[0] = criba[1] = true;
12 for (int i=2; i<=SIZE; i += 2){
    criba[i] = true;
14 }
16 for (int i=3; i<=SIZE; i += 2){
    if (!criba[i]){
18         for (int j=i+i; j<=SIZE; j += i){
            criba[j] = true;
20         }
    }
22 }
}

```

1.3. Divisores de un número

Este algoritmo imprime todos los divisores de un número (en desorden) en $O(\sqrt{n})$. Hasta 4294967295 (máximo *unsigned long*) responde instantaneamente. Se puede forzar un poco más usando *unsigned long long* pero más allá de 10^{12} empieza a responder muy lento.

Listing 3: Divisores

```

1 for (int i=1; i*i<=n; i++) {

```

```

3   if (n%a == 0) {
      cout << i << endl;
5   if (i*i<n) cout << (n/i) << endl;
      }
  }
}

```

2. Programación dinámica

2.1. Longest common subsequence

Listing 4: Longest common subsequence

```

1 #define MAX(a,b) ((a>b)?(a):(b))
2 int dp[1001][1001];
3
4 int lcs(const string &s, const string &t){
      int m = s.size(), n = t.size();
5   if (m == 0 || n == 0) return 0;
      for (int i=0; i<=m; ++i)
6   for (int j=1; j<=n; ++j)
8   for (int i=0; i<=m; ++i)
10  for (int j=0; j<=n; ++j)
12  for (int j=0; j<=n; ++j)
14  if (s[i] == t[j])
      dp[i+1][j+1] = dp[i][j]+1;
      else
16  dp[i+1][j+1] = MAX(dp[i+1][j], dp[i][j+1]);
      return dp[m][n];
18 }

```