Resumen de algoritmos para torneos de programación

Andrés Mejía

22 de marzo de 2008

Índice

Listings

1. Teoría de números

1.1. Big mod

Listing 1: Big mod

```
//retorna (b^p)mod(m)
// 0 <= b,p <= 2147483647
// 1 <= m <= 46340
long f(long b, long p, long m){
  long mask = 1;
  long pow2 = b %m;
  long r = 1;

  while (mask){
    if (p & mask)
      r = (r * pow2) %m;
    pow2 = (pow2*pow2) %m;
    mask <<= 1;
```

```
  }
  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>

const int SIZE = 1000000;

//criba[i] = false si i es primo
bool criba[SIZE+1];

void buildCriba(){
  memset(criba, false, sizeof(criba));

  criba[0] = criba[1] = true;
  for (int i=2; i<=SIZE; i += 2){
    criba[i] = true;
  }

  for (int i=3; i<=SIZE; i += 2){
    if (!criba[i]){
      for (int j=i+i; j<=SIZE; j += i){
        criba[j] = true;
      }
    }
  }
}
```

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.

```
001 #include<iostream>
002 #include<math.h>
003 #include <set>
004
```

```
using namespace std;

typedef unsigned long long ull;

inline ull cube(ull x){ return x*x*x; }

int main(){
  ull n;
  while(cin>>n &&n){
    bool found = false;
    for (ull k = 1; k*k<=n && k <= 29241; ++k){
      if (n% k != 0 ||
    3*k*k*k*k > 12*k*n){
  continue;
      }
      ull y = ( -3*k*k + (ull)sqrt(12*k*n - 3*k*k*k*k) ) / (6*k);
      if (y > 0 && cube(y+k) - cube(y) == n){
  //  cout << "k: " << k << endl;
  cout << y + k << " " << y << endl;
  found = true;
  break;
      }
      /*if (k*k < n){
  k = n/k;
  ull y = ( -3*k*k + (ull)sqrt(12*k*n - 3*k*k*k*k) ) / (6*k);
  if (y > 0 && cube(y+k) - cube(y) == n){
    //  cout << "k: " << k << endl;
    cout << y + k << " " << y << endl;
    found = true;
    break;
  }
  k = n/k;
  }*/
    }

    if (!found){
      cout << "No solution" << endl;
    }
  }
  return 0;
}

```

Listing 3: Divisores

```
for (int i=1; i*i<=n; i++) {
  if (n%i == 0) {
    cout << i << endl;
    if (i*i<n) cout << (n/i) << endl;
  }
}
```

2. Programación dinámica

2.1. Longest common subsequence

Listing 4: Longest common subsequence

```
#define MAX(a,b) ((a>b)?(a):(b))
int dp[1001][1001];

int lcs(const string &s, const string &t){
  int m = s.size(), n = t.size();
  if (m == 0 || n == 0) return 0;
  for (int i=0; i<=m; ++i)
    dp[i][0] = 0;
  for (int j=1; j<=n; ++j)
    dp[0][j] = 0;
  for (int i=0; i<m; ++i)
    for (int j=0; j<n; ++j)
      if (s[i] == t[j])
        dp[i+1][j+1] = dp[i][j]+1;
      else
        dp[i+1][j+1] = MAX(dp[i+1][j], dp[i][j+1]);
  return dp[m][n];
}
```