.

Contents

1 Data structures

1.1 Segment tree

```
#define oper min
#define NEUT INF
struct STree { // segment tree for min over integers
  vector<int> st;int n;
  STree(int n): st(4*n+5,NEUT), n(n) {}
  void init(int k, int s, int e, int *a){
    if(s+1==e){st[k]=a[s];return;}
    int m=(s+e)/2;
    init(2*k,s,m,a);init(2*k+1,m,e,a);
    st[k]=oper(st[2*k],st[2*k+1]);
  }
  void upd(int k, int s, int e, int p, int v){
    if(s+1==e){st[k]=v;return;}
    int m=(s+e)/2;
    if(p<m)upd(2*k,s,m,p,v);
    else upd(2*k+1,m,e,p,v);
    st[k]=oper(st[2*k],st[2*k+1]);
  }
  int query(int k, int s, int e, int a, int b){
    if(s>=b||e<=a)return NEUT;
    if(s>=a&&e<=b)return st[k];
    int m=(s+e)/2;
    return oper(query(2*k,s,m,a,b),query(2*k+1,m,e,a,b));
  }
  void init(int *a){init(1,0,n,a);}
  void upd(int p, int v){upd(1,0,n,p,v);}
  int query(int a, int b){return query(1,0,n,a,b);}
}; // usage: STree rmq(n);rmq.init(x);rmq.upd(i,v);rmq.query(s,e);
```

2 Matemáticas

2.1 Matrices

Multiplicación de matrices

```
vector<vector<ll>> mult_mm(vector<vector<ll>>&A,vector<vector<ll>>&B){
  int n = A.size(), m = A[0].size(), p = B[0].size();
  vector<vector<ll>> ans(n, vector<ll>(p, 0));
  for( int i=0; i<n; i++ ){
    for( int j=0; j<m; j++ ){
```

```
      for( int k=0; k<p; k++ ){
        ans[i][k]+=A[i][j]*B[j][k];
        ans%=mod;
      }
    }
  }
  return ans;
}
```

2.2 Divisores

```
// Divisores
vector<int> divisores = calcular_divisores(30);
for( int div:divisores ) {
  cout << div << "\n";
}
```

El siguiente código sirve para implementar la función `calcular_divisores`. El algoritmo tiene complejidad de $O(\sqrt{n})$.

```
vector<int> calcular_divisores(int numero) {
  vector<int> divisores;
  for( int i=1; i*i <= numero; i++ ) {
    if( numero%i == 0 ) {
      divisores.push_back(i);
      if( i*i != numero ) {
        divisores.push_back(numero/i);
      }
    }
  }
  return divisores;
}
```

2.3 Criba de primos

La siguiente criba les servirá para calcular los primeros 78,499 primos (todos los primos menores o iguales a un millón). Si quieren saber si un número es primo, lo pueden consultar de manera constante checando el arreglo `es_primo`. Antes de usar los arreglos `es_primo` y `primos`, hay que llamar a la función `criba_primos`.