No me Baño

24 de septiembre de 2019

Resumen

Repositorio 'Guardian de la tortuga' de 'No me baño' UNR FCEIA

1. Template

```cpp
#include <bits/stdc++.h>
using namespace std;
#define forr(i,a,b) for(int i=(a); i<(b); i++)
#define forn(i,n) forr(i,0,n)
#define sz(c) ((int)c.size())
#define zero(v) memset(v, 0, sizeof(v))
#define forall(it,v) for(auto it=v.begin();it!=v.end();++it
   )
#define pb push_back
#define fst first
#define snd second
typedef unsigned long long ull;
typedef long long ll;
typedef pair<int,int> ii;
#define dforn(i,n) for(int i=n-1; i>=0; i--)
#define dprint(v) cout << #v"=" << v << endl
#define endl "\n"

const int MAXN=100100;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(nullptr);
        return 0;
}
```

2. DataStructure

Binary Search

```
//the following code counts how many times the key appears
    in the array.
//After the search, [p+1..q] is the range with value k.
int count(int x[], int n, int k) {
    int p = -1, q = -1;
    for (int a = n; a >= 1; a /= 2) {
        while (p+a < n && x[p+a] < k) p += a;
        while (q+a < n && x[q+a] <= k) q += a;
    }
    return q-p;
}
```

Disjoint Intervals Olaf

```
// stores disjoint intervals as [first, second)
struct disjoint_intervals {
        set<pair<int,int> > s;
        void insert(pair<int,int> v){
                if(v.fst>=v.snd) return;
                auto at=s.lower_bound(v);auto it=at;
                if(at!=s.begin()&&(--at)->snd>=v.fst)v.fst=
                    at->fst,--it;
                for(;it!=s.end()&&it->fst<=v.snd;s.erase(it
                    ++))
                        v.snd=max(v.snd,it->snd);
                segs.insert(v);
        }
};
```

Binary Search

```
struct STree { // [cerrado-abierto), defininir "oper" y "
    NEUT"
    vector<ll> st;int n;
    STree(int n): st(4*n+5,NEUT), n(n) {}
    void upd(int k, int s, int e, int p, ll 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]);
    }
    ll 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 upd(int p, ll v){upd(1,0,n,p,v);}
    ll query(int a, int b){return query(1,0,n,a,b);}
}; // usage: STree st(n);st.upd(i,v);st.query(s,e);
```

3. Factorizacion

Divs

```
#define MAXP 100000     //no necesariamente primo

int criba[MAXP+1];

void crearcriba(){
        int w[] = {4,2,4,2,4,6,2,6};
        for(int p=25;p<=MAXP;p+=10) criba[p]=5;
        for(int p=9;p<=MAXP;p+=6) criba[p]=3;
        for(int p=4;p<=MAXP;p+=2) criba[p]=2;
        for(int p=7,cur=0;p*p<=MAXP;p+=w[cur++&7]) if (!
           criba[p])
                for(int j=p*p;j<=MAXP;j+=(p<<1)) if(!criba[
                   j]) criba[j]=p;
}

vector<int> primos;

void buscarprimos(){
        crearcriba();
        forr (i,2,MAXP+1) if (!criba[i]) primos.push_back(i
           );
}

//factoriza bien numeros hasta MAXP^2
map<ll,ll> fact(ll n){ //O (cant primos)
        map<ll,ll> ret;
        forall(p, primos){
                while(!(n%*p)){
                        ret[*p]++;//divisor found
                        n/=*p;
                }
        }
        if(n>1) ret[n]++;
        return ret;
}

//factoriza bien numeros hasta MAXP
map<ll,ll> fact2(ll n){ //O (lg n)
        map<ll,ll> ret;
        while (criba[n]){
                ret[criba[n]]++;
                n/=criba[n];
        }
        if(n>1) ret[n]++;
        return ret;
}

//Usar asi:  divisores(fac, divs, fac.begin()); NO ESTA
    ORDENADO
void divisores(const map<ll,ll> &f, vector<ll> &divs, map<
    ll,ll>::iterator it, ll n=1){
     if(it==f.begin()) divs.clear();
     if(it==f.end()) { divs.pb(n);  return; }
     ll p=it->fst, k=it->snd; ++it;
     forn(_, k+1) divisores(f, divs, it, n), n*=p;
}

ll sumDiv (ll n){
  ll rta = 1;
  map<ll,ll> f=fact(n);
  forall(it, f) {
        ll pot = 1, aux = 0;
        forn(i, it->snd+1) aux += pot, pot *= it->fst;
        rta*=aux;
  }
  return rta;
}

int main() {
        buscarprimos();
        return 0;
}
```

Fact Basic

```
vector<ll> factorize(ll m){
  if(m==1) return{};
  vector<ll> fact;
  for(ll a=2; a*a<=m; a++){
    while(m%a==0){
      fact.push_back(a);
      m/=a;
    }
  }
  if(m!=1) fact.push_back(m);
  return fact;
}
```

Fact

```
#define MAXP 100000     //no necesariamente primo

int criba[MAXP+1];

void crearcriba(){
        int w[] = {4,2,4,2,4,6,2,6};
        for(int p=25;p<=MAXP;p+=10) criba[p]=5;
        for(int p=9;p<=MAXP;p+=6) criba[p]=3;
        for(int p=4;p<=MAXP;p+=2) criba[p]=2;
        for(int p=7,cur=0;p*p<=MAXP;p+=w[cur++&7]) if (!
           criba[p])
                for(int j=p*p;j<=MAXP;j+=(p<<1)) if(!criba[
                   j]) criba[j]=p;
}

vector<int> primos;
void buscarprimos(){
        crearcriba();
        forr (i,2,MAXP+1) if (!criba[i]) primos.push_back(i
           );
}

//factoriza bien numeros hasta MAXP^2
map<ll,ll> fact(ll n){ //O (cant primos)
        map<ll,ll> ret;
        forall(p, primos){
                while(!(n%*p)){
                        ret[*p]++;//divisor found
                        n/=*p;
                }
        }
        if(n>1) ret[n]++;
        return ret;
}

//factoriza bien numeros hasta MAXP
map<ll,ll> fact2(ll n){ //O (lg n)
        map<ll,ll> ret;
        while (criba[n]){
                ret[criba[n]]++;
                n/=criba[n];
        }
        if(n>1) ret[n]++;
        return ret;
}

int main() {
        buscarprimos();
        return 0;
}
```

Rho

```
ll gcd(ll a, ll b){return a?gcd(b%a,a):b;}

ull mulmod(ull a, ull b, ull m){ // 0 <= a, b < m
   long double x; ull c; ll r;
   x = a; c = x * b / m;
   r = (ll)(a * b - c * m) % (ll)m;
   return r < 0 ? r + m : r;
}

ll rho(ll n){
    if(!(n&1))return 2;
    ll x=2,y=2,d=1;
    ll c=rand()%n+1;
    while(d==1){
        x=(mulmod(x,x,n)+c)%n;
        y=(mulmod(y,y,n)+c)%n;
        y=(mulmod(y,y,n)+c)%n;
        if(x>=y)d=gcd(x-y,n);
        else d=gcd(y-x,n);
    }
    return d==n?rho(n):d;
}

void fact(ll n, map<ll,int>& f){ //O (lg n)^3
        if(n==1)return;
        if(rabin(n)){f[n]++;return;}
        ll q=rho(n);fact(q,f);fact(n/q,f);
}
```

4. Grafos

BFS List

```
const int MAXN=10010;
vector<ll> g[MAXN];

vector<int> BFS(int nodoInicial, int n){
        int t;
        queue<int> cola;
        vector<int> distancias(n,n);
        cola.push(nodoInicial);
        distancias[nodoInicial] = 0;
        while(!cola.empty()){
                t = cola.front();
                cola.pop();
                for(unsigned int i = 0; i < g[t].size(); i
                    ++){
                        if(distancias[g[t][i]] == n){
                                distancias[g[t][i]] =
                                    distancias[t]+1;
                                cola.push(g[t][i]);
                        }
                }
        }
        return distancias;
}
```

BFS Tablero

```
#define INF 1000000007
int n,m;
bool a[8][8];
int dist[8][8];
int CANT_MOVE = 4;
pair<int,int> mov[CANT_MOVE];

void inimove(void){
    mov[0].x = 0; mov[0].y = 1;
    mov[1].x = 0; mov[1].y = 0;
    mov[2].x = 1; mov[2].y = 1;
    mov[3].x = 1; mov[3].y = 0;
}

bool ok(pair<int,int> movement, pair<int,int> a){
    return movement.x+a.x>=0 and movement.x+a.x<n and
       movement.y+a.y>=0 and movement.y+a.y<m;
}

void bfs(void){
    bool visit[8][8];
    int nivel[8][8];
    pair<int,int> padre[8][8];
    memset(visit,false,sizeof(visit));
    queue<pair<int,int> > q;
    for(int i=0;i<n;i++) for(int j=0;j<m;j++) if(a[i][j]){
        pair<int,int> node = make_pair(i,j);
        q.push(make_pair(i,j));
        nivel[node.x][node.y] = -1;
        padre[node.x][node.y] = node;
        visit[node.x][node.y] = true;
    }
    while(q.size()){
        pair<int, int> current = q.front();
        q.pop();
        pair<int,int> mi_padre = padre[current.x][current.y
           ];
        nivel[current.x][current.y] = nivel[mi_padre.x][
           mi_padre.y]+1;
        dist[current.x][current.y] = min(dist[current.x][
           current.y],nivel[current.x][current.y]);
        for(int i=0;i<CANT_MOVE;i++){
            if(ok(mov[i],current)){
                pair<int,int> vecino =make_pair(mov[i].x+
                   current.x,mov[i].y+current.y);
                if(!visit[vecino.x][vecino.y]){
                    q.push(vecino);
                    padre[vecino.x][vecino.y] = current;
                    visit[vecino.x][vecino.y] = true;
                }
            }
        }
    }
}

int main(){
    inimove();
    forn(i,8) forn(j,8) dist[i][j] = INF;
    n = m = 8;
    bfs();
}
```

BFS Tablero Caballo

```
#define INF 1000000007
int n,m;
bool a[8][8];
int dist[8][8];
int CANT_MOVE = 8;
pair<int,int> mov[CANT_MOVE];

void inimove(void){
    mov[0].x = -1; mov[0].y = 2;
    mov[1].x = 1; mov[1].y = 2;
    mov[2].x = -2; mov[2].y = 1;
    mov[3].x = 2; mov[3].y = 1;
    mov[4].x = -1; mov[4].y = -2;
    mov[5].x = 1; mov[5].y = -2;
    mov[6].x = 2; mov[6].y = -1;
    mov[7].x = -2; mov[7].y = -1;
}

bool ok(pair<int,int> movement, pair<int,int> a){
    return movement.x+a.x>=0 and movement.x+a.x<n and
       movement.y+a.y>=0 and movement.y+a.y<m;
}

void bfs(void){
    bool visit[8][8];
    int nivel[8][8];
    pair<int,int> padre[8][8];
    memset(visit,false,sizeof(visit));
    queue<pair<int,int> > q;
    for(int i=0;i<n;i++) for(int j=0;j<m;j++) if(a[i][j]){
        pair<int,int> node = make_pair(i,j);
        q.push(make_pair(i,j));
        nivel[node.x][node.y] = -1;
        padre[node.x][node.y] = node;
        visit[node.x][node.y] = true;
    }
    while(q.size()){
        pair<int, int> current = q.front();
        q.pop();
        pair<int,int> mi_padre = padre[current.x][current.y
           ];
        nivel[current.x][current.y] = nivel[mi_padre.x][
           mi_padre.y]+1;
        dist[current.x][current.y] = min(dist[current.x][
           current.y],nivel[current.x][current.y]);
        for(int i=0;i<CANT_MOVE;i++){
            if(ok(mov[i],current)){
                pair<int,int> vecino =make_pair(mov[i].x+
                    current.x,mov[i].y+current.y);
                if(!visit[vecino.x][vecino.y]){
                    q.push(vecino);
                    padre[vecino.x][vecino.y] = current;
                    visit[vecino.x][vecino.y] = true;
                }
            }
        }
    }
}

int main(){
    inimove();
    forn(i,8) forn(j,8) dist[i][j] = INF;
    n = m = 8;
    bfs();
}
```

Dijkstra Olaf

```
vector<pair<int,int> > g[MAXN];  // u->[(v,cost)]
ll dist[MAXN];
void dijkstra(int x){
        memset(dist,-1,sizeof(dist));
        priority_queue<pair<ll,int> > q;
        dist[x]=0;q.push(mp(0,x));
        while(!q.empty()){
                x=q.top().snd;ll c=-q.top().fst;q.pop();
                if(dist[x]!=c)continue;
                forn(i,g[x].size()){
                        int y=g[x][i].fst,c=g[x][i].snd;
                        if(dist[y]<0||dist[x]+c<dist[y])
                                dist[y]=dist[x]+c,q.push(mp
                                    (-dist[y],y));
                }
        }
}
```

Floyd Warshall Olaf

```
// g[i][j]: weight of edge (i, j) or INF if there's no edge
// g[i][i]=0
ll g[MAXN][MAXN];int n;
```

```
void floyd(){ // O(n^3) . Replaces g with min distances
	forn(k,n)forn(i,n)if(g[i][k]<INF)forn(j,n)if(g[k][j
	   ]<INF)
		g[i][j]=min(g[i][j],g[i][k]+g[k][j]);
}
bool inNegCycle(int v){return g[v][v]<0;}
bool hasNegCycle(int a, int b){ // true iff there's neg
    cycle in between
	forn(i,n)if(g[a][i]<INF&&g[i][b]<INF&&g[i][i]<0)
	   return true;
	return false;
}
```

Kruskal Olaf

```
int uf[MAXN];
void uf_init(){memset(uf,-1,sizeof(uf));}
int uf_find(int x){return uf[x]<0?x:uf[x]=uf_find(uf[x]);}
bool uf_join(int x, int y){
	x=uf_find(x);y=uf_find(y);
	if(x==y)return false;
	if(uf[x]>uf[y])swap(x,y);
	uf[x]+=uf[y];uf[y]=x;
	return true;
}
vector<pair<ll,pair<int,int> > > es; // edges (cost,(u,v))
ll kruskal(){  // assumes graph is connected
	sort(es.begin(),es.end());uf_init();
	ll r=0;
	forn(i,es.size()){
		int x=es[i].snd.fst,y=es[i].snd.snd;
		if(uf_join(x,y))r+=es[i].fst; // (x,y,c)
		    belongs to mst
	}
	return r; // total cost
}
```

5. Maths

Combinatoria

```
const int MAXN = 200;

#define MOD 1000000007;

//Luego de llamar a combinatoria(), comb queda: (M DULO
    10^9 +7)
//comb[i][k]= i tomados de a k
long long comb[MAXN+1][MAXN+2];

/* combinatoria : Void
 * Genera el triangulo de pascal hasta la fila MAXN
 * Como a partir de la fila ~60 los valores no entran en un
     long long
 * se guarda la informaci n m dulo 10^9+7
 * SE LLAMA UNA SOLA VEZ esta funci n
 */
void combinatoria(){
        forn(i, MAXN+1){ //comb[i][k]=i tomados de a k
                comb[i][0]=comb[i][i]=1;
                forr(k, 1, i)
                        comb[i][k]=(comb[i-1][k]+comb[i-1][
                           k-1])%MOD;
        }
}

void outputTrianguloPascal(){
        forn(i, MAXN+1){
                cout << "FILA: " << i << endl;
                forn(k, i){
                        cout << comb[i][k] << "\t";
                }
                cout << endl;
        }
}

int main(){
        combinatoria();
        outputTrianguloPascal();
        return 0;
}
```

Exp Mod

```
ll expmod(ll b, ll e, ll m){
        b%=m;
        if(!e) return 1;
        ll q = expmod(b, e/2, m); q = (q*q) % m;
        return e%2 ? (b*q) % m : q;
}
```

Factorial

```
int factorial(int n){
  return (n == 1 || n == 0) ? 1 : factorial(n - 1) * n;
}
```

Fib

```
long long afib[10000000];
bool b[10000000] = {false};

//O(n)
long long fib(long long n){
        if(n == 1 || n == 2) {
                afib[n] = 1;
                b[n] = true;
                return 1;
        }
        if(b[n]) return afib[n];
        b[n] = true;
        afib[n] = fib(n-1) + afib[n-2];
        return afib[n];
}
```

GCD-MCM

```
ll gcd(ll a, ll b){
  if(b<a) swap(a,b);
  if(a==0) return b;
  else{
    return gcd(b%a,a);
  }
}

ll mcm(ll a, ll b){
  return (a/gcd(a,b))*b;
}
```

GCD Array

```
int GCD(int n, int d[]){
        int a = __gcd(d[0], d[1]);
        forr(i, 1, n-1){
                if(__gcd(d[i], d[i+1]) < a)
                        a = __gcd(d[i], d[i+1]);
        }
        return a;
}
```

6. Primos

Criba

```
#define MAXP 100000     //no necesariamente primo
int criba[MAXP+1];

void crearcriba(){
        int w[] = {4,2,4,2,4,6,2,6};
        for(int p=25;p<=MAXP;p+=10) criba[p]=5;
        for(int p=9;p<=MAXP;p+=6) criba[p]=3;
        for(int p=4;p<=MAXP;p+=2) criba[p]=2;
        for(int p=7,cur=0;p*p<=MAXP;p+=w[cur++&7]) if (!
           criba[p])
                for(int j=p*p;j<=MAXP;j+=(p<<1)) if(!criba[
                   j]) criba[j]=p;
}

vector<int> primos;

void buscarprimos(){
        crearcriba();
        forr (i,2,MAXP+1) if (!criba[i]) primos.push_back(i
           );
}

int main() {
        buscarprimos();
        return 0;
}
```

Is Prime

```
bool is_prime(int m){
  for(int a=2; a*a<=m; a++){
    if(m%a==0)
      return false;
  }
  return true;
}
```

Rabin

```
ull mulmod(ull a, ull b, ull m){ // 0 <= a, b < m
   long double x; ull c; ll r;
   x = a; c = x * b / m;
   r = (ll)(a * b - c * m) % (ll)m;
   return r < 0 ? r + m : r;
}

ll expmod(ll b, ll e, ll m){
        if(!e)return 1;
        ll q=expmod(b,e/2,m);q=mulmod(q,q,m);
        return e&1?mulmod(b,q,m):q;
}
bool is_prime_prob(ll n, int a){
        if(n==a)return true;
        ll s=0,d=n-1;
        while(d%2==0)s++,d/=2;
        ll x=expmod(a,d,n);
        if((x==1)||(x+1==n))return true;
        forn(_,s-1){
                x=mulmod(x,x,n);
                if(x==1)return false;
                if(x+1==n)return true;
        }
        return false;
}
bool rabin(ll n){ // true iff n is prime
        if(n==1)return false;
        int ar[]={2,3,5,7,11,13,17,19,23};
        forn(i,9)if(!is_prime_prob(n,ar[i]))return false;
        return true;
}
```

7. Random

Tiene X

```c
int tieneX(int u, int x){
        int g = u%10;
        if(u%x == 0) return 1;
        while (u > 0){
                g=u%10;
                if(u%10 == x)
                        return 1;
                else
                        u = (u-g)/10;
        }
        return 0;
}
```