Descongelen a Victor Moreno

Contents

1 Estructuras de Datos

1.1 Unordered Map

```cpp
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;

struct custom_hash {
    static uint64_t splitmix64(uint64_t x) {
        // http://xorshift.di.unimi.it/splitmix64.c
        x += 0x9e3779b97f4a7c15;
        x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
        x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
        return x ^ (x >> 31);
    }

    size_t operator()(uint64_t x) const {
        static const uint64_t FIXED_RANDOM = chrono::steady_clock::now().
            time_since_epoch().count();
        return splitmix64(x + FIXED_RANDOM);
    }
};

gp_hash_table<int, int,custom_hash> m1;

//Funcion count
m1.find(x)!=m1.end()
```

1.2 Segment tree Recursivo

```
%%
%% This is file `.tex',
%% generated with the docstrip utility.
%%
%% The original source files were:
%%
%% fileerr.dtx  (with options: `return')
%%
%% This is a generated file.
%%
%% The source is maintained by the LaTeX Project team and bug
%% reports for it can be opened at https://latex-project.org/bugs/
%% (but please observe conditions on bug reports sent to that address!)
```

```
%%
%%
%% Copyright (C) 1993-2021
%% The LaTeX Project and any individual authors listed elsewhere
%% in this file.
%%
%% This file was generated from file(s) of the Standard LaTeX `Tools Bundle
    '.
%%
    ----------------------------------------------------------------------


%%
%% It may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either version 1.3c
%% of this license or (at your option) any later version.
%% The latest version of this license is in
%%    https://www.latex-project.org/lppl.txt
%% and version 1.3c or later is part of all distributions of LaTeX
%% version 2005/12/01 or later.
%%
%% This file may only be distributed together with a copy of the LaTeX
%% `Tools Bundle'. You may however distribute the LaTeX `Tools Bundle'
%% without such generated files.
%%
%% The list of all files belonging to the LaTeX `Tools Bundle' is
%% given in the file `manifest.txt'.
%%
 \message{File ignored}
\endinput
%%
%% End of file `.tex'.
```

1.3 Segment Tree Iterativo

```
//Para procesar querys de tipo k-esimo es necesario crear un arbol binario
    perfector(llenar con 0's)
template<typename T>
struct SegmentTree{
  int N;
  vector<T> ST;

  //Creacion a partir de un arreglo O(n)
  SegmentTree(int N, vector<T> & arr): N(N){
    ST.resize(N << 1);
    for(int i = 0; i < N; ++i)
      ST[N + i] = arr[i];     //Dato normal
      ST[N + i] = creaNodo(); //Dato compuesto
    for(int i = N - 1; i > 0; --i)
      ST[i] = ST[i << 1] + ST[i << 1 | 1];        //Dato normal
      ST[i] = merge(ST[i << 1] , ST[i << 1 | 1]); //Dato compuesto
  }

  //Actualizacion de un elemento en la posicion i
  void update(int i, T value){
    ST[i += N] = value;     //Dato normal
    ST[i += N] = creaNodo();//Dato compuesto
    while(i >>= 1)
      ST[i] = ST[i << 1] + ST[i << 1 | 1];        //Dato normal
      ST[i] = merge(ST[i << 1] , ST[i << 1 | 1]); //Dato compuesto
  }

  //query en [l, r]
  T query(int l, int r){
    T res = 0;  //Dato normal
    nodo resl = creaNodo(), resr = creaNodo();//Dato compuesto
    for(l += N, r += N; l <= r; l >>= 1, r >>= 1){
      if(l & 1)       res += ST[l++]; //Dato normal
      if(!(r & 1))    res += ST[r--]; //Dato normal

      if(l & 1)       resl = merge(resl,ST[l++]); //Dato compuesto
      if(!(r & 1))    resr = merge(ST[r--],resr); //Dato compuesto
    }
    return res;                 //Dato normal
    return merge(resl,resr);    //Dato compuesto
  }

  //Para estas querys es necesario que el st tenga el tam de la siguiente
      potencia de 2
  //ll nT = 1;
  // while(nT<n) nT<<=1;
  //vector<int> a(nT,0);

  //Encontrar k-esimo 1 en un st de 1's
  int Kth_One(int k) {
    int i = 0, s = N >> 1;
    for(int p = 2; p < 2 * N; p <<= 1, s >>= 1) {
```

```
      if(k < ST[p]) continue;
      k -= ST[p++]; i += s;
    }
    return i;
  }

  //i del primer elemento >= k en todo el arr
  int atLeastX(int k){
    int i = 0, s = N >> 1;
    for(int p = 2; p < 2 * N; p <<= 1, s >>= 1) {
      if(ST[p] < k) p++, i += s;
    }
    if(ST[N + i] < k) i = -1;
    return i;
  }

  //i del primer elemento >= k en [l,fin]
  //Uso atLeastX(k,l,1,nT)
  int atLeastX(int x, int l, int p, int s) {
    if(ST[p] < x or s <= l) return -1;
    if((p << 1) >= 2 * N)
      return (ST[p] >= x) - 1;
    int i = atLeastX(x, l, p << 1, s >> 1);
    if(i != -1) return i;
    i = atLeastX(x, l - (s >> 1), p << 1 | 1, s >> 1);
    if(i == -1) return -1;
    return (s >> 1) + i;
  }
};
```

1.4 Segment Tree Lazy Recursivo

```
%%
%% This is file `.tex',
%% generated with the docstrip utility.
%%
%% The original source files were:
%%
%% fileerr.dtx  (with options: `return')
%%
%% This is a generated file.
%%
%% The source is maintained by the LaTeX Project team and bug
%% reports for it can be opened at https://latex-project.org/bugs/
%% (but please observe conditions on bug reports sent to that address!)
%%
%%
%% Copyright (C) 1993-2021
%% The LaTeX Project and any individual authors listed elsewhere
%% in this file.
%%
%% This file was generated from file(s) of the Standard LaTeX `Tools Bundle
    '.
%%
    ----------------------------------------------------------------------


%%
%% It may be distributed and/or modified under the
%% conditions of the LaTeX Project Public License, either version 1.3c
%% of this license or (at your option) any later version.
%% The latest version of this license is in
%%   https://www.latex-project.org/lppl.txt
%% and version 1.3c or later is part of all distributions of LaTeX
%% version 2005/12/01 or later.
%%
%% This file may only be distributed together with a copy of the LaTeX
%% `Tools Bundle'. You may however distribute the LaTeX `Tools Bundle'
%% without such generated files.
%%
%% The list of all files belonging to the LaTeX `Tools Bundle' is
%% given in the file `manifest.txt'.
%%
 \message{File ignored}
\endinput
%%
%% End of file `.tex'.
```

1.5 Segment Tree Lazy Iterativo

```
//Lazy propagation con incremento de u en rango y minimo
//Hay varias modificaciones necesarias para suma en ambos
template<typename T>
struct SegmentTreeLazy{
  int N,h;
  vector<T> ST, d;

```

```
  //Creacion a partir de un arreglo
  SegmentTreeLazy(int n, vector<T> &a): N(n){
    //En caso de inicializar en cero o algo similar, revisar que la
        construccion tenga su respectivo neutro mult y 1
    ST.resize(N << 1);
    d.resize(N);
    h = 64 - __builtin_clzll(n);

    for(int i = 0; i < N; ++i)
      ST[N + i] = a[i];
    //Construir el st sobre la query que se necesita
    for(int i = N - 1; i > 0; --i)
      ST[i] = min(ST[i << 1] , ST[i << 1 | 1]);
  }

  //Modificar de acuerdo al tipo modificacion requerida, +,*,|,^,etc
  void apply(int p, T value) {
    ST[p] += value;
    if(p<N) d[p]+= value;
  }

  // Modifica valores de los padres de p
  //Modificar de acuerdo al tipo modificacion requerida, +,*,|,^,etc y a la
      respectiva query
  void build(int p){
    while(p>1){
      p >>= 1;
      ST[p] = min(ST[p << 1], ST[p << 1 | 1]) + d[p];
      //ST[p] = (ST[p << 1] & ST[p << 1 | 1]) | d[p]; Ejemplos con bitwise
    }
  }

  // Propagacion desde la raiz a p
  void push(int p){
    for (int s = h; s > 0; --s) {
      int i = p >> s;
      if (d[i] != 0) {
        apply(i << 1, d[i]);
        apply(i << 1 | 1, d[i]);
        d[i] = 0;   //Tener cuidado si estoy haciendo multiplicaciones
      }
    }
  }


  // Sumar v a cada elemento en el intervalo [l, r)
  void increment(int l, int r, T value) {
    l += N, r += N;
    int l0 = l, r0 = r;
    for (; l < r; l >>= 1, r >>= 1) {
      if(l & 1) apply(l++, value);
      if(r & 1) apply(--r, value);
    }
    build(l0);
    build(r0 - 1);
  }

  // min en el intervalo [l, r)
  T range_min(int l, int r) {
    l += N, r += N;
    push(l);
    push(r - 1);
    T res = LLONG_MAX;
    //T res = (1 << 30) - 1;    Requerir operacion and
    for (; l < r; l >>= 1, r >>= 1) {
      if(l & 1) res = min(res, ST[l++]);
      //if(res >= mod) res -= mod;
      if(r & 1) res = min(res, ST[--r]);
      //if(res >= mod) res -= mod;
    }
    return res;
  }

};
```

1.6 Rope

```
#include <ext/rope>
using namespace __gnu_cxx;
rope<int> s;
// Sequence with O(log(n)) random access, insert, erase at any position
// s.push_back(x);
// s.insert(i,r) // insert rope r at position i
// s.erase(i,k) // erase subsequence [i,i+k)
// s.substr(i,k) // return new rope corresponding to subsequence [i,i+k)
// s[i] // access ith element (cannot modify)
// s.mutable_reference_at(i) // acces ith element (allows modification)
```

```
// s.begin() and s.end() are const iterators (use mutable_begin(),
    mutable_end() to allow modification)
```

1.7 Ordered Set

```
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;
// find_by_order(i) -> iterator to ith element
// order_of_key(k) -> position (int) of lower_bound of k
```

1.8 Union Find

```
vector<pair<int,int>>ds(MAX,{-1,0});
// Solo siu requeires los elementos del union find, utiliza
// dsext en caso contrario borrarlo
list<int>dsext[MAX];
void init(int n){
    for(int i=0;i<n;i++)dsext[i].push_back(i);
}
int find(int x){
    if(-1==ds[x].first) return x;
    return ds[x].first=find(ds[x].first);
}
bool unionDs(int x, int y){
    int px=find(x),py=find(y);
    int &rx=ds[px].second,&ry=ds[py].second;
    if(px==py) return false;
    else{
        if(rx>ry){
            ds[py].first=px;
        }
        else{
            ds[px].first=py;
            if(rx==ry) ry+=1;
        }
    }
    return true;
}
```

2 Varios

2.1 Template

```
#include<bits/stdc++.h>
using namespace std;

#define forn(i,n)       for(int i=0; i<n; i++)
#define forr(i,a,n)     for(int i=a; i<n; i++)
#define fore(i,a,n)     for(int i=a; i<=n; i++)
#define each(a,b)       for(auto a: b)
#define all(v)          v.begin(),v.end()
#define sz(a)           (int)a.size()
#define debln(a)        cout << a << "\n"
#define deb(a)          cout << a << " "
#define pb              push_back

typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> ii;

void sol(){

}

int main(){
    ios::sync_with_stdio(false);cin.tie(0);

    int t=1;
    cin>>t;
    while(t--){
        sol();
    }

    return 0;
}
```

2.2 String a vector<int>

```
//Convertir una cadena de numeros separados por " " en vector de enteros
//Leer varias de esas querys
cin.ignore();
while(q--){
```

```
  string s;
  getline(cin, s);
  vector<int> qr;
  stringstream ss(s);
  int num;
  while (ss >> num)   qr.push_back(num);
}
```

2.3 Generar permutaciones

```
//Generar todas las permutaciones de un arreglo
sort(all(a));
do{
  //hacer lo que quieras con la perm generada
}while(next_permutation(all(a)));
```