.
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!)
```

1.3 Segment Tree Iterativo

```
1  //Para procesar querys de tipo k-esimo es necesario crear un arbol binario
       perfector(llenar con 0's)
2  template<typename T>
3  struct SegmentTree{
4    int N;
5    vector<T> ST;
6
7    //Creacion a partir de un arreglo O(n)
8    SegmentTree(int N, vector<T> & arr): N(N){
9      ST.resize(N << 1);
10     for(int i = 0; i < N; ++i)
11       ST[N + i] = arr[i];        //Dato normal
12       ST[N + i] = creaNodo(); //Dato compuesto
13     for(int i = N - 1; i > 0; --i)
14       ST[i] = ST[i << 1] + ST[i << 1 | 1];         //Dato normal
15       ST[i] = merge(ST[i << 1] , ST[i << 1 | 1]); //Dato compuesto
16   }
17
18   //Actualizacion de un elemento en la posicion i
19   void update(int i, T value){
20     ST[i += N] = value;       //Dato normal
21     ST[i += N] = creaNodo();//Dato compuesto
22     while(i >>= 1)
23       ST[i] = ST[i << 1] + ST[i << 1 | 1];         //Dato normal
24       ST[i] = merge(ST[i << 1] , ST[i << 1 | 1]); //Dato compuesto
25   }
26
27   //query en [l, r]
28   T query(int l, int r){
29     T res = 0;  //Dato normal
30     nodo resl = creaNodo(), resr = creaNodo();//Dato compuesto
31     for(l += N, r += N; l <= r; l >>= 1, r >>= 1){
32       if(l & 1)        res += ST[l++]; //Dato normal
33       if(!(r & 1))     res += ST[r--]; //Dato normal
34
35       if(l & 1)        resl = merge(resl,ST[l++]); //Dato compuesto
36       if(!(r & 1))     resr = merge(ST[r--],resr); //Dato compuesto
37     }
38     return res;                //Dato normal
39     return merge(resl,resr);   //Dato compuesto
40   }
41
42   //Para estas querys es necesario que el st tenga el tam de la siguiente
        potencia de 2
43   //ll nT = 1;
44   // while(nT<n) nT<<=1;
45   //vector<int> a(nT,0);
46
47   //Encontrar k-esimo 1 en un st de 1's
48   int Kth_One(int k) {
49     int i = 0, s = N >> 1;
50     for(int p = 2; p < 2 * N; p <<= 1, s >>= 1) {
```

```
51        if(k < ST[p]) continue;
52        k -= ST[p++]; i += s;
53      }
54      return i;
55    }
56
57    //i del primer elemento >= k en todo el arr
58    int atLeastX(int k){
59      int i = 0, s = N >> 1;
60      for(int p = 2; p < 2 * N; p <<= 1, s >>= 1) {
61        if(ST[p] < k) p++, i += s;
62      }
63      if(ST[N + i] < k) i = -1;
64      return i;
65    }
66
67    //i del primer elemento >= k en [l,fin]
68    //Uso atLeastX(k,l,1,nT)
69    int atLeastX(int x, int l, int p, int s) {
70      if(ST[p] < x or s <= l) return -1;
71      if((p << 1) >= 2 * N)
72        return (ST[p] >= x) - 1;
73      int i = atLeastX(x, l, p << 1, s >> 1);
74      if(i != -1) return i;
75      i = atLeastX(x, l - (s >> 1), p << 1 | 1, s >> 1);
76      if(i == -1) return -1;
77      return (s >> 1) + i;
78    }
79  };
```

1.4   Segment Tree Lazy Recursivo

```
1  %%
2  %% This is file `.tex',
3  %% generated with the docstrip utility.
4  %%
5  %% The original source files were:
6  %%
7  %% fileerr.dtx  (with options: `return')
8  %%
9  %% This is a generated file.
10 %%
11 %% The source is maintained by the LaTeX Project team and bug
12 %% reports for it can be opened at https://latex-project.org/bugs/
13 %% (but please observe conditions on bug reports sent to that address!)
14 %%
15 %%
16 %% Copyright (C) 1993-2021
17 %% The LaTeX Project and any individual authors listed elsewhere
18 %% in this file.
19 %%
20 %% This file was generated from file(s) of the Standard LaTeX `Tools Bundle
       '.
21 %%
       -----------------------------------------------------------------------

22 %%
23 %% It may be distributed and/or modified under the
24 %% conditions of the LaTeX Project Public License, either version 1.3c
25 %% of this license or (at your option) any later version.
26 %% The latest version of this license is in
27 %%    https://www.latex-project.org/lppl.txt
28 %% and version 1.3c or later is part of all distributions of LaTeX
29 %% version 2005/12/01 or later.
30 %%
31 %% This file may only be distributed together with a copy of the LaTeX
32 %% `Tools Bundle'. You may however distribute the LaTeX `Tools Bundle'
33 %% without such generated files.
34 %%
35 %% The list of all files belonging to the LaTeX `Tools Bundle' is
36 %% given in the file `manifest.txt'.
37 %%
38  \message{File ignored}
39 \endinput
40 %%
41 %% End of file `.tex'.
```

1.5   Segment Tree Lazy Iterativo

```
1  //Lazy propagation con incremento de u en rango y minimo
2  //Hay varias modificaciones necesarias para suma en ambos
3  template<typename T>
4  struct SegmentTreeLazy{
5    int N,h;
6    vector<T> ST, d;
7
```

```
8     //Creacion a partir de un arreglo
9     SegmentTreeLazy(int n, vector<T> &a): N(n){
10        //En caso de inicializar en cero o algo similar, revisar que la
              construccion tenga su respectivo neutro mult y 1
11        ST.resize(N << 1);
12        d.resize(N);
13        h = 64 - __builtin_clzll(n);
14
15        for(int i = 0; i < N; ++i)
16          ST[N + i] = a[i];
17        //Construir el st sobre la query que se necesita
18        for(int i = N - 1; i > 0; --i)
19          ST[i] = min(ST[i << 1] , ST[i << 1 | 1]);
20      }
21
22    //Modificar de acuerdo al tipo modificacion requerida, +,*,|,^,etc
23    void apply(int p, T value) {
24        ST[p] += value;
25        if(p<N) d[p]+= value;
26    }
27
28    // Modifica valores de los padres de p
29    //Modificar de acuerdo al tipo modificacion requerida, +,*,|,^,etc y a la
              respectiva query
30    void build(int p){
31        while(p>1){
32          p >>= 1;
33          ST[p] = min(ST[p << 1], ST[p << 1 | 1]) + d[p];
34          //ST[p] = (ST[p << 1] & ST[p << 1 | 1]) | d[p]; Ejemplos con bitwise
35        }
36    }
37
38    // Propagacion desde la raiz a p
39    void push(int p){
40        for (int s = h; s > 0; --s) {
41          int i = p >> s;
42          if (d[i] != 0) {
43            apply(i << 1, d[i]);
44            apply(i << 1 | 1, d[i]);
45            d[i] = 0;   //Tener cuidado si estoy haciendo multiplicaciones
46          }
47        }
48    }
```

```
49
50    // Sumar v a cada elemento en el intervalo [l, r)
51    void increment(int l, int r, T value) {
52        l += N, r += N;
53        int l0 = l, r0 = r;
54        for (; l < r; l >>= 1, r >>= 1) {
55          if(l & 1) apply(l++, value);
56          if(r & 1) apply(--r, value);
57        }
58        build(l0);
59        build(r0 - 1);
60    }
61
62    // min en el intervalo [l, r)
63    T range_min(int l, int r) {
64        l += N, r += N;
65        push(l);
66        push(r - 1);
67        T res = LLONG_MAX;
68        //T res = (1 << 30) - 1;    Requerir operacion and
69        for (; l < r; l >>= 1, r >>= 1) {
70          if(l & 1) res = min(res, ST[l++]);
71          //if(res >= mod) res -= mod;
72          if(r & 1) res = min(res, ST[--r]);
73          //if(res >= mod) res -= mod;
74        }
75        return res;
76    }
77
78  };
```

1.6   Rope

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

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

1.7 Ordered Set

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

1.8 Union Find

```
1  vector<pair<int,int>>ds(MAX,{-1,0});
2  // Solo siu requeires los elementos del union find, utiliza
3  // dsext en caso contrario borrarlo
4  list<int>dsext[MAX];
5  void init(int n){
6      for(int i=0;i<n;i++)dsext[i].push_back(i);
7  }
8  int find(int x){
9      if(-1==ds[x].first) return x;
10     return ds[x].first=find(ds[x].first);
11 }
12 bool unionDs(int x, int y){
13     int px=find(x),py=find(y);
14     int &rx=ds[px].second,&ry=ds[py].second;
15     if(px==py) return false;
16     else{
17         if(rx>ry){
18             ds[py].first=px;
19         }
20         else{
21             ds[px].first=py;
22             if(rx==ry) ry+=1;
23         }
24     }
25     return true;
26 }
```

1.9 Segment Tree Persistente

```
1  #define inf INT_MAX
2  const int MAX=5e5+2;
3  typedef pair<ll, ll> item;
4  struct node{
5      item val;
6      node *l, *r;
7      node(): l(nullptr),r(nullptr),val({inf,inf}){};
8      node(node *_l,node *_r):l(_l),r(_r){
9          val=min(l->val,r->val);
10     }
11     node(ll value,ll pos):r(nullptr),l(nullptr){
12         val=make_pair(value,pos);
13     }
14 };
15 pair<ll,ll>all;
16 vector<node*>versions(MAX,nullptr);
17 node* build(int l,int r){
18     if(l==r)return  new node(inf,l);
19     int m=(l+r)/2;
20     return new node(build(l,m),build(m+1,r));
21 }
22
23 node* update(node *root,int l,int r,int pos,int val){
24     if(l==r){
25         return new node(val,pos);}
26     int m=(l+r)/2;
27     if(pos<=m) return new node(update(root->l,l,m,pos,val),root->r);
28     return new node(root->l,update(root->r,m+1,r,pos,val));
29 }
30 item query(node *root,int l,int r,int a,int b){
31     if(a>r || b<l) return all;
32     if(a<=l && r<=b) return root->val;
33     int m=(l+r)/2;
34     return min(query(root->l,l,m,a,b),query(root->r,m+1,r,a,b));
35 }
```

1.10 Sparce Table

```
1  //Se usa para RMQ porque se puede hacer en O(1), no acepta updates
2  vector<int>lg;
3  vector<vector<int>>st;
4  int *nums;
5  void init(int n){
```

```
6      int logn=(int) log2(n)+1;
7      lg.assign(n+1,0);
8      st.assign(logn,vector<int>(n+1));
9      for(int i=0;i<n;i++) st[0][i]=nums[i];
10     lg[1]=0;
11     for(int i=2;i<=n;i++) lg[i]=lg[i/2]+1;
12     for(int i=1;i<logn;i++)
13         for(int j=0;j+(1<<i)<n;j++)st[i][j]=min(st[i-1][j],st[i-1][j+(1<<(i
               -1))]);
14  }
15  int query(int a,int b){
16      int logn=lg[(b-a+1)];
17      cout<<st[logn][a]<<endl;
18      return min(st[logn][a],st[logn][b-(1<<logn)+1]);
19  }
```

1.11   Walvet Tree

```
1  // indexed in 1
2  // from pointer to first element and to to end
3  // x and y The minimun element and y the max element
4  // If you need only one function or more erase the others
5  // If you need tu construct other function you only required to undertand
       the limit, this
6  // are the same
7  struct wavelet_tree{
8    int lo, hi;
9    wavelet_tree *l, *r;
10   vector<int> b;
11   wavelet_tree(int *from, int *to, int x, int y){
12     lo = x, hi = y;
13     if(lo == hi or from >= to) return;
14     int mid = (lo+hi)/2;
15     auto f = [mid](int x){ return x <= mid;};
16     b.reserve(to-from+1);
17     b.pb(0);
18     for(auto it = from; it != to; it++)
19       b.push_back(b.back() + f(*it));
20     auto pivot = stable_partition(from, to, f);
21     l = new wavelet_tree(from, pivot, lo, mid);
22     r = new wavelet_tree(pivot, to, mid+1, hi);
23   }
24   //kth smallest element in [l, r]
```

```
25   int kth(int l, int r, int k){
26     if(l > r) return 0;
27     if(lo == hi) return lo;
28     int inLeft = b[r] - b[l-1];
29     int lb = b[l-1];
30     int rb = b[r];
31     if(k <= inLeft) return this->l->kth(lb+1, rb , k);
32     return this->r->kth(l-lb, r-rb, k-inLeft);
33   }
34   //count of nos in [l, r] Less than or equal to k
35   int LTE(int l, int r, int k) {
36     if(l > r or k < lo) return 0;
37     if(hi <= k) return r - l + 1;
38     int lb = b[l-1], rb = b[r];
39     return this->l->LTE(lb+1, rb, k) + this->r->LTE(l-lb, r-rb, k);
40   }
41   //count of nos in [l, r] equal to k
42   int count(int l, int r, int k) {
43     if(l > r or k < lo or k > hi) return 0;
44     if(lo == hi) return r - l + 1;
45     int lb = b[l-1], rb = b[r], mid = (lo+hi)/2;
46     if(k <= mid) return this->l->count(lb+1, rb, k);
47     return this->r->count(l-lb, r-rb, k);
48   }
49 };
```

2   Strings

2.1   Aho Corasick

```
1  int K, I = 1;
2  struct node {
3      int fail, ch[26] = {};
4      vector<int> lens;
5  } T[500005];
6
7  void add(string s) {
8      int x = 1;
9      for (int i = 0; i < s.size(); i++) {
10         if (T[x].ch[s[i] - 'a'] == 0)
11             T[x].ch[s[i] - 'a'] = ++I;
12         x = T[x].ch[s[i] - 'a'];
13     }
```

```
14        T[x].lens.PB(s.size());
15  }
16
17  void build() {
18      queue<int> Q;
19      int x = 1;
20      T[1].fail = 1;
21      for (int i = 0; i < 26; i++) {
22          if (T[x].ch[i])
23              T[T[x].ch[i]].fail = x, Q.push(T[x].ch[i]);
24          else
25              T[x].ch[i] = 1;
26      }
27      while (!Q.empty()) {
28          x = Q.front(); Q.pop();
29          for (int i = 0; i < 26; i++) {
30              if (T[x].ch[i])
31                  T[T[x].ch[i]].fail = T[T[x].fail].ch[i], Q.push(T[x].ch[i])
                        ;
32              else
33                  T[x].ch[i] = T[T[x].fail].ch[i];
34          }
35      }
36  }
```

2.2   Hashing

```
1  struct Hash{
2    const int mod=1e9+123;
3    const int p=257;
4    vector<int> prefix;
5    static vector<int>pow;
6    Hash(string str){
7      int n=str.size();
8      while(pow.size()<=n){
9        pow.push_back(1LL*pow.back()*p%mod);
10     }
11     vector<int> aux(n+1);
12     prefix=aux;
13     for(int i=0;i<n;i++){
14       prefix[i+1]=(prefix[i]+1LL*str[i]*pow[i])%mod;
15     }
16   }
```

```
17    inline int getHashInInerval(int i,int len,int MxPow){
18      int hashing=prefix[i+len]-prefix[i];
19      if(hashing<0) hashing+=mod;
20      hashing=1LL*hashing*pow[MxPow-(len+i-1)]%mod;
21      return hashing;
22    }
23  };
24  vector<int> Hash::pow{1};
```

2.3   KMP

```
1  vector<int> kmp(string s){
2      int n=s.size();
3      vector<int>pi(n);
4      for(int i=1;i<n;i++){
5          int j=pi[i-1];
6          while(j>0 && s[i]!=s[j])j=pi[j-1];
7          if(s[i]==s[j]) j++;
8          pi[i]=j;
9      }
10     return pi;
11 }
```

2.4   Manacher

```
1  vector<int> manacher_odd(string s) {
2      int n = s.size();
3      s = "$" + s + "^";
4      vector<int> p(n + 2);
5      int l = 1, r = 1;
6      for(int i = 1; i <= n; i++) {
7          p[i] = max(0, min(r - i, p[l + (r - i)]));
8          while(s[i - p[i]] == s[i + p[i]]) {
9              p[i]++;
10         }
11         if(i + p[i] > r) {
12             l = i - p[i], r = i + p[i];
13         }
14     }
15     return vector<int>(begin(p) + 1, end(p) - 1);
16 }
17 vector<int> manacher_even(string s){
18     string even;
19     for(auto c:s){
```

```
20        even+='#'+c;
21      }
22      even+='#';
23      return manacher_odd(even);
24  }
```

2.5   Suffix Automata

```
1   struct node{
2     map<char,int>edges;
3     int link,length,terminal=0;
4     node(int link,int length): link(link),length(length){};
5   };vector<node>sa;
6   // init in main with sa.push_back(node(-1,0));
7   int last=0;
8   // add one by one chars in order
9   void addChar(char s, int pos){
10      sa.push_back(node(0,pos+1));
11      int r=sa.size()-1;
12      int p=last;
13      while(p >= 0 && sa[p].edges.find(s) == sa[p].edges.end()) {
14        sa[p].edges[s] = r;
15        p = sa[p].link;
16      }
17      if(p != -1) {
18        int q = sa[p].edges[s];
19        if(sa[p].length + 1 == sa[q].length) {
20          sa[r].link = q;
21        } else {
22          sa.push_back(node(sa[q].link,sa[p].length+1));
23          sa[sa.size()-1].edges=sa[q].edges;
24          int qq = sa.size()-1;
25          sa[q].link = qq;
26          sa[r].link= qq;
27          while(p >= 0 && sa[p].edges[s] == q) {
28            sa[p].edges[s] = qq;
29            p = sa[p].link;
30          }
31        }
32      }
33      last = r;
34  }
35  // Not necesary functions
```

```
36  void findTerminals(){
37      int p = last;
38      while(p > 0) {
39        sa[p].terminal=1;
40        p = sa[p].link;
41      }
42  }
```

2.6   Trie

```
1   struct trie{
2       int len,id;
3       int children[26];
4       trie(int _id){
5           len=0,id=_id;
6           for(int i=0;i<26;i++)children[i]=-1;
7       }
8   };vector<trie>Trie;Trie.push_back(trie());
9   void inserString(string str,int root){
10      int aux=root;
11      for(int i=0;i<str.size();i++){
12          int index=str[i]-'a';
13          if(Trie[aux].children[index]==-1){
14              Trie.push_back(trie(Trie.size()));
15              Trie[aux].children[index]=Trie.size()-1;
16          }
17          aux=Trie[aux].children[index];
18      }
19      Trie[aux].len=str.size();
20  }
21  bool existInTrie(string str,int root){
22      int aux=root;
23      for(int i=0;i<str.size();i++){
24          int index=str[i]-'a';
25          if(Trie[aux].children[index]==-1) return false;
26          aux=Trie[aux].children[index];
27      }
28      return Trie[aux].len;
29  }
```

3   Geometria

3.1   Puntos y lineas

```
1  using ld = long double;
2  const ld eps = 1e-9, inf = numeric_limits<ld>::max(), pi = acos(-1);
3  // For use with integers, just set eps=0 and everything remains the same
4  bool geq(ld a, ld b){return a-b >= -eps;}      //a >= b
5  bool leq(ld a, ld b){return b-a >= -eps;}      //a <= b
6  bool ge(ld a, ld b){return a-b > eps;}         //a > b
7  bool le(ld a, ld b){return b-a > eps;}         //a < b
8  bool eq(ld a, ld b){return abs(a-b) <= eps;}   //a == b
9  bool neq(ld a, ld b){return abs(a-b) > eps;}   //a != b
10
11 struct point{
12   ld x, y;
13   point(): x(0), y(0){}
14   point(ld x, ld y): x(x), y(y){}
15
16   point operator+(const point & p) const{return point(x + p.x, y + p.y);}
17   point operator-(const point & p) const{return point(x - p.x, y - p.y);}
18   point operator*(const ld & k) const{return point(x * k, y * k);}
19   point operator/(const ld & k) const{return point(x / k, y / k);}
20
21   point operator+=(const point & p){*this = *this + p; return *this;}
22   point operator-=(const point & p){*this = *this - p; return *this;}
23   point operator*=(const ld & p){*this = *this * p; return *this;}
24   point operator/=(const ld & p){*this = *this / p; return *this;}
25
26   point rotate(const ld & a) const{return point(x*cos(a) - y*sin(
           a), x*sin(a) + y*cos(a));}
27   point perp() const{return point(-y, x);}
28   ld ang() const{
29     ld a = atan2l(y, x); a += le(a, 0) ? 2*pi : 0; return a;
30   }
31   ld dot(const point & p) const{return x * p.x + y * p.y;}
32   ld cross(const point & p) const{return x * p.y - y * p.x;}
33   ld norm() const{return x * x + y * y;}
34   ld length() const{return sqrtl(x * x + y * y);}
35   point unit() const{return (*this) / length();}
36
37   bool operator==(const point & p) const{return eq(x, p.x) && eq(y, p.y);}
38   bool operator!=(const point & p) const{return !(*this == p);}
39   bool operator<(const point & p) const{return le(x, p.x) || (eq(x, p.x) &&
           le(y, p.y));}
40   bool operator>(const point & p) const{return ge(x, p.x) || (eq(x, p.x) &&
           ge(y, p.y));}
41   bool half(const point & p) const{return le(p.cross(*this), 0) || (eq(p.
           cross(*this), 0) && le(p.dot(*this), 0));}
42 };
43
44 istream &operator>>(istream &is, point & p){return is >> p.x >> p.y;}
45 ostream &operator<<(ostream &os, const point & p){return os << "(" << p.x
           << ", " << p.y << ")";}
46
47 int sgn(ld x){
48   if(ge(x, 0)) return 1;
49   if(le(x, 0)) return -1;
50   return 0;
51 }
52
53 void polarSort(vector<point> & P, const point & o, const point & v){
54   //sort points in P around o, taking the direction of v as first angle
55   sort(P.begin(), P.end(), [&](const point & a, const point & b){
56     return point((a - o).half(v), 0) < point((b - o).half(v), (a - o).cross
           (b - o));
57   });
58 }
59
60 bool pointInLine(const point & a, const point & v, const point & p){
61   //line a+tv, point p
62   return eq((p - a).cross(v), 0);
63 }
64
65 bool pointInSegment(const point & a, const point & b, const point & p){
66   //segment ab, point p
67   return pointInLine(a, b - a, p) && leq((a - p).dot(b - p), 0);
68 }
69
70 int intersectLinesInfo(const point & a1, const point & v1, const point & a2
           , const point & v2){
71   //lines a1+tv1 and a2+tv2
72   ld det = v1.cross(v2);
73   if(eq(det, 0)){
74     if(eq((a2 - a1).cross(v1), 0)){
75       return -1; //infinity points
76     }else{
77       return 0; //no points
78     }
79   }else{
```

```
80      return 1; //single point
81    }
82  }
83
84  point intersectLines(const point & a1, const point & v1, const point & a2,
        const point & v2){
85    //lines a1+tv1, a2+tv2
86    //assuming that they intersect
87    ld det = v1.cross(v2);
88    return a1 + v1 * ((a2 - a1).cross(v2) / det);
89  }
90
91  int intersectLineSegmentInfo(const point & a, const point & v, const point
        & c, const point & d){
92    //line a+tv, segment cd
93    point v2 = d - c;
94    ld det = v.cross(v2);
95    if(eq(det, 0)){
96      if(eq((c - a).cross(v), 0)){
97        return -1; //infinity points
98      }else{
99        return 0; //no point
100     }
101   }else{
102     return sgn(v.cross(c - a)) != sgn(v.cross(d - a)); //1: single point,
            0: no point
103   }
104 }
105
106 int intersectSegmentsInfo(const point & a, const point & b, const point & c
        , const point & d){
107   //segment ab, segment cd
108   point v1 = b - a, v2 = d - c;
109   int t = sgn(v1.cross(c - a)), u = sgn(v1.cross(d - a));
110   if(t == u){
111     if(t == 0){
112       if(pointInSegment(a, b, c) || pointInSegment(a, b, d) ||
              pointInSegment(c, d, a) || pointInSegment(c, d, b)){
113         return -1; //infinity points
114       }else{
115         return 0; //no point
116       }
117     }else{
```

```
118       return 0; //no point
119     }
120   }else{
121     return sgn(v2.cross(a - c)) != sgn(v2.cross(b - c)); //1: single point,
            0: no point
122   }
123 }
124
125 ld distancePointLine(const point & a, const point & v, const point & p){
126   //line: a + tv, point p
127   return abs(v.cross(p - a)) / v.length();
128 }
```

3.2   Circulos

```
1   ld distancePointCircle(const point & c, ld r, const point & p){
2     //point p, circle with center c and radius r
3     return max((ld)0, (p - c).length() - r);
4   }
5
6   point projectionPointCircle(const point & c, ld r, const point & p){
7     //point p (outside the circle), circle with center c and radius r
8     return c + (p - c).unit() * r;
9   }
10
11  pair<point, point> pointsOfTangency(const point & c, ld r, const point & p)
        {
12    //point p (outside the circle), circle with center c and radius r
13    point v = (p - c).unit() * r;
14    ld d2 = (p - c).norm(), d = sqrt(d2);
15    point v1 = v * (r / d), v2 = v.perp() * (sqrt(d2 - r*r) / d);
16    return {c + v1 - v2, c + v1 + v2};
17  }
18
19  vector<point> intersectLineCircle(const point & a, const point & v, const
        point & c, ld r){
20    //line a+tv, circle with center c and radius r
21    ld h2 = r*r - v.cross(c - a) * v.cross(c - a) / v.norm();
22    point p = a + v * v.dot(c - a) / v.norm();
23    if(eq(h2, 0)) return {p}; //line tangent to circle
24    else if(le(h2, 0)) return {}; //no intersection
25    else{
26      point u = v.unit() * sqrt(h2);
```

```
27      return {p - u, p + u}; //two points of intersection (chord)
28    }
29  }
30
31  vector<point> intersectSegmentCircle(const point & a, const point & b,
        const point & c, ld r){
32    //segment ab, circle with center c and radius r
33    vector<point> P = intersectLineCircle(a, b - a, c, r), ans;
34    for(const point & p : P){
35      if(pointInSegment(a, b, p)) ans.push_back(p);
36    }
37    return ans;
38  }
39
40  pair<point, ld> getCircle(const point & m, const point & n, const point & p
        ){
41    //find circle that passes through points p, q, r
42    point c = intersectLines((n + m) / 2, (n - m).perp(), (p + n) / 2, (p - n
        ).perp());
43    ld r = (c - m).length();
44    return {c, r};
45  }
46
47  vector<point> intersectionCircles(const point & c1, ld r1, const point & c2
        , ld r2){
48    //circle 1 with center c1 and radius r1
49    //circle 2 with center c2 and radius r2
50    point d = c2 - c1;
51    ld d2 = d.norm();
52    if(eq(d2, 0)) return {}; //concentric circles
53    ld pd = (d2 + r1*r1 - r2*r2) / 2;
54    ld h2 = r1*r1 - pd*pd/d2;
55    point p = c1 + d*pd/d2;
56    if(eq(h2, 0)) return {p}; //circles touch at one point
57    else if(le(h2, 0)) return {}; //circles don't intersect
58    else{
59      point u = d.perp() * sqrt(h2/d2);
60      return {p - u, p + u};
61    }
62  }
63
64  int circleInsideCircle(const point & c1, ld r1, const point & c2, ld r2){
65    //test if circle 2 is inside circle 1
66    //returns "-1" if 2 touches internally 1, "1" if 2 is inside 1, "0" if
        they overlap
67    ld l = r1 - r2 - (c1 - c2).length();
68    return (ge(l, 0) ? 1 : (eq(l, 0) ? -1 : 0));
69  }
70
71  int circleOutsideCircle(const point & c1, ld r1, const point & c2, ld r2){
72    //test if circle 2 is outside circle 1
73    //returns "-1" if they touch externally, "1" if 2 is outside 1, "0" if
        they overlap
74    ld l = (c1 - c2).length() - (r1 + r2);
75    return (ge(l, 0) ? 1 : (eq(l, 0) ? -1 : 0));
76  }
77
78  int pointInCircle(const point & c, ld r, const point & p){
79    //test if point p is inside the circle with center c and radius r
80    //returns "0" if it's outside, "-1" if it's in the perimeter, "1" if it's
        inside
81    ld l = (p - c).length() - r;
82    return (le(l, 0) ? 1 : (eq(l, 0) ? -1 : 0));
83  }
84
85  vector<vector<point>> tangents(const point & c1, ld r1, const point & c2,
        ld r2, bool inner){
86    //returns a vector of segments or a single point
87    if(inner) r2 = -r2;
88    point d = c2 - c1;
89    ld dr = r1 - r2, d2 = d.norm(), h2 = d2 - dr*dr;
90    if(eq(d2, 0) || le(h2, 0)) return {};
91    point v = d*dr/d2;
92    if(eq(h2, 0)) return {{c1 + v*r1}};
93    else{
94      point u = d.perp()*sqrt(h2)/d2;
95      return {{c1 + (v - u)*r1, c2 + (v - u)*r2}, {c1 + (v +
          u)*r1, c2 + (v +
          u)*r2}};
96    }
97  }
98
99  ld signed_angle(const point & a, const point & b){
100   return sgn(a.cross(b)) * acosl(a.dot(b) / (a.length() * b.length()));
101 }
102
103 ld intersectPolygonCircle(const vector<point> & P, const point & c, ld r){
```

```
104    //Gets the area of the intersection of the polygon with the circle
105    int n = P.size();
106    ld ans = 0;
107    for(int i = 0; i < n; ++i){
108      point p = P[i], q = P[(i+1)%n];
109      bool p_inside = (pointInCircle(c, r, p) != 0);
110      bool q_inside = (pointInCircle(c, r, q) != 0);
111      if(p_inside && q_inside){
112        ans += (p - c).cross(q - c);
113      }else if(p_inside && !q_inside){
114        point s1 = intersectSegmentCircle(p, q, c, r)[0];
115        point s2 = intersectSegmentCircle(c, q, c, r)[0];
116        ans += (p - c).cross(s1 - c) + r*r * signed_angle(s1 - c, s2 - c);
117      }else if(!p_inside && q_inside){
118        point s1 = intersectSegmentCircle(c, p, c, r)[0];
119        point s2 = intersectSegmentCircle(p, q, c, r)[0];
120        ans += (s2 - c).cross(q - c) + r*r * signed_angle(s1 - c, s2 - c);
121      }else{
122        auto info = intersectSegmentCircle(p, q, c, r);
123        if(info.size() <= 1){
124          ans += r*r * signed_angle(p - c, q - c);
125        }else{
126          point s2 = info[0], s3 = info[1];
127          point s1 = intersectSegmentCircle(c, p, c, r)[0];
128          point s4 = intersectSegmentCircle(c, q, c, r)[0];
129          ans += (s2 - c).cross(s3 - c) + r*r * (signed_angle(s1 - c, s2 - c)
                   + signed_angle(s3 - c, s4 - c));
130        }
131      }
132    }
133    return abs(ans)/2;
134  }
```

3.3   Poligonos

```
1  ld perimeter(vector<point> & P){
2    int n = P.size();
3    ld ans = 0;
4    for(int i = 0; i < n; i++){
5      ans += (P[i] - P[(i + 1) % n]).length();
6    }
7    return ans;
8  }
```

```
9
10  ld area(vector<point> & P){
11    int n = P.size();
12    ld ans = 0;
13    for(int i = 0; i < n; i++){
14      ans += P[i].cross(P[(i + 1) % n]);
15    }
16    return abs(ans / 2);
17  }
18
19  vector<point> convexHull(vector<point> P){
20    sort(P.begin(), P.end());
21    vector<point> L, U;
22    for(int i = 0; i < P.size(); i++){
23      while(L.size() >= 2 && leq((L[L.size() - 2] - P[i]).cross(L[L.size() -
             1] - P[i]), 0)){
24        L.pop_back();
25      }
26      L.push_back(P[i]);
27    }
28    for(int i = P.size() - 1; i >= 0; i--){
29      while(U.size() >= 2 && leq((U[U.size() - 2] - P[i]).cross(U[U.size() -
             1] - P[i]), 0)){
30        U.pop_back();
31      }
32      U.push_back(P[i]);
33    }
34    L.pop_back();
35    U.pop_back();
36    L.insert(L.end(), U.begin(), U.end());
37    return L;
38  }
39
40  bool pointInPerimeter(const vector<point> & P, const point & p){
41    int n = P.size();
42    for(int i = 0; i < n; i++){
43      if(pointInSegment(P[i], P[(i + 1) % n], p)){
44        return true;
45      }
46    }
47    return false;
48  }
49
```

```
50  bool crossesRay(const point & a, const point & b, const point & p){
51    return (geq(b.y, p.y) - geq(a.y, p.y)) * sgn((a - p).cross(b - p)) > 0;
52  }
53
54  int pointInPolygon(const vector<point> & P, const point & p){
55    if(pointInPerimeter(P, p)){
56      return -1; //point in the perimeter
57    }
58    int n = P.size();
59    int rays = 0;
60    for(int i = 0; i < n; i++){
61      rays += crossesRay(P[i], P[(i + 1) % n], p);
62    }
63    return rays & 1; //0: point outside, 1: point inside
64  }
65
66  //point in convex polygon in O(log n)
67  //make sure that P is convex and in ccw
68  //before the queries, do the preprocess on P:
69  // rotate(P.begin(), min_element(P.begin(), P.end()), P.end());
70  // int right = max_element(P.begin(), P.end()) - P.begin();
71  //returns 0 if p is outside, 1 if p is inside, -1 if p is in the perimeter
72  int pointInConvexPolygon(const vector<point> & P, const point & p, int
        right){
73    if(p < P[0] || P[right] < p) return 0;
74    int orientation = sgn((P[right] - P[0]).cross(p - P[0]));
75    if(orientation == 0){
76      if(p == P[0] || p == P[right]) return -1;
77      return (right == 1 || right + 1 == P.size()) ? -1 : 1;
78    }else if(orientation < 0){
79      auto r = lower_bound(P.begin() + 1, P.begin() + right, p);
80      int det = sgn((p - r[-1]).cross(r[0] - r[-1])) - 1;
81      if(det == -2) det = 1;
82      return det;
83    }else{
84      auto l = upper_bound(P.rbegin(), P.rend() - right - 1, p);
85      int det = sgn((p - l[0]).cross((l == P.rbegin() ? P[0] : l[-1]) - l[0])
            ) - 1;
86      if(det == -2) det = 1;
87      return det;
88    }
89  }
90
```

```
91  vector<point> cutPolygon(const vector<point> & P, const point & a, const
        point & v){
92    //returns the part of the convex polygon P on the left side of line a+tv
93    int n = P.size();
94    vector<point> lhs;
95    for(int i = 0; i < n; ++i){
96      if(geq(v.cross(P[i] - a), 0)){
97        lhs.push_back(P[i]);
98      }
99      if(intersectLineSegmentInfo(a, v, P[i], P[(i+1)%n]) == 1){
100       point p = intersectLines(a, v, P[i], P[(i+1)%n] - P[i]);
101       if(p != P[i] && p != P[(i+1)%n]){
102         lhs.push_back(p);
103       }
104     }
105   }
106   return lhs;
107 }
```

4 Varios

4.1 Template

```
1   #include<bits/stdc++.h>
2   using namespace std;
3
4   #define forn(i,n)      for(int i=0; i<n; i++)
5   #define forr(i,a,n)    for(int i=a; i<n; i++)
6   #define fore(i,a,n)    for(int i=a; i<=n; i++)
7   #define each(a,b)      for(auto a: b)
8   #define all(v)         v.begin(),v.end()
9   #define sz(a)          (int)a.size()
10  #define debln(a)       cout << a << "\n"
11  #define deb(a)         cout << a << " "
12  #define pb             push_back
13
14  typedef long long ll;
15  typedef vector<int> vi;
16  typedef pair<int,int> ii;
17
18  void sol(){
19
20  }
```

```
21
22  int main(){
23      ios::sync_with_stdio(false);cin.tie(0);
24
25      int t=1;
26      cin>>t;
27      while(t--){
28          sol();
29      }
30
31      return 0;
32  }
```

4.2   String a vector<int>

```
1  //Convertir una cadena de numeros separados por " " en vector de enteros
2  //Leer varias de esas querys
3  cin.ignore();
4  while(q--){
5    string s;
6    getline(cin, s);
7    vector<int> qr;
8    stringstream ss(s);
9    int num;
10   while (ss >> num)    qr.push_back(num);
11 }
```

4.3   Generar permutaciones

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