Chulalongkorn University

"" "",\0\n

Teetat T., Thawin T., Thitrin S.

2025-07-18

Contest (1)

template.hpp
`<bits/stdc++.h>`, `<ext/pb_ds/assoc_container.hpp>`, `<ext/pb_ds/tree_policy.hpp>` 1ac233,
22 lines

```cpp
using namespace std;
using namespace __gnu_pbds;

using ll = long long;
using db = long double;
using vi = vector<int>;
using vl = vector<ll>;
using vd = vector<db>;
using pii = pair<int,int>;
using pll = pair<ll,ll>;
using pdd = pair<db,db>;
const int INF=0x3fffffff;
const ll LINF=0x1fffffffffffffff;
const db DINF=numeric_limits<db>::infinity();
const db EPS=1e-9;
const db PI=acos(db(-1));

template<class T>
using ordered_set = tree<T,null_type,less<T>,rb_tree_tag,
    tree_order_statistics_node_update>;

mt19937 rng(chrono::steady_clock::now().time_since_epoch().
    count());
mt19937_64 rng64(chrono::steady_clock::now().time_since_epoch()
    .count());
```

.bashrc
3 lines

```
alias c='g++ -Wall -Wconversion -Wfatal-errors -g -std=c++17 \
  -fsanitize=undefined,address'
xmodmap -e 'clear lock' -e 'keycode 66=less greater' #caps = <>
```

.vimrc
6 lines

```
set cin aw ai is ts=4 sw=4 tm=50 nu noeb bg=dark ru cul
sy on   |   im jk <esc>   |   im kj <esc>   |   no ; :
" Select region and then type :Hash to hash your selection.
" Useful for verifying that there aren't mistypes.
ca Hash w !cpp -dD -P -fpreprocessed \| tr -d '[:space:]' \
  \| md5sum \| cut -c-6
```

hash.sh
3 lines

```
# Hashes a file, ignoring all whitespace and comments. Use for
# verifying that code was correctly typed.
cpp -dD -P -fpreprocessed | tr -d '[:space:]'| md5sum |cut -c-6
```

troubleshoot.txt
52 lines

```
Pre-submit:
Write a few simple test cases if sample is not enough.
Are time limits close? If so, generate max cases.
Is the memory usage fine?
Could anything overflow?
Make sure to submit the right file.

Wrong answer:
Print your solution! Print debug output, as well.
```

Are you clearing all data structures between test cases?
Can your algorithm handle the whole range of input?
Read the full problem statement again.
Do you handle all corner cases correctly?
Have you understood the problem correctly?
Any uninitialized variables?
Any overflows?
Confusing N and M, i and j, etc.?
Are you sure your algorithm works?
What special cases have you not thought of?
Are you sure the STL functions you use work as you think?
Add some assertions, maybe resubmit.
Create some testcases to run your algorithm on.
Go through the algorithm for a simple case.
Go through this list again.
Explain your algorithm to a teammate.
Ask the teammate to look at your code.
Go for a small walk, e.g. to the toilet.
Is your output format correct? (including whitespace)
Rewrite your solution from the start or let a teammate do it.

Runtime error:
Have you tested all corner cases locally?
Any uninitialized variables?
Are you reading or writing outside the range of any vector?
Any assertions that might fail?
Any possible division by 0? (mod 0 for example)
Any possible infinite recursion?
Invalidated pointers or iterators?
Are you using too much memory?
Debug with resubmits (e.g. remapped signals, see Various).

Time limit exceeded:
Do you have any possible infinite loops?
What is the complexity of your algorithm?
Are you copying a lot of unnecessary data? (References)
How big is the input and output? (consider scanf)
Avoid vector, map. (use arrays/unordered_map)
What do your teammates think about your algorithm?

Memory limit exceeded:
What is the max amount of memory your algorithm should need?
Are you clearing all data structures between test cases?

Data structures (2)

HashMap.h
Description: Hash map with mostly the same API as unordered_map, but
~3x faster. Uses 1.5x memory. Initial capacity must be a power of 2 (if
provided). d77092, 7 lines

```cpp
#include <bits/extc++.h>
// To use most bits rather than just the lowest ones:
struct chash { // large odd number for C
  const uint64_t C = ll(4e18 * acos(0)) | 71;
  ll operator()(ll x) const { return __builtin_bswap64(x*C); }
};
__gnu_pbds::gp_hash_table<ll,int,chash> h({},{},{},{},{1<<16});
```

LineContainer.hpp
Description: Line Container (Minimize).
Time: $\mathcal{O}(\log N)$
"src/contest/template.hpp"   24a5c4, 37 lines

```cpp
struct Line{
  static bool querymode;
    ll m,c;
  mutable ll p;
  Line(ll m,ll c):m(m),c(c),p(0){}
  Line(ll p):m(0),c(0),p(p){}
  bool operator<(const Line &o)const{
      return querymode?p<o.p:m>o.m;
  }
};

bool Line::querymode=false;

struct LineContainer:multiset<Line>{
    ll div(ll a,ll b){
      return a/b-((a^b)<0&&a%b);
  }
  bool isect(iterator x,iterator y){
      if(y==end())return x->p=LINF,false;
      if(x->m==y->m)x->p=x->c<=y->c?LINF:-LINF;
      else x->p=div(x->c-y->c,y->m-x->m);
      return x->p>=y->p;
  }
  void add(ll m,ll c){
      auto x=insert(Line(m,c)),y=next(x);
      while(isect(x,y))y=erase(y);
      if((y=x)!=begin()&&isect(--x,y))isect(x,erase(y));
      while((y=x)!=begin()&&(--x)->p>=y->p)isect(x,erase(y));
  }
  ll get(ll x){
      if(empty())return LINF;
      Line::querymode=true;
      auto l=lower_bound(Line(x));
      Line::querymode=false;
      return l->m*x+l->c;
  }
};
```

Treap.h
Description: A short self-balancing tree. It acts as a sequential container
with log-time splits/joins, and is easy to augment with additional data.
Time: $\mathcal{O}(\log N)$
1754b4, 53 lines

```cpp
struct Node {
  Node *l = 0, *r = 0;
  int val, y, c = 1;
  Node(int val) : val(val), y(rand()) {}
  void recalc();
};

int cnt(Node* n) { return n ? n->c : 0; }
void Node::recalc() { c = cnt(l) + cnt(r) + 1; }

template<class F> void each(Node* n, F f) {
  if (n) { each(n->l, f); f(n->val); each(n->r, f); }
}

pair<Node*, Node*> split(Node* n, int k) {
  if (!n) return {};
  if (cnt(n->l) >= k) { // "n->val >= k" for lower_bound(k)
    auto [L,R] = split(n->l, k);
    n->l = R;
    n->recalc();
    return {L, n};
  } else {
    auto [L,R] = split(n->r,k - cnt(n->l) - 1); // and just "k"
    n->r = L;
    n->recalc();
    return {n, R};
  }
}

Node* merge(Node* l, Node* r) {
  if (!l) return r;
```

```cpp
  if (!r) return l;
  if (l->y > r->y) {
    l->r = merge(l->r, r);
    return l->recalc(), l;
  } else {
    r->l = merge(l, r->l);
    return r->recalc(), r;
  }
}

Node* ins(Node* t, Node* n, int pos) {
  auto [l,r] = split(t, pos);
  return merge(merge(l, n), r);
}

// Example application: move the range [l, r) to index k
void move(Node*& t, int l, int r, int k) {
  Node *a, *b, *c;
  tie(a,b) = split(t, l); tie(b,c) = split(b, r - l);
  if (k <= l) t = merge(ins(a, b, k), c);
  else t = merge(a, ins(c, b, k - r));
}
```

LinkCutTree.hpp
Description: Link Cut Tree (1-indexed)

```cpp
template<int N,class T>
struct LinkCutTree{
    int ch[N][2],par[N],lz[N],rev[N];
    T val[N],sum[N],rsum[N];
    void toggle(int v){
        if(!v)return;
        swap(ch[v][0],ch[v][1]);
        swap(sum[v],rsum[v]);
        rev[v]^=1;
    }
    void push(int v){
        if(!v||!rev[v])return;
        toggle(ch[v][0]);
        toggle(ch[v][1]);
        rev[v]=0;
    }
    void pull(int v){
        if(!v)return;
        sum[v]=sum[ch[v][0]]+val[v]+sum[ch[v][1]];
        rsum[v]=rsum[ch[v][0]]+val[v]+rsum[ch[v][1]];
    }
    bool is_root(int v){
        return ch[par[v]][0]!=v&&ch[par[v]][1]!=v;
    }
    bool pos(int v){
        return ch[par[v]][1]==v;
    }
    void rotate(int v){
        int u=par[v],g=par[u];
        bool x=pos(v);
        if(!is_root(u))ch[g][pos(u)]=v;
        ch[u][x]=ch[v][!x];
        if(ch[u][x])par[ch[u][x]]=u;
        ch[v][!x]=u,par[u]=v,par[v]=g;
        pull(u),pull(v);
    }
    void splay(int v){
        if(!v)return;
        for(push(v);!is_root(v);rotate(v)){
            int u=par[v];
            if(is_root(u))push(u),push(v);
            else push(par[u]),push(u),push(v),rotate(pos(u)==
                pos(v)?u:v);
```

```cpp
        }
    }
    void access(int v){
        for(int u=v,c=0;u;u=par[u]){
            splay(u);
            ch[u][1]=c;
            pull(c=u);
        }
        splay(v);
    }
    void evert(int v){
        access(v),toggle(v);
    }
    void link(int u,int v){
        evert(u);
        access(v);
        par[u]=v;
    }
    void cut(int u,int v){
        evert(u);
        access(v);
        assert(par[u]==v);
        ch[v][0]=par[u]=0;
        pull(v);
    }
    T aggregate(int u,int v){
        evert(u);
        access(v);
        return sum[v];
    }
    void set(int u,T v){
        evert(u);
        val[u]=v;
        pull(u);
    }
};
```