National Taiwan University

TokidokiBosottoSegukideNaguru TonarinoSamusan

FHVirus, nathanlee726, Ji_Kuai

NCPC Preliminary

Sep 28, 2024

Contest (1)

.bashrc
3 lines

```
alias c='g++ -Wall -Wconversion -Wfatal-errors -g -std=c++17 \
  -fsanitize=undefined,address -DTOKI'
xmodmap -e 'clear Lock' -e 'keycode 0x42 = Escape'
```

.vimrc
5 lines

```
se nu ru cul cin ai is hls ls=2 bs=2 ts=3 sw=3 sts=3 et | sy on
" 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
```

template.cpp
3b8b37, 13 lines

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define all(x) begin(x), end(x)
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

int main() {
  cin.tie(0)->sync_with_stdio(0);
  cin.exceptions(cin.failbit);
}
```

debug.cpp
1f3f35, 11 lines

```
#ifdef TOKI
#define debug(args...) LKJ("\033[1;32m["#args"]\033[0m", args)
template<class I> void LKJ(I&&x){ cerr << x << endl; }
template<class I, class...T> void LKJ(I&&x, T&&...t)
{ cerr << x << ", "; LKJ(t...); }
template<class I> void print(I a, I b)
{ while (a != b) cerr << *a++ << ' '; cerr << endl; }
#else
#define debug(...) ((void)0)
#define print(...) ((void)0)
#endif
```

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?
```

cmp.sh
10 lines

```
# A script that checks the correctness of sol.cpp
# using testdata generated by gen.cpp and bru.cpp
# as a reference. Outputs hack if found.
source ~/.bashrc && shopt -s expand_aliases
c gen.cpp -o g && c bru.cpp -o b && c sol.cpp -o s
for i in {1..100000}; do
  echo $i && ./g>i && ./b<i>a && ./s<i>o && diff -y a o
  if [ $? == 1 ]; then echo $i; cat i; break; fi
done
echo Done.
```

Data structures (2)

OrderStatisticTree.h

Description: A set (not multiset!) with support for finding the n'th element, and finding the index of an element. To get a map, change null_type.
Time: $\mathcal{O}(\log N)$

782797, 16 lines

```
#include <bits/extc++.h>
using namespace __gnu_pbds;

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

void example() {
  Tree<int> t, t2; t.insert(8);
  auto it = t.insert(10).first;
  assert(it == t.lower_bound(9));
  assert(t.order_of_key(10) == 1);
  assert(t.order_of_key(11) == 2);
  assert(*t.find_by_order(0) == 8);
  t.join(t2); // assuming T < T2 or T > T2, merge t2 into t
}
```

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

```
#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});
```

UnionFindRollback.h

Description: Disjoint-set data structure with undo. If undo is not needed, skip (A)'s: st, time() and rollback().
Usage: int t = uf.time(); ...; uf.rollback(t);
Time: $\mathcal{O}(\log(N))$

de4ad0, 21 lines

```
struct RollbackUF {
  vi e; vector<pii> st; // (A)
  RollbackUF(int n) : e(n, -1) {}
  int size(int x) { return -e[find(x)]; }
  int find(int x) { return e[x] < 0 ? x : find(e[x]); }
  int time() { return sz(st); } // (A)
  void rollback(int t) { // (A)
    for (int i = time(); i --> t;)
      e[st[i].first] = st[i].second;
    st.resize(t);
  }
  bool join(int a, int b) {
    a = find(a), b = find(b);
    if (a == b) return false;
    if (e[a] > e[b]) swap(a, b);
    st.push_back({a, e[a]}); // (A)
    st.push_back({b, e[b]}); // (A)
    e[a] += e[b]; e[b] = a;
    return true;
  }
};
```

LineContainer.h

Description: Container where you can add lines of the form kx+m, and query maximum values at points x. Useful for dynamic programming ("convex hull trick").
Time: $\mathcal{O}(\log N)$

8ec1c7, 30 lines

```
struct Line {
  mutable ll k, m, p; // minimum: change to k > o.k;
  bool operator<(const Line& o) const { return k < o.k; }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  static const ll inf = LLONG_MAX;
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) return x->p = inf, 0;
    if (x->k == y->k) x->p = x->m > y->m ? inf : -inf; // <
    else x->p = div(y->m - x->m, x->k - y->k);
    return x->p >= y->p;
```

```
  }
  void add(ll k, ll m) {
    auto z = insert({k, m, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y));
  }
  ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.k * x + l.m;
  }
};
```

LazySegmentTree.h

Description: ZKW with ACL style nodes. [l, r), 0-base.
Usage: `SGT<Val, Tag> sgt(n);`
Time: $\mathcal{O}(N + Q \log N)$.

0f94ce, 55 lines

```
struct Val {
  int v;
  Val(int v = 0) : v(v) {} // must return identity element
  Val operator + (const Val& o) const {
    return Val(max(v, o.v)); }
  // merge two Vals, order is important
};
struct Tag {
  int t;
  Tag(int t = 0) : t(t) {} // must return identity element
  Tag operator + (const Tag& o) const { return Tag(t + o.t); }
  // compose two Tags, order is important
  Val operator() (Val v) const { return Val(v.v + t); }
  // apply the Tag to v
};

int bc(int u) { return u <= 1 ? 1 : (2 << __lg(u-1)); }
template <class V, class T> struct SGT {
  int n; vector<V> val; vector<T> tag;
  SGT(int _n): n(bc(_n)), val(n*2), tag(n*2) {}
  SGT(const vector<V>& v): n(bc(sz(v))), val(n*2), tag(n*2) {
    rep (i, 0, sz(v)) val[i+n] = v[i];
    for (int i = n; --i; ) val[i] = val[i*2] + val[i*2+1];
  }
  void upd(int u, T t)
  { val[u] = t(val[u]); if (u < n) tag[u] = tag[u] + t; }
  void pull(int u)
  { while (u /= 2) val[u] = tag[u](val[u*2] + val[u*2+1]); }
  void push(int u) {
    for (int h = __lg(n)+1, i; --h;) {
      i = u >> h;
      upd(i * 2, tag[i]);
      upd(i * 2 + 1, tag[i]);
      tag[i] = T();
    }
  }
  void set(int p, V v)
  { push(p += n); val[p] = v; pull(p); }
  V query(int l, int r) {
    V rl, rr;
    for (push(l+=n), push((r+=n)-1); l < r; l /= 2, r /= 2) {
      if (l & 1) rl = rl + val[l++];
      if (r & 1) rr = val[--r] + rr;
    }
    return rl + rr;
  }
  void modify(int l, int r, T t) {
    int tl = (l += n), tr = (r += n) - 1;
    for (push(tl), push(tr); l < r; l >>= 1, r >>= 1) {
      if (l & 1) upd(l++, t);
      if (r & 1) upd(--r, t);
    }
    pull(tl); pull(tr);
  }
};
```

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: $\mathcal{O}(\log N)$

dd38d5, 50 lines

```
struct Node {
  Node *l = 0, *r = 0;
  int val, c = 1, y; // maybe not use rand()!
  Node(int val) : val(val), y(rand()) {}
  void pull();
};
int cnt(Node* n) { return n ? n->c : 0; }
void Node::pull() { 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 pa = split(n->l, k);
    n->l = pa.second;
    n->pull();
    return {pa.first, n};
  }
  auto pa = split(n->r, k - cnt(n->l) - 1); // and just "k"
  n->r = pa.first;
  n->pull();
  return {n, pa.second};
}

Node* merge(Node* l, Node* r) {
  if (!l) return r;
  if (!r) return l;
  if (l->y > r->y) {
    l->r = merge(l->r, r);
    l->pull();
    return l;
  }
  r->l = merge(l, r->l);
  r->pull();
  return r;
}

Node* ins(Node* t, Node* n, int pos) { // 0-base
  auto pa = split(t, pos);
  return merge(merge(pa.first, n), pa.second);
}

// 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));
// }
```

RMQ.h

Description: Range Minimum Queries on an array. Returns min(V[a], V[a + 1], ... V[b - 1]) in constant time.
Usage: `RMQ rmq(values); rmq.query(inclusive, exclusive);`
Time: $\mathcal{O}(|V| \log |V| + Q)$

05c3c0, 16 lines

```
template <class T>
struct RMQ {
  vector<vector<T>> a;
  RMQ(const vector<T>& v) : a(1, v) {
    for (int p = 1, k = 1; p * 2 <= sz(v); p *= 2, ++k) {
      a.emplace_back(sz(v) - p * 2 + 1);
      rep(j, 0, sz(a[k]))
        a[k][j] = min(a[k - 1][j], a[k - 1][j + p]);
    }
  }
  T query(int l, int r) {
    assert(l < r);
    int d = 31 - __builtin_clz(r - l);
    return min(a[d][l], a[d][r - (1 << d)]);
  }
};
```

MoQueries.h

Description: Answer interval or tree path queries by finding an approximate TSP through the queries, and moving from one query to the next by adding/removing points at the ends. If values are on tree edges, change `step` to add/remove the edge (a, c) and remove the initial `add` call (but keep `in`).
Time: $\mathcal{O}\left(N\sqrt{Q}\right)$

a12ef4, 49 lines

```
void add(int ind, int end) { ... } // add a[ind] (end = 0 or 1)
void del(int ind, int end) { ... } // remove a[ind]
int calc() { ... } // compute current answer

vi mo(vector<pii> Q) {
  int L = 0, R = 0, blk = 350; // ~N/sqrt(Q)
  vi s(sz(Q)), res = s;
#define K(x) pii(x.first/blk, x.second ^ -(x.first/blk & 1))
  iota(all(s), 0);
  sort(all(s), [&](int s, int t){ return K(Q[s]) < K(Q[t]); });
  for (int qi : s) {
    pii q = Q[qi];
    while (L > q.first) add(--L, 0);
    while (R < q.second) add(R++, 1);
    while (L < q.first) del(L++, 0);
    while (R > q.second) del(--R, 1);
    res[qi] = calc();
  }
  return res;
}

vi moTree(vector<array<int, 2>> Q, vector<vi>& ed, int root=0){
  int N = sz(ed), pos[2] = {}, blk = 350; // ~N/sqrt(Q)
  vi s(sz(Q)), res = s, I(N), L(N), R(N), in(N), par(N);
  add(0, 0), in[0] = 1;
  auto dfs = [&](int x, int p, int dep, auto& f) -> void {
    par[x] = p;
    L[x] = N;
    if (dep) I[x] = N++;
    for (int y : ed[x]) if (y != p) f(y, x, !dep, f);
    if (!dep) I[x] = N++;
    R[x] = N;
  };
  dfs(root, -1, 0, dfs);
#define K(x) pii(I[x[0]] / blk, I[x[1]] ^ -(I[x[0]] / blk & 1))
  iota(all(s), 0);
  sort(all(s), [&](int s, int t){ return K(Q[s]) < K(Q[t]); });
  for (int qi : s) rep(end,0,2) {
    int &a = pos[end], b = Q[qi][end], i = 0;
#define step(c) { if (in[c]) { del(a, end); in[a] = 0; } \
                  else { add(c, end); in[c] = 1; } a = c; }
    while (!(L[b] <= L[a] && R[a] <= R[b]))
      I[i++] = b, b = par[b];
    while (a != b) step(par[a]);
    while (i--) step(I[i]);
```

```
    if (end) res[qi] = calc();
  }
  return res;
}
```