Justyna To Chlep

Na Przypale Albo Wcale

Tomasz Nowak, Michał Staniewski, Justyna Jaworska

2019-06-25

Data structures (1)

find-union

Description: mniejszy do wiekszego
Time: $\mathcal{O}(\alpha(n))$

3a2363, 19 lines

```
struct FindUnion {
  vector<int> rep;
  bool sameSet(int a, int b) { return find(a) == find(b); }
  int size(int x) { return -rep[find(x)]; }
  int find(int x) {
    return rep[x] < 0 ? x : rep[x] = find(rep[x]);
  }
  bool join(int a, int b) {
    a = find(a), b = find(b);
    if(a == b)
      return false;
    if(-rep[a] < -rep[b])
      swap(a, b);
    rep[a] += rep[b];
    rep[b] = a;
    return true;
  }
  FindUnion(int n) : rep(n, -1) {}
};
```

lazy-segment-tree

Description: Michal popisz sie opisem
Usage: add(l, r, val) dodaje na przedziale
quert(l, r) bierze maxa z przedzialu

414b27, 57 lines

```
struct Node {
  int val, lazy;
  int size = 1;
};

struct Tree {
  vector<Node> nodes;
  int size = 1;

  Tree(int n) {
    while(size < n) size *= 2;
    nodes.resize(size * 2);
    for(int i = size - 1; i >= 1; i--)
      nodes[i].size = nodes[i * 2].size * 2;
  }

  void add_val(int v, int val) {
    nodes[v].val += val;
    nodes[v].lazy += val;
  }

  void propagate(int v) {
    REP(i, 2)
      add_val(v * 2 + i, nodes[v].lazy);
    nodes[v].lazy = 0;
  }

  int query(int l, int r, int v = 1) {
    if(l == 0 && r == nodes[v].size - 1)
      return nodes[v].val;
    propagate(v);
    int m = nodes[v].size / 2;
    if(r < m)
      return query(l, r, v * 2);
    else if(m <= l)
      return query(l - m, r - m, v * 2 + 1);
    else
      return max(query(l, m - 1, v * 2), query(0, r - m, v * 2
          + 1));
  }

  void add(int l, int r, int val, int v = 1) {
    if(l == 0 && r == nodes[v].size - 1) {
      add_val(v, val);
      return;
    }
    propagate(v);
    int m = nodes[v].size / 2;
    if(r < m)
      add(l, r, val, v * 2);
    else if(m <= l)
      add(l - m, r - m, val, v * 2 + 1);
    else
      add(l, m - 1, val, v * 2), add(0, r - m, val, v * 2 + 1);

    nodes[v].val = max(nodes[v * 2].val, nodes[v * 2 + 1].val);
  }
};
```

segment-tree

Description: Michal popisz sie opisem
Usage: todo

541b9e, 29 lines

```
struct Tree {
  using T = int;
  T f(T a, T b) { return a + b; }
  vector<T> nodes;
  int size = 1;

  Tree(int n, T val = 0) {
    while(size < n) size *= 2;
    nodes.resize(size * 2, val);
  }

  void update(int pos, T val) {
    nodes[pos += size] = val;
    while(pos /= 2)
      nodes[pos] = f(nodes[pos * 2], nodes[pos * 2 + 1]);
  }

  T query(int l, int r) {
    l += size, r += size;
    T ret = (l != r ? f(nodes[l], nodes[r]) : nodes[l]);
    while(l + 1 < r) {
      if(l % 2 == 0)
        ret = f(ret, nodes[l + 1]);
      if(r % 2 == 1)
        ret = f(ret, nodes[r - 1]);
      l /= 2, r /= 2;
    }
  }
};
```

fenwick-tree

Description: indexowanie od 0
Usage: update(pos, val) dodaje val do elementu pos
query(pos) zwraca sumę pierwszych pos elementów
lower_bound(val) zwraca pos, że suma [0, pos] <= val

78e5fe, 26 lines

```
struct Fenwick {
  vector<LL> s;
  Fenwick(int n) : s(n) {}

  void update(int pos, LL val) {
    for(; pos < size(s); pos |= pos + 1)
      s[pos] += val;
  }

  LL query(int pos) {
    LL ret = 0;
    for(; pos > 0; pos &= pos - 1)
      ret += s[pos - 1];
    return ret;
  }

  int lower_bound(LL val) {
    if(val <= 0) return -1;
    int pos = 0;
    for(int pw = 1 << 25; pw; pw /= 2) {
      if(pos + pw <= size(s) && s[pos + pw - 1] < sum)
        pos += pw, sum -= s[pos - 1];
    }
    return pos;
  }
};
```

ordered-set

Description: lepszy set. Jeśli chcemy multiseta, to używamy par val, id. Nie dziala z -D_GLIBCXX_DEBUG
Usage: insert(x) dodaje element x
find_by_order(i) zwraca iterator do i-tego elementu
order_of_key(x) zwraca, ile jest mniejszych elementów,
x nie musi być w secie

<ext/pb_ds/assoc_container.hpp>, <ext/pb_ds/tree_policy.hpp> 0a779f, 9 lines

```
using namespace __gnu_pbds;

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

lichao-tree

Description: Dla funkcji, których pary przecinaja sie co najwyżej raz, oblicza maximum w punkcie x. Podany kod jest dla funkcji liniowych.

a7f64a, 50 lines

```
struct Function {
  int a, b;
  L operator()(int x) {
    return x * L(a) + b;
  }
  Function(int p = 0, int q = inf) : a(p), b(q) {}
};
ostream& operator<<(ostream &os, Function f) {
  return os << make_pair(f.a, f.b);
}

struct LiChaoTree {
  int size = 1;
  vector<Function> tree;

  LiChaoTree(int n) {
    while(size < n)
      size *= 2;
    tree.resize(size << 1);
  }

  L get_min(int x) {
    int v = x + size;
    L ans = inf;
    while(v) {
      ans = min(ans, tree[v](x));
      v >>= 1;
    }
    return ans;
  }

  void add_func(Function new_func, int v, int l, int r) {
    int m = (l + r) / 2;
    bool domin_l = tree[v](l) > new_func(l),
       domin_m = tree[v](m) > new_func(m);
    if(domin_m)
      swap(tree[v], new_func);

    if(l == r)
      return;
    else if(domin_l == domin_m)
      add_func(new_func, v << 1 | 1, m + 1, r);
    else
      add_func(new_func, v << 1, l, m);
  }

  void add_func(Function new_func) {
    add_func(new_func, 1, 0, size - 1);
  }
};
```

Geometry (2)

Graphs (3)

Math (4)

extended-gcd

Description: Dla danego (a, b) znajduje takie $(gcd(a, b), x, y)$, że $ax + by = gcd(a, b)$

Time: $\mathcal{O}(\log(\max(a, b)))$

Usage: `LL gcd, x, y; tie(gcd, x, y) = extendedGcd(a, b);`

402db5, 7 lines

```
tuple<LL, LL, LL> extendedGcd(LL a, LL b) {
  if(a == 0)
    return {b, 0, 1};
  LL x, y, nwd;
  tie(nwd, x, y) = extendedGcd(b % a, a);
  return {nwd, y - x * (b / a), x};
}
```

Optimizations (5)

Random stuff (6)

Strings (7)

Utils (8)

headers

Description: Naglówki używane w każdym kodzie. Dziala na każdy kontener i pary

Usage: `debug(a, b, c) << d << e;` wypisze `a, b, c: a; b; c; de`

<bits/stdc++.h> cd38cb, 34 lines

```
using namespace std;
using LL = long long;
#define FOR(i, l, r) for(int i = (l); i <= (r); ++i)
#define REP(i, n) FOR(i, 0, (n) - 1)
template<class T> int size(T &&x) {
  return int(x.size());
}
template<class A, class B> ostream& operator<<(ostream &out,
    const pair<A, B> &p) {
  return out << ’(’ << p.first << ", " << p.second << ’)’;
}
template<class T> auto operator<<(ostream &out, T &&x) ->
    decltype(x.begin(), out) {
  out << ’{’;
  for(auto it = x.begin(); it != x.end(); ++it)
    out << *it << (it == prev(x.end()) ? "" : ", ");
  return out << ’}’;
}
void dump() {}
template<class T, class... Args> void dump(T &&x, Args... args)
     {
  cerr << x << ";  ";
  dump(args...);
}
#ifdef DEBUG
  const int seed = 1;
  struct Nl{~Nl(){cerr << ’\n’;}};
# define debug(x...) cerr << (#x != "" ? #x ":  " : ""), dump(x
    ), Nl(), cerr
#else
  const int seed = chrono::system_clock::now().time_since_epoch
      ().count();
# define debug(...) 0 && cerr
#endif
mt19937_64 rng(seed);
int rd(int l, int r) {
  return uniform_int_distribution<int>(l, r)(rng);
}
// end of templates
```

example-code

Description: jakiś tam opis, można walnąć latexa: $2 + 2 = 5$. ęóąślżźćńĘÓĄŚLŻŹĆŃ

Time: $\mathcal{O}\left(n\sqrt(n)\log^2(n)\right)$, gdzie n to jakaś fajna zmienna

Memory: $\mathcal{O}(n \log n)$

Usage: `int rd = getRandomValue(0, 5);`
`int rd01 = ExampleStruct().get();`
`ęóąślżźćńĘÓĄŚLŻŹĆŃ`

bbd845, 24 lines

```
mt19937_64 rng(chrono::system_clock::now().time_since_epoch().
    count());
int getRandomValue(int l, int r) {
  return uniform_int_distribution<int>(l, r)(rng);
}

struct ExampleStruct {
  int random_variable;
  constexpr int left = 0, right = 1;

  ExampleStruct() {
    random_variable = getRandomValue(left, right);
    if(random_variable == 0) {
      // some random bulls**t to show the style
      ++random_variable;
    }
    else
      --random_variable;
  }

  int& get_value() {
    return random_variable;
  }
};
```