KTH Royal Institute of Technology

Omogen Heap

Simon Lindholm, Johan Sannemo, Mårten Wiman

2020-09-22

1 Contest 1

2 Data structures 1

3 Geometry 1

4 Graphs 1

5 Mathematics 2

6 Miscellaneous 2

7 Strings 2

Contest (1)

template.cpp

9 lines

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

using ll = long long;

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

Data structures (2)

BIT.h

Description: Query [l, r] sums, and point updates. kth() returns the smallest index i s.t. query(0, i) >= k
Time: $\mathcal{O}(\log n)$ for all ops.

33f78c, 22 lines

```
template <typename T>
struct BIT {
  vector<T> s;
  int n;
  BIT(int n) : s(n + 1), n(n) {}
  void update(int i, T v) {
    for (i++; i <= n; i += i & -i) s[i] += v;
  }
  T query(int i) {
    T ans = 0;
    for (i++; i; i -= i & -i) ans += s[i];
    return ans;
  }
  T query(int l, int r) { return query(r) - query(l - 1); }
  int kth(T k) { // returns n if k > sum of tree
    if (k <= 0) return -1;
    int i = 0;
    for (int pw = 1 << __lg(n); pw; pw >>= 1)
      if (i + pw <= n && s[i + pw] < k) k -= s[i += pw];
    return i;
  }
};
```

KDBIT.h

Description: k-dimensional BIT. `BIT<int, N, M>` gives an $N \times M$ BIT. Query: `bit.query(x1, x2, y1, y2)` Update: `bit.update(x, y, delta)`
Time: $O(\log^k n)$ Status: Tested

3b9692, 28 lines

```
template <class T, int... Ns>
struct BIT {
  T val = 0;
  void update(T v) { val += v; }
  T query() { return val; }
};
template <class T, int N, int... Ns>
struct BIT<T, N, Ns...> {
  BIT<T, Ns...> bit[N + 1];
  // map<int, BIT<T, Ns...>> bit;
  // if the memory use is too high
  template <class... Args>
  void update(int i, Args... args) {
    for (i++; i <= N; i += i & -i) bit[i].update(args...);
  }
  template <class... Args>
  T query(int i, Args... args) {
    T ans = 0;
    for (i++; i; i -= i & -i) ans += bit[i].query(args...);
    return ans;
  }
  template <class... Args,
            enable_if_t<(sizeof...(Args) ==
                         2 * sizeof...(Ns))>* = nullptr>
  T query(int l, int r, Args... args) {
    return query(r, args...) - query(l - 1, args...);
  }
};
```

DSU.h

Description: Maintains union of disjoint sets
Time: $\mathcal{O}(\alpha(N))$

c22586, 14 lines

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

RMQ.h

Description: Constant time subarray min/max queries for a fixed array
Time: $O(nlogn)$ initialization and $O(1)$ queries. Status: Tested

536eac, 15 lines

```
template <typename T, class Compare = less<T>>
struct RMQ {
  vector<vector<T>> t;
  Compare cmp;
  RMQ(vector<T>& a) : t(__lg(a.size()) + 1, a) {
    int n = a.size(), lg = __lg(n);
    for (int k = 1, len = 1; k <= lg; k++, len <<= 1)
      for (int i = 0; i + 2 * len - 1 < n; i++)
        t[k][i] = min(t[k - 1][i], t[k - 1][i + len], cmp);
  }
  T query(int a, int b) {
    int k = __lg(b - a + 1), len = 1 << k;
    return min(t[k][a], t[k][b - len + 1], cmp);
  }
};
```

Geometry (3)

Graphs (4)

SCCTarjan.h

Description: Finds strongly connected components of a directed graph. Visits/indexes SCCs in reverse topological order.
Usage: `scc(graph)` returns an array that has the ID of each node's SCC. `scc(graph, [&](vector<int>& v) { ... })` calls the lambda on each SCC, and returns the same array.
Time: $\mathcal{O}(|V| + |E|)$

358d18, 37 lines

```
namespace SCCTarjan {
  vector<int> val, comp, z, cont;
  int Time, ncomps;
  template <class G, class F>
  int dfs(int j, G& g, F& f) {
    int low = val[j] = ++Time, x;
    z.push_back(j);
    for (auto e : g[j])
      if (comp[e] < 0) low = min(low, val[e] ?: dfs(e, g, f));
    if (low == val[j]) {
      do {
        x = z.back();
        z.pop_back();
        comp[x] = ncomps;
        cont.push_back(x);
      } while (x != j);
      f(cont);
      cont.clear();
      ncomps++;
    }
    return val[j] = low;
  }
  template <class G, class F>
  vector<int> scc(G& g, F f) {
    int n = g.size();
    val.assign(n, 0);
    comp.assign(n, -1);
    Time = ncomps = 0;
    for (int i = 0; i < n; i++)
      if (comp[i] < 0) dfs(i, g, f);
    return comp;
  }
  template <class G> // convenience function w/o lambda
  vector<int> scc(G& g) {
    return scc(g, [](auto& v) {});
  }
} // namespace SCCTarjan
```

SCCKosaraju.h

Description: Finds strongly connected components of a directed graph. Visits/indexes SCCs in topological order.
Usage: `scc(graph)` returns an array that has the ID of each node's SCC.
Time: $\mathcal{O}(|V| + |E|)$

9b78e7, 29 lines

```
namespace SCCKosaraju {
  vector<vector<int>> adj, radj;
  vector<int> todo, comp;
  vector<bool> vis;
  void dfs1(int x) {
    vis[x] = 1;
    for (int y : adj[x])
      if (!vis[y]) dfs1(y);
    todo.push_back(x);
  }
  void dfs2(int x, int i) {
```

```
    comp[x] = i;
    for (int y : radj[x])
      if (comp[y] == -1) dfs2(y, i);
  }
  vector<int> scc(vector<vector<int>>& _adj) {
    adj = _adj;
    int time = 0, n = adj.size();
    comp.resize(n, -1), radj.resize(n), vis.resize(n);
    for (int x = 0; x < n; x++)
      for (int y : adj[x]) radj[y].push_back(x);
    for (int x = 0; x < n; x++)
      if (!vis[x]) dfs1(x);
    reverse(todo.begin(), todo.end());
    for (int x : todo)
      if (comp[x] == -1) dfs2(x, time++);
    return comp;
  }
}; // namespace SCCKosaraju
```

Mathematics (5)

Miscellaneous (6)

NDimensionalVector.h

3c0f61, 12 lines

```
template <int D, typename T>
struct Vec : public vector<Vec<D - 1, T>> {
  static_assert(D >= 1,
                "Vector dimension must be greater than zero!");
  template <typename... Args>
  Vec(int n = 0, Args... args)
    : vector<Vec<D - 1, T>>(n, Vec<D - 1, T>(args...)) {}
};
template <typename T>
struct Vec<1, T> : public vector<T> {
  Vec(int n = 0, const T& val = T()) : vector<T>(n, val) {}
};
```

Submasks.h

35424b, 3 lines

```
for (int mask = 0; mask < (1 << n); mask++)
  for (int sub = mask; sub; sub = (sub - 1) & mask)
// do thing
```

Strings (7)