

## KTH Royal Institute of Technology

# Omogen Heap

Simon Lindholm, Johan Sannemo, Mårten Wiman

1	Contest	
2	Data structures	
3	Geometry	
4	Graphs	
5	Mathematics	
6	Miscellaneous	
7	Strings	
<u>C</u>	$\frac{\text{Contest}}{\text{Contest}}$ (1)	
template.cpp		9 1i
	nclude <bits stdc++.h=""> ing namespace std;</bits>	
us	ing 11 = long long;	
	<pre>t main() { cin.tie(0)-&gt;sync_with_stdio(0); cin.exceptions(cin.failbit);</pre>	

## 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 1, int r) { return query(r) - query(1 - 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];</pre>
    return i:
};
```

#### KDBIT.h

```
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...);</pre>
  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 1, int r, Args... args) {
    return query(r, args...) - query(l - 1, args...);
};
```

#### DSU.ł

1

1

1

1

 $\textbf{Description:} \ \ \text{Maintains union of disjoint sets}$ 

Time:  $\mathcal{O}\left(\alpha(N)\right)$ 

```
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 36cac. 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);
};
</pre>
```

## Geometry (3)

## Graphs (4)

## Mathematics (5)

### Miscellaneous (6)

35424b, 3 lines

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

## Strings (7)