ICPCライブラリ（んぐ）

union-find

```
class UnionFind {
public:
   UnionFind() : _n(0) {}
   UnionFind(int n) : _n(n), parent_or_size(n, -1) {}

   int unite(int a, int b) {
      assert(0 <= a && a < _n);
      assert(0 <= b && b < _n);
      int x = leader(a), y = leader(b);
      if (x == y) return x;
      if (-parent_or_size[x] < -parent_or_size[y]) std::swap(x, y);
      parent_or_size[x] += parent_or_size[y];
      parent_or_size[y] = x;
      return x;
   }

   bool equiv(int a, int b) {
      assert(0 <= a && a < _n);
      assert(0 <= b && b < _n);
      return leader(a) == leader(b);
   }

   int leader(int a) {
      assert(0 <= a && a < _n);
      if (parent_or_size[a] < 0) return a;
      return parent_or_size[a] = leader(parent_or_size[a]);
   }

   int size(int a) {
      assert(0 <= a && a < _n);
      return -parent_or_size[leader(a)];
   }

   vvi groups() {
      vi leader_buf(_n), group_size(_n);
      rep (i, _n) {
         leader_buf[i] = leader(i);
         group_size[leader_buf[i]]++;
      }
      vvi result(_n);
      rep (i, _n) result[i].reserve(group_size[i]);
      rep (i, _n) result[leader_buf[i]].pb(i);
      result.erase(
            remove_if(all(result), [&](const vi& v) { return v.empty();
}), result.end());
      return result;
```

```
    }

  private:
    int _n;

    // root node: -1 * component size
    // otherwise: parent
    vi parent_or_size;
};
```

static-range-sum

```
template <class T>
struct StaticRangeSum {
   StaticRangeSum() = default;
   explicit StaticRangeSum(const vec<T>& seq) {
      const int n = len(seq);
      sums.resize(n + 1);
      sums[0] = 0;
      partial_sum(all(seq), begin(sums) + 1);
   }

   T get(int r) const {
      return get(0, r);
   }
   T operator ()(int r) const { return get(0, r); }

   T get(int l, int r) const {
      assert(0 <= l and l < r and r <= len(sums) - 1);
      return sums[r] - sums[l];
   }
   T operator ()(int l, int r) const { return get(l, r); }

   int lower_bound(T val) const {
      return distance(cbegin(sums) + 1, lower_bound(cbegin(sums) + 1,
sums.cend(), val));
   }

   int upper_bound(T val) const {
      return distance(cbegin(sums) + 1, upper_bound(cbegin(sums) + 1,
sums.cend(), val));
   }

   vec<T> sums;
};
```

prime

```
struct Prime {
    Prime() : n_max(0) {}
    // O ( N loglog(N) )
    Prime(int n) : n_max(n), table(n+1, true), osak(n+1) {
        iota(osak.begin(), osak.end(), 0);
        osak[0] = 1;
        if (n >= 0) table[0] = false;
        if (n >= 1) table[1] = false;
        for (int i = 2; i * i <= n; i++) {
            if (not table[i]) continue;
            for(int k = i + i; k <= n; k += i) {
                table[k] = false;
                osak[k] = i;
            }
        }
        reps (i, 2, n) if (table[i]) lst.pb(i);
    }

    // n <= n_max のとき: O(1)
    // それ超えのとき: O( sqrt(N) )
    bool is(const int n) {
        if (n <= n_max) return table[n];
        if (n <= 4) return n == 2 || n == 3;
        if (n % 2 == 0 || n % 3 == 0 || (n % 6 != 1 && n % 6 != 5)) return
false;
        for (int i = 5; i * i <= n; i += 6) if (n % i == 0 || n % (i + 2)
== 0) return false;
        return true;
    }

    // O( sqrt(N) )
    map<int, int> factor(int n) {
        if (n == 1) {
            map<int, int> one;
            one[1] = 1;
            return one;
        }
        if (n <= n_max) return impl_factor_fast(n);
        map<int, int> ret;
        for (int i = 2; i * i <= n; i++) {
            while (n % i == 0) {
                ret[i]++;
                n /= i;
            }
        }
        if (n != 1) ret[n] = 1;
        return ret;
    }

    // O( log(N) )
```

```
    map<int, int> impl_factor_fast(int n) {
        map<int, int> ret;
        while (n != 1) {
            int p = osak[n];
            ret[p]++;
            n /= p;
        }
        return ret;
    }

    // O( len(v) log(v_max) )
    bool to(vi v) {
        unordered_set<int> s;
        for (auto& n : v) {
            while (n != 1) {
                int p = osak[n];
                if (s.count(p)) return false;
                else s.insert(p);
                while (n % p == 0) n /= p;
            }
        }
        return true;
    }

    const int n_max;
    vb table;
    vi osak;
    vi lst;
} PRIME(1e7);
```