ICPC Team Reference Material

Contents

1 Setup

1.1 Vimrc

```
1  let mapleader = " "
2  syntax on
3  filetype plugin on
4  set nocompatible
5  set autoread
6  set foldmethod=marker
7  set autoindent
8  set clipboard+=unnamedplus
9  set number relativenumber
10 colorscheme desert
11 set cursorline
12 set shiftwidth=2 softtabstop=2 expandtab
13 map cr :w! && !compile %:p:r<CR>
14 map cx ggVGy
15 vmap < <gv
16 vmap > >gv
17 autocmd TextChanged,TextChangedI * write
18 set undofile
19 set undodir=~/.vim/undo
20 set undolevels=1000
21 set undoreload=10000
```

1.2 Compilation

```
1  #!/bin/bash
2  # put this file in .local/bin or add its dir to the PATH variable
3  compile() {
4      g++ -Wall -Wextra -Wshadow -Ofast -std=c++17 -pedantic -Wformat=2 -Wconversion -Wlogical-op -Wshift-
           overflow=2 -Wduplicated-cond -Wfloat-equal -fno-sanitize-recover -fstack-protector -fsanitize=
           address,undefined -fmax-errors=2 -o "$1"{,.cpp}
5  }
6  compile "$1"
```

2 Data Structure

2.1 segmented tree

```
1  class SegmentTree {
2  public:
3    SegmentTree(int n) {
4      size = 1;
5      while (size < n) size <<= 1;
6      sums.assign(2 * size, 0LL);
7      mins.assign(2 * size, LLONG_MAX);
8      maxs.assign(2 * size, LLONG_MIN);
9    }
10
11   void build(const vi &a) {
12     build(a, 0, 0, size);
13   }
14
15   void update(int i, int v) {
16     update(i, v, 0, 0, size);
17   }
18
19   ll sumSeg(int l, int r) {
20     return sumSeg(l, r, 0, 0, size);
21   }
22
23   ll minSeg(int l, int r) {
```

```
24     return minSeg(l, r, 0, 0, size);
25   }
26
27   ll maxSeg(int l, int r) {
28     return maxSeg(l, r, 0, 0, size);
29   }
30
31 private:
32   int size;
33   vector<ll> sums, mins, maxs;
34
35   void build(const vi &a, int x, int lx, int rx) {
36     if (rx - lx == 1) {
37       if (lx < (int)a.size()) {
38         sums[x] = a[lx];
39         mins[x] = a[lx];
40         maxs[x] = a[lx];
41       }
42       return;
43     }
44     int mid = (lx + rx) / 2;
45     build(a, 2 * x + 1, lx, mid);
46     build(a, 2 * x + 2, mid, rx);
47     sums[x] = sums[2 * x + 1] + sums[2 * x + 2];
48     mins[x] = min(mins[2 * x + 1], mins[2 * x + 2]);
49     maxs[x] = max(maxs[2 * x + 1], maxs[2 * x + 2]);
50   }
51
52   void update(int i, int v, int x, int lx, int rx) {
53     if (rx - lx == 1) {
54       sums[x] = v;
55       mins[x] = v;
56       maxs[x] = v;
57       return;
58     }
59     int mid = (lx + rx) / 2;
60     if (i < mid) {
61       update(i, v, 2 * x + 1, lx, mid);
62     } else {
63       update(i, v, 2 * x + 2, mid, rx);
64     }
65     sums[x] = sums[2 * x + 1] + sums[2 * x + 2];
66     mins[x] = min(mins[2 * x + 1], mins[2 * x + 2]);
67     maxs[x] = max(maxs[2 * x + 1], maxs[2 * x + 2]);
68   }
69
70   ll sumSeg(int l, int r, int x, int lx, int rx) {
71     if (lx >= r || l >= rx) return 0;
72     if (lx >= l && rx <= r) return sums[x];
73     int mid = (lx + rx) / 2;
74     ll left = sumSeg(l, r, 2 * x + 1, lx, mid);
75     ll right = sumSeg(l, r, 2 * x + 2, mid, rx);
76     return left + right;
77   }
78
79   ll minSeg(int l, int r, int x, int lx, int rx) {
80     if (lx >= r || l >= rx) return LLONG_MAX;
81     if (lx >= l && rx <= r) return mins[x];
82     int mid = (lx + rx) / 2;
83     ll left = minSeg(l, r, 2 * x + 1, lx, mid);
84     ll right = minSeg(l, r, 2 * x + 2, mid, rx);
85     return min(left, right);
86   }
87
88   ll maxSeg(int l, int r, int x, int lx, int rx) {
89     if (lx >= r || l >= rx) return LLONG_MIN;
90     if (lx >= l && rx <= r) return maxs[x];
91     int mid = (lx + rx) / 2;
92     ll left = maxSeg(l, r, 2 * x + 1, lx, mid);
93     ll right = maxSeg(l, r, 2 * x + 2, mid, rx);
94     return max(left, right);
95   }
96 };
```

3 Graph algorithms

4 Mathematics

4.1 ncr

```
1  #include <bits/stdc++.h>
2  #define int long long
3  using namespace std;
```

```cpp
4
5   template<class T>
6   using rpq = priority_queue<T, vector<T>, greater<T>>;
7
8   template<int32_t mod>
9   struct mint {
10      using Z = mint;
11      int32_t x;
12      mint(int32_t x = 0) : x(norm(x)) {}
13      mint(long long x) : x(norm(x % mod)) {}
14      inline int32_t norm(int32_t x) const {
15          return x >= mod ? x - mod : (x < 0 ? x + mod : x);
16      }
17      Z power(long long b) const {
18          Z res = 1, a = x;
19          for (; b; b >>= 1, a *= a)
20              if (b & 1) res *= a;
21          return res;
22      }
23      Z inv() const { return assert(x != 0), power(mod - 2); }
24      Z operator-() const { return -x; }
25      Z &operator*=(const Z &r) { return *this = (long long) x * r.x; }
26      Z &operator+=(const Z &r) { return *this = x + r.x; }
27      Z &operator-=(const Z &r) { return *this = x - r.x; }
28      Z &operator/=(const Z &r) { return *this *= r.inv(); }
29      friend Z operator*(const Z &l, const Z &r) { return Z(l) *= r; }
30      friend Z operator+(const Z &l, const Z &r) { return Z(l) += r; }
31      friend Z operator-(const Z &l, const Z &r) { return Z(l) -= r; }
32      friend Z operator/(const Z &l, const Z &r) { return Z(l) /= r; }
33      friend ostream &operator<<(ostream &os, const Z &a) { return os << a.x; }
34      friend istream &operator>>(istream &is, Z &a) {
35          long long y = 0;
36          return is >> y, a = y, is;
37      }
38  };
39
40  // constexpr int MOD = 998244353;
41  constexpr int MOD = 1000000007;
42  using Z = mint<MOD>;
43
44  vector<Z> fact = {1};
45  vector<Z> fact_inv = {1};
46
47  void build_fact(int n = 1e6) {
48      while ((int) fact.size() < n + 1)
49          fact.push_back(fact.back() * (int) fact.size());
50      fact_inv.resize(fact.size());
51      fact_inv.back() = fact.back().inv();
52      for (int j = fact_inv.size() - 2; fact_inv[j].x == 0; j--)
53          fact_inv[j] = fact_inv[j + 1] * (j + 1);
54  }
55
56  Z ncr(int n, int r) {
57      if (r > n || r < 0) return 0;
58      if ((int) fact.size() < n + 1) build_fact(n);
59      return fact[n] * fact_inv[r] * fact_inv[n - r];
60  }
61
62  Z npr(int n, int r) {
63      if (r > n || r < 0) return 0;
64      if ((int)fact.size() < n + 1) build_fact(n);
65      return fact[n] * fact_inv[n - r];
66  }
```

4.2   fastpower

```cpp
1   int fast_power(int a, int b) {
2       int res = 1;
3       while (b) {
4           if (b & 1) res = res * a % mod;
5           a = a * a % mod;
6           b >>= 1;
7       }
8       return res;
9   }
```

4.3   simple-sieve

```cpp
1   const int NMAX = 1000000;
2   bitset<NMAX / 2> bits;
3
4   void precalcseive() {
```

```cpp
5       bits.set();
6       for (int i = 3; i / 2 < bits.size(); i = 2 * bits._Find_next(i / 2) + 1) {
7           for (auto j = (int64_t)i * i / 2; j < bits.size(); j += i)
8               bits[j] = 0;
9       }
10  }
11
12  //count all the divisors of a number
13  int divCount(int n) {
14      int total = 1;
15      int count = 0;
16      int p = 2;
17      if (n % p == 0) {
18          while (n % p == 0) {
19              n = n / p;
20              count++;
21          }
22          total = total * (count + 1);
23      }
24      for (p = 3; p <= n; p += 2) {
25          if (bits[p / 2]) {
26              count = 0;
27              if (n % p == 0) {
28                  while (n % p == 0) {
29                      n = n / p;
30                      count++;
31                  }
32                  total = total * (count + 1);
33              }
34          }
35      }
36      return total;
37  }
```

4.4   calculate all divisors

```cpp
1   void divs() {
2       const int n = 1e6;
3       vector<vector<int>> divs(n);
4
5       for (int i = 1; i < n; i++) {
6           for (int j = i; j < n; j += i) {
7               divs[j].push_back(i);
8           }
9       }
10
11      // all divisors of 12
12      for (auto fact : divs[12])
13          cout << fact << endl;
14  }
```

4.5   calculate all prime factors

```cpp
1   {
2       map<int, int> primes;
3       for (int i = 2; i * i <= n; i++) {
4           while (n % i == 0) {
5               primes[i]++; // i is a prime
6               n /= i;
7           }
8       }
9
10      if (n != 1) {
11          primes[n]++;
12      }
13
14      for (auto [a, b] : primes)
15          cout << a << endl;
16  }
```

5 Geometry

6 Miscellaneous

6.1 C++ template

```
1  #include <bits/stdc++.h>
2
3  #define endl '\n'
4  #define int long long
5  #define ld  long double
6  #define all(a) (a).begin(), (a).end()
7  #define sz(a) (int)(a).size()
8  #define pb push_back
9  #define F first
10 #define S second
11 #define vi vector<int>
12
13 using namespace std;
14
15 void file() {
16   freopen("input.in", "r", stdin);
17   freopen("output.out", "w", stdout);
18 }
19
20 void Solve() {
21 }
22
23 int32_t main() {
24   ios_base::sync_with_stdio(false);
25   cin.tie(nullptr);
26   // file();
27
28   int t = 1;
29   // cin >> t;
30   for(int i = 1; i <= t; ++i) {
31     Solve();
32   }
33 }
```

6.2 Gcd & Lcm

```
1  i64 gcd(i64 a, i64 b) { // binary GCD uses about 60% fewer bit operations
2    if (!a) return b;
3
4    u64 shift = __builtin_ctzll(a | b);
5    a >>= __builtin_ctzll(a);
6
7    while (b) {
8      b >>= __builtin_ctzll(b);
9
10     if (a > b)
11       swap(a, b);
12     b -= a;
13   }
14   return a << shift;
15 }
16
17 i64 lcm(i64 a, i64 b) {
18   return a / gcd(a, b) * b;
19 }
```

6.3 Debugging tools

```
1  #define  rforeach(_it, c)    for(__typeof((c).rbegin()) _it = (c).rbegin(); _it != (c).rend(); ++_it)
2  #define  foreach(_it, c)     for(__typeof((c).begin()) _it = (c).begin(); _it != (c).end(); ++_it)
3  #define  all(a)              (a).begin(), (a).end()
4  #define  sz(a)               (int)a.size()
5  #define  endl                '\n'
6
7  typedef int64_t  ll;
8
9  template <typename F, typename S>
10 ostream & operator << (ostream & os, const pair <F, S> & p)
11 { return os << "(" << p.first << ", " << p.second << ")"; }
12
13 template <typename F, typename S>
14 ostream & operator << (ostream & os, const map <F, S> & _mp)
15 { os << "["; foreach(it, _mp) { if(it != _mp.begin()) os << ", "; os << it->first << " = " << it->
       second; } return os << "]"; }
16
17 template <typename T>
18 ostream & operator << (ostream & os, const vector <T> & _v)
19 { os << "["; foreach(it, _v) { if(it != _v.begin()) os << ", "; os << *it; } return os << "]"; }
20
21 template <typename T>
22 ostream & operator << (ostream & os, const set <T> & _st)
23 { os << "["; foreach(it, _st) { if(it != _st.begin() ) os << ", "; os << *it; } return os << "]"; }
24
25 template <typename T, size_t S>
26 ostream & operator << (ostream & os, const array <T, S> & _ar)
27 { os << "["; foreach(it, _ar) { if(it != _ar.begin() ) os << ", "; os << *it; } return os << "]"; }
28
29 template <typename T> void write(T _begin, T _end)
30 { for(auto i = _begin; i != _end; ++i) cout << (*i) << ' '; cout << endl; }
31
32 template <typename T> void read(T _begin, T _end)
33 { for(auto i = _begin; i != _end; ++i) cin >> (*i); }
```

6.4 Pseudo random number generator

```
1  /** pseudo-random number generator | C++xx >= C++11 **/
2
3  mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
4
5  T myRand(T a, T b) {
6    return uniform_int_distribution <T> (a, b)(rng);
7  }
```

6.5 Stress test

```
1  g++ -o A A.cpp
2  g++ -o B B.cpp
3  g++ -o gen gen.cpp
4  for ((i = 1; ; ++i)); do  # if they are same then will loop forever
5    echo $i
6    ./gen $i > int
7    ./A < int > out1
8    ./B < int > out2
9    #diff -w out1 out2 || break
10    diff -w <(./A < int) <(./B < int) || break
11 done
```