ICPC Team Reference Material

Contents

1 Setup

1.1 Vimrc

```
let mapleader = " "
syntax on
filetype plugin on
set nocompatible
set autoread
set foldmethod=marker
set autoindent
set clipboard+=unnamedplus
set number relativenumber
colorscheme  desert
set cursorline
set shiftwidth=2 softtabstop=2 expandtab
map cr :w! && !compile %:p:r<CR>
map cx ggVGy
vmap < <gv
vmap > >gv
autocmd TextChanged,TextChangedI * write
set undofile
set undodir=~/.vim/undo
set undolevels=1000
set undoreload=10000
```

1.2 Compilation

```
#!/bin/bash
# put this file in .local/bin or add its dir to the PATH variable
compile() {
  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}
}
compile "$1"
```

2 Data Structure

2.1 segmented tree

```
class SegmentTree {
public:
  SegmentTree(int n) {
    size = 1;
    while (size < n) size <<= 1;
    sums.assign(2 * size, 0LL);
    mins.assign(2 * size, LLONG_MAX);
    maxs.assign(2 * size, LLONG_MIN);
  }

  void build(const vi &a) {
    build(a, 0, 0, size);
  }

  void update(int i, int v) {
    update(i, v, 0, 0, size);
  }

  ll sumSeg(int l, int r) {
    return sumSeg(l, r, 0, 0, size);
  }

  ll minSeg(int l, int r) {
    return minSeg(l, r, 0, 0, size);
  }

  ll maxSeg(int l, int r) {
    return maxSeg(l, r, 0, 0, size);
  }

private:
  int size;
  vector<ll> sums, mins, maxs;

  void build(const vi &a, int x, int lx, int rx) {
    if (rx - lx == 1) {
      if (lx < (int)a.size()) {
        sums[x] = a[lx];
        mins[x] = a[lx];
        maxs[x] = a[lx];
      }
      return;
    }
    int mid = (lx + rx) / 2;
    build(a, 2 * x + 1, lx, mid);
    build(a, 2 * x + 2, mid, rx);
    sums[x] = sums[2 * x + 1] + sums[2 * x + 2];
    mins[x] = min(mins[2 * x + 1], mins[2 * x + 2]);
    maxs[x] = max(maxs[2 * x + 1], maxs[2 * x + 2]);
  }

  void update(int i, int v, int x, int lx, int rx) {
    if (rx - lx == 1) {
      sums[x] = v;
      mins[x] = v;
      maxs[x] = v;
      return;
    }
    int mid = (lx + rx) / 2;
    if (i < mid) {
      update(i, v, 2 * x + 1, lx, mid);
    } else {
      update(i, v, 2 * x + 2, mid, rx);
    }
    sums[x] = sums[2 * x + 1] + sums[2 * x + 2];
    mins[x] = min(mins[2 * x + 1], mins[2 * x + 2]);
    maxs[x] = max(maxs[2 * x + 1], maxs[2 * x + 2]);
  }

  ll sumSeg(int l, int r, int x, int lx, int rx) {
    if (lx >= r || l >= rx) return 0;
    if (lx >= l && rx <= r) return sums[x];
    int mid = (lx + rx) / 2;
    ll left = sumSeg(l, r, 2 * x + 1, lx, mid);
    ll right = sumSeg(l, r, 2 * x + 2, mid, rx);
    return left + right;
  }

  ll minSeg(int l, int r, int x, int lx, int rx) {
    if (lx >= r || l >= rx) return LLONG_MAX;
    if (lx >= l && rx <= r) return mins[x];
    int mid = (lx + rx) / 2;
    ll left = minSeg(l, r, 2 * x + 1, lx, mid);
    ll right = minSeg(l, r, 2 * x + 2, mid, rx);
    return min(left, right);
  }

  ll maxSeg(int l, int r, int x, int lx, int rx) {
    if (lx >= r || l >= rx) return LLONG_MIN;
    if (lx >= l && rx <= r) return maxs[x];
    int mid = (lx + rx) / 2;
    ll left = maxSeg(l, r, 2 * x + 1, lx, mid);
    ll right = maxSeg(l, r, 2 * x + 2, mid, rx);
    return max(left, right);
  }
};
```

3 Graph algorithms

3.1 Dfs

```
vector<vector<int>> graph;
vector<bool> visited;

// manual
void dfs(int start) {
    stack<int> stack;
    stack.push(start);

    while (!stack.empty()) {
        int node = stack.top();
        stack.pop();

        if (!visited[node]) {
            cout << char(node + 'A') << ' ';
            visited[node] = true;
        }

        for (auto it = graph[node].rbegin(); it != graph[node].rend(); ++it) {
            if (!visited[*it]) {
                stack.push(*it);
            }
        }
    }
}

// with recursion
void dfs(int node) {
    visited[node] = true;
    cout << node << ' ';

    for (int neighbor : graph[node]) {
        if (!visited[neighbor]) {
            dfs(neighbor);
        }
    }
}
```

3.2 Bfs

```
void bfs(int start) {
    queue<int> q;

    visited[start] = true;
    q.push(start);

    while (!q.empty()) {
        int curr = q.front();
        q.pop();
        cout << char(curr + 'A') << ' ';

        for (int neighbor : graph[curr]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }
    }
}
```

3.3 Dijkstra’s

```
#include <bits/stdc++.h>
#include "debug.hpp"
using namespace std;
#define vi vector<int>
#define OO 1e8
#define pii pair<int, int>
int n;
vector<vector<pair<int, int> > > adj;
void addEdge(int u, int v, int w) {
  adj[u].emplace_back(v, w);
  adj[v].emplace_back(u, w);
}
vi visited;
vi dijkstra(int x) {
  vi distance(adj.size(), OO);
  priority_queue<pii, vector<pii>, greater<> > q;
  q.emplace(0, x);
  distance[x] = 0;
  q.emplace(0, x);
  while (!q.empty()) {
    int a = q.top().second;

    q.pop();
    if (visited[a]) continue;
    visited[a] = true;
    for (auto u : adj[a]) {
      int b = u.first, w = u.second;
      if (distance[a] + w < distance[b]) {
        distance[b] = distance[a] + w;
        q.emplace(distance[b], b);
        debug(distance);
      }
    }
  }
  return distance;
}
void printSolution(vi dist) {
  cout << "Vertex \t Distance from Source" << endl;
  for (int i = 0; i < n; i++) cout << i << " \t\t-->\t\t" << dist[i] << endl;
}
int main() {
  n = 14;
  adj.resize(n);
  visited.resize(n);

  addEdge(0, 1, 4);
  addEdge(0, 7, 8);
  addEdge(1, 2, 8);
  addEdge(1, 7, 11);
  addEdge(2, 3, 7);
  printSolution(dijkstra(0));
  return 0;
}
```

4 Mathematics

4.1 ncr

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

template<class T>
using rpq = priority_queue<T, vector<T>, greater<T>>;

template<int32_t mod>
struct mint {
    using Z = mint;
    int32_t x;
    mint(int32_t x = 0) : x(norm(x)) {}
    mint(long long x) : x(norm(x % mod)) {}
    inline int32_t norm(int32_t x) const {
        return x >= mod ? x - mod : (x < 0 ? x + mod : x);
    }
    Z power(long long b) const {
        Z res = 1, a = x;
        for (; b; b >>= 1, a *= a)
            if (b & 1) res *= a;
        return res;
    }
    Z inv() const { return assert(x != 0), power(mod - 2); }
    Z operator-() const { return -x; }
    Z &operator*=(const Z &r) { return *this = (long long) x * r.x; }
    Z &operator+=(const Z &r) { return *this = x + r.x; }
    Z &operator-=(const Z &r) { return *this = x - r.x; }
    Z &operator/=(const Z &r) { return *this *= r.inv(); }
    friend Z operator*(const Z &l, const Z &r) { return Z(l) *= r; }
    friend Z operator+(const Z &l, const Z &r) { return Z(l) += r; }
    friend Z operator-(const Z &l, const Z &r) { return Z(l) -= r; }
    friend Z operator/(const Z &l, const Z &r) { return Z(l) /= r; }
    friend ostream &operator<<(ostream &os, const Z &a) { return os << a.x; }
    friend istream &operator>>(istream &is, Z &a) {
        long long y = 0;
        return is >> y, a = y, is;
    }
};

// constexpr int MOD = 998244353;
constexpr int MOD = 1000000007;
using Z = mint<MOD>;

vector<Z> fact = {1};
vector<Z> fact_inv = {1};

void build_fact(int n = 1e6) {
    while ((int) fact.size() < n + 1)
        fact.push_back(fact.back() * (int) fact.size());
    fact_inv.resize(fact.size());
    fact_inv.back() = fact.back().inv();
    for (int j = fact_inv.size() - 2; fact_inv[j].x == 0; j--)
        fact_inv[j] = fact_inv[j + 1] * (j + 1);
}

Z ncr(int n, int r) {
    if (r > n || r < 0) return 0;
    if ((int) fact.size() < n + 1) build_fact(n);
    return fact[n] * fact_inv[r] * fact_inv[n - r];
}

```

```
Z npr(int n, int r) {
    if (r > n || r < 0) return 0;
    if ((int)fact.size() < n + 1) build_fact(n);
    return fact[n] * fact_inv[n - r];
}
```

4.2 fastpower

```
int fast_power(int a, int b) {
    int res = 1;
    while (b) {
        if (b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}
```

4.3 simple-sieve

```
const int NMAX = 1000000;
bitset<NMAX / 2> bits;

void precalcseive() {
  bits.set();
  for (int i = 3; i / 2 < bits.size(); i = 2 * bits._Find_next(i / 2) + 1) {
    for (auto j = (int64_t)i * i / 2; j < bits.size(); j += i)
      bits[j] = 0;
  }
}

//count all the divisors of a number
int divCount(int n) {
  int total = 1;
  int count = 0;
  int p = 2;
  if (n % p == 0) {
    while (n % p == 0) {
      n = n / p;
      count++;
    }
    total = total * (count + 1);
  }
  for (p = 3; p <= n; p += 2) {
    if (bits[p / 2]) {
      count = 0;
      if (n % p == 0) {
        while (n % p == 0) {
          n = n / p;
          count++;
        }
        total = total * (count + 1);
      }
    }
  }
  return total;
}
```

4.4 calculate all divisors

```
void divs() {
    const int n = 1e6;
    vector<vector<int>> divs(n);

    for (int i = 1; i < n; i++) {
        for (int j = i; j < n; j += i) {
            divs[j].push_back(i);
        }
    }

    // all divisors of 12
    for (auto fact : divs[12])
        cout << fact << endl;
}
```

4.5 calculate all prime factors

```
{
  map<int, int> primes;
  for (int i = 2; i * i <= n; i++) {
    while (n % i == 0) {
      primes[i]++; // i is a prime
      n /= i;
    }
  }

  if (n != 1) {
    primes[n]++;
  }

  for (auto [a, b] : primes)
    cout << a << endl;
}
```

5 Geometry

6 Miscellaneous

6.1 C++ template

```
#include <bits/stdc++.h>

#define endl '\n'
#define int long long
#define ld  long double
#define all(a)  (a).begin(), (a).end()
#define sz(a)  (int)(a).size()
#define pb push_back
#define F first
#define S second
#define vi vector<int>

using namespace std;

void file() {
  freopen("input.in", "r", stdin);
  freopen("output.out", "w", stdout);
}

void Solve() {
}

int32_t main() {
  ios_base::sync_with_stdio(false);
  cin.tie(nullptr);
  // file();

  int t = 1;
  // cin >> t;
  for(int i = 1; i <= t; ++i) {
    Solve();
  }
}
```

6.2 Gcd & Lcm

```
i64 gcd(i64 a, i64 b) { // binary GCD uses about 60% fewer bit operations
  if (!a) return b;

  u64 shift = __builtin_ctzll(a | b);
  a >>= __builtin_ctzll(a);

  while (b) {
    b >>= __builtin_ctzll(b);

    if (a > b)
      swap(a, b);
    b -= a;
  }
  return a << shift;
}
```

```

i64 lcm(i64 a, i64 b) {
  return a / gcd(a, b) * b;
}
```

6.3 Debugging tools

```cpp
#define  rforeach(_it, c)    for(__typeof((c).rbegin()) _it = (c).rbegin(); _it != (c).rend(); ++_it)
#define  foreach(_it, c)     for(__typeof((c).begin()) _it = (c).begin(); _it != (c).end(); ++_it)
#define  all(a)              (a).begin(), (a).end()
#define  sz(a)               (int)a.size()
#define  endl                '\n'

typedef int64_t  ll;

template <typename F, typename S>
ostream & operator << (ostream & os, const pair <F, S> & p)
{ return os << "(" << p.first << ", " << p.second << ")"; }

template <typename F, typename S>
ostream & operator << (ostream & os, const map <F, S> & _mp)
{ os << "["; foreach(it, _mp) { if(it != _mp.begin()) os << ", "; os << it->first << " = " << it->second; } return os << "]"; }

template <typename T>
ostream & operator << (ostream & os, const vector <T> & _v)
{ os << "["; foreach(it, _v) { if(it != _v.begin()) os << ", "; os << *it; } return os << "]"; }

template <typename T>
ostream & operator << (ostream & os, const set <T> & _st)
{ os << "["; foreach(it, _st) { if(it != _st.begin() ) os << ", "; os << *it; } return os << "]"; }

template <typename T, size_t S>
ostream & operator << (ostream & os, const array <T, S> & _ar)
{ os << "["; foreach(it, _ar) { if(it != _ar.begin() ) os << ", "; os << *it; } return os << "]"; }

template <typename T> void write(T _begin, T _end)
{ for(auto i = _begin; i != _end; ++i) cout << (*i) << ' '; cout << endl; }

template <typename T> void read(T _begin, T _end)
{ for(auto i = _begin; i != _end; ++i) cin >> (*i); }
```

6.4 Pseudo random number generator

```cpp
/** pseudo-random number generator | C++xx >= C++11 **/

mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

T myRand(T a, T b) {
  return uniform_int_distribution <T> (a, b)(rng);
}
```

6.5 Stress test

```bash
g++ -o A A.cpp
g++ -o B B.cpp
g++ -o gen gen.cpp
for ((i = 1; ; ++i)); do  # if they are same then will loop forever
    echo $i
    ./gen $i > int
    ./A < int > out1
    ./B < int > out2
    #diff -w out1 out2 || break
     diff -w <(./A < int) <(./B < int) || break
done
```