# .*RU.*

Team Reference Document

24/09/2023

Contents



## 1. Code Templates

**1.1. C++ Header.** A C++ header.

```
#pragma GCC optimize("Ofast","unroll-loops") --------------//c2
#pragma GCC target("avx2,fma") ----------------------------//ca
#include <bits/stdc++.h> ----------------------------------//82
#include <ext/pb_ds/assoc_container.hpp> ------------------//4f
#include <ext/pb_ds/tree_policy.hpp> ----------------------//c5
using namespace std; --------------------------------------//a0
using namespace __gnu_pbds; -------------------------------//a2
#define mp make_pair --------------------------------------//3b
typedef pair<int, int> ii; --------------------------------//3c
typedef vector<int> vi; -----------------------------------//c5
typedef vector<ii> vii; -----------------------------------//a2
typedef long long ll; -------------------------------------//5c
typedef tree<int, null_type, less<int>, rb_tree_tag, \ ----//0f
- tree_order_statistics_node_update> orderedTree; ---------//1f
const double pi = acos(-1); -------------------------------//70
mt19937 rng(chrono::steady_clock::now(). ------------------//4e
- time_since_epoch().count()); ----------------------------//91
// usage: rng() -------------------------------------------//97
```

## 2. Data Structures

**2.1. Disjoint Set Union.**

```
struct DSU ------------------------------------------------//c9
{ ---------------------------------------------------------//cc
- int *parent; --------------------------------------------//11
- DSU(int n) ----------------------------------------------//c9
- { -------------------------------------------------------//b7
--- parent = new int [n+1]; -------------------------------//a5
--- for(int i=0; i<=n; i++) -------------------------------//02
----- parent[i] = i; --------------------------------------//ce
- } -------------------------------------------------------//86
 ----------------------------------------------------------//20
- int findSet(int id) -------------------------------------//fa
- { -------------------------------------------------------//20
--- if(parent[id]==id) ------------------------------------//37
----- return id; ------------------------------------------//2f
--- else --------------------------------------------------//2b
----- return parent[id] = findSet(parent[id]); ------------//d9
- } -------------------------------------------------------//75
 ----------------------------------------------------------//7c
- void unionSets(int i, int j) ----------------------------//4a
- { -------------------------------------------------------//03
--- if(findSet(i)==findSet(j)) ----------------------------//26
----- return; ---------------------------------------------//cf
--- else --------------------------------------------------//49
----- parent[findSet(i)] = findSet(j); --------------------//b6
- } -------------------------------------------------------//80
}; --------------------------------------------------------//f3
 ----------------------------------------------------------//1c
 ----------------------------------------------------------//f5
int main() ------------------------------------------------//1c
{ ---------------------------------------------------------//aa
- int n; --------------------------------------------------//ed
- cin >> n; -----------------------------------------------//f4
- DSU naujas(n); ------------------------------------------//4a
- return 0; -----------------------------------------------//33
} ---------------------------------------------------------//a7
```

**2.2. Lazy Segment Tree.**

```
struct node -----------------------------------------------//74
{ ---------------------------------------------------------//32
- int start, finish; --------------------------------------//2b
- long long value, lazy; ----------------------------------//9d
- node *left, *right; -------------------------------------//e6
- node() { } ----------------------------------------------//2d
- node(int pr, int pb, int A[]) ---------------------------//06
- { -------------------------------------------------------//9f
--- start = pr; -------------------------------------------//2a
--- finish = pb; ------------------------------------------//51
--- lazy = 0; ---------------------------------------------//bb
--- if(start == finish) -----------------------------------//bf
--- { -----------------------------------------------------//b0
----- left = NULL; ----------------------------------------//1f
----- right = NULL; ---------------------------------------//e5
----- value = A[start]; -----------------------------------//10
--- } -----------------------------------------------------//b1
--- else --------------------------------------------------//28
--- { -----------------------------------------------------//43
----- left = new node(pr, (pr + pb) / 2, A); --------------//2c
----- right = new node((pr + pb) / 2+1, pb, A); -----------//16
----- value = min(left->value, right->value); -------------//a7
--- } -----------------------------------------------------//f2
- } -------------------------------------------------------//a9
- long long get(int pr, int pb) ---------------------------//12
- { -------------------------------------------------------//08
--- fix(); ------------------------------------------------//12
--- if((pr <= start) && (finish <= pb)) -------------------//27
----- return value; ---------------------------------------//9f
--- else if ((finish < pr) || (pb < start)) ---------------//3b
----- return INT_MAX; -------------------------------------//5b
--- else --------------------------------------------------//76
----- return min(left->get(pr, pb), right->get(pr, pb)); --//a9
- } -------------------------------------------------------//a6
- void fix() -----------------------------------------------//c3
- { -------------------------------------------------------//0c
--- if(lazy != 0) -----------------------------------------//49
--- { -----------------------------------------------------//86
----- if(left != NULL) ------------------------------------//19
----- { ---------------------------------------------------//b2
------- left->lazy += lazy; -------------------------------//0d
------- left->value += lazy; ------------------------------//0f
------- right->lazy += lazy; ------------------------------//d0
------- right->value += lazy; -----------------------------//ce
----- } ---------------------------------------------------//a3
----- lazy = 0; -------------------------------------------//60
--- } -----------------------------------------------------//df
- } -------------------------------------------------------//7c
- void update(int pr, int pb, long long delta) ------------//cf
- { -------------------------------------------------------//f0
--- fix(); ------------------------------------------------//6c
--- if((pr <= start) && (finish <= pb)) -------------------//22
--- { -----------------------------------------------------//e7
------- lazy += delta; ------------------------------------//53
------- value += delta; -----------------------------------//53
--- } -----------------------------------------------------//5e
--- else if ((finish < pr) || (pb < start)) ---------------//eb
--- { -----------------------------------------------------//d5
----- return; ---------------------------------------------//b1
--- } -----------------------------------------------------//77
--- else --------------------------------------------------//20
--- { -----------------------------------------------------//90
----- left->update(pr, pb, delta); ------------------------//e0
----- right->update(pr, pb, delta); -----------------------//14
----- value = min(left->value, right->value); -------------//51
--- } -----------------------------------------------------//4d
- } -------------------------------------------------------//82
}; --------------------------------------------------------//38
```

**2.3. Sparse Table.**

```
#include<bits/stdc++.h> -----------------------------------//84
using namespace std; --------------------------------------//16
const int max_N = 1000003; --------------------------------//a9
const int logn = 22; --------------------------------------//51
int lookup[max_N][logn]; ----------------------------------//c5
int loga[max_N]; ------------------------------------------//9a
void buildsparsetable(int N, int a[]) ---------------------//19
{ ---------------------------------------------------------//25
- for(int i = 0; i < N; i++) ------------------------------//d4
--- lookup[i][0] = a[i]; ----------------------------------//ed
- for(int j = 1; j < logn; j++) ---------------------------//4e
- { -------------------------------------------------------//ff
```

```
--- for(int i = 0; i + (1<<j) <= N; i++) ------------------//70
----- lookup[i][j] = min(lookup[i][j - 1], ----------------//4c
-----         lookup[i + (1<<(j - 1))][j - 1]); -----------//02
- } -------------------------------------------------------//c2
- loga[1] = 0; --------------------------------------------//7c
- for(int i = 2; i <= N; i++) -----------------------------//30
--- loga[i] = loga[i / 2] + 1; ----------------------------//37
} ---------------------------------------------------------//90
 ----------------------------------------------------------//a1
int query(int L, int R) -----------------------------------//7a
{ ---------------------------------------------------------//ea
- int sk = loga[R - L + 1]; -------------------------------//64
- return min(lookup[L][sk], lookup[R - (1<<sk) + 1][sk]); -//8d
} ---------------------------------------------------------//c0
```

2.4. **Trie.**

```
const int numberOfChildren = 12; --------------------------//56
struct Trie -----------------------------------------------//37
{ ---------------------------------------------------------//60
- struct Node ---------------------------------------------//73
- { -------------------------------------------------------//7c
--- int count = 0; ----------------------------------------//4a
--- Node* child[numberOfChildren]; ------------------------//e2
--- Node() ------------------------------------------------//f2
--- { -----------------------------------------------------//85
----- for(int i = 0; i < numberOfChildren; i++) -----------//12
------- child[i] = NULL; ----------------------------------//2c
--- } -----------------------------------------------------//dd
- }; ------------------------------------------------------//a4
- Node *root; ---------------------------------------------//a8
 ----------------------------------------------------------//f3
- Trie() --------------------------------------------------//c1
- { -------------------------------------------------------//c1
--- root = new Node(); ------------------------------------//11
- } -------------------------------------------------------//94
 ----------------------------------------------------------//93
 ----------------------------------------------------------//fb
- void add(vector<int> sequence) --------------------------//68
- { -------------------------------------------------------//cf
--- root->count++; ----------------------------------------//ec
--- Node* cur = root; -------------------------------------//e9
--- for(int i = 0; i < sequence.size(); i++) --------------//c3
--- { -----------------------------------------------------//87
----- int next = sequence[i]; -----------------------------//9e
----- if(cur->child[next] == NULL) ------------------------//e0
----- { ---------------------------------------------------//c8
------- cur->child[next] = new Node(); --------------------//b4
----- } ---------------------------------------------------//4e
----- cur = cur->child[next]; -----------------------------//c2
----- cur->count++; ---------------------------------------//20
--- } -----------------------------------------------------//b4
- } -------------------------------------------------------//49
 ----------------------------------------------------------//b8
- int numberOfOccurences(vector<int> sequence) ------------//d8
- { -------------------------------------------------------//cd
--- Node* cur = root; -------------------------------------//96
--- for(int i = 0; i < sequence.size(); i++) --------------//aa
--- { -----------------------------------------------------//3c
----- int next = sequence[i]; -----------------------------//a6
----- if(cur->child[next] == NULL) ------------------------//ec
------- return 0; -----------------------------------------//ee
----- cur = cur->child[next]; -----------------------------//d1
--- } -----------------------------------------------------//1b
--- return cur->count; ------------------------------------//15
- } -------------------------------------------------------//11
}; --------------------------------------------------------//e7
```

## 3. Graph Theory

3.1. **Maximum Bipartite Matching.** Kopcroft-Karp algorithm. Time complexity is $O(E\sqrt{V})$

```
#include <bits/stdc++.h> ----------------------------------//84
 ----------------------------------------------------------//8e
#define mp make_pair --------------------------------------//d9
using namespace std; --------------------------------------//a2
 ----------------------------------------------------------//1a
const int maxN = 1e6; -------------------------------------//20
// HopCroft Karp algorithm --------------------------------//71
// abi puses numeruojamos nuo 0. --------------------------//55
// Kaires dydis - |U|, desines - |V| ----------------------//20
vii bipartiteMatching(vii edgeList, int U, int V) ---------//21
{ ---------------------------------------------------------//c4
- vector<int> pairFromU(U, -1), pairFromV(V, -1); ---------//cc
- int d[U]; -----------------------------------------------//19
- vector<int> adjU[U]; ------------------------------------//4b
- for(auto e : edgeList) ----------------------------------//01
--- adjU[e.first].push_back(e.second); --------------------//57
 ----------------------------------------------------------//83
- while(true) ---------------------------------------------//7b
- { -------------------------------------------------------//db
--- const int INF = 1e9; ----------------------------------//d0
--- int minDist = INF; ------------------------------------//98
--- for(int i = 0; i < U; i++) ---------------------------//4f
----- d[i] = minDist; -------------------------------------//12
--- queue<int> q; -----------------------------------------//b3
--- for(int i = 0; i < U; i++) ---------------------------//a6
--- { -----------------------------------------------------//c9
----- if(pairFromU[i] == -1) ------------------------------//95
----- { ---------------------------------------------------//be
------- d[i] = 0; -----------------------------------------//93
------- q.push(i); ----------------------------------------//97
----- } ---------------------------------------------------//73
--- } -----------------------------------------------------//b0
 ----------------------------------------------------------//d8
--- while(!q.empty()) -------------------------------------//a1
--- { -----------------------------------------------------//bf
----- int g = q.front(); ----------------------------------//7e
----- q.pop(); --------------------------------------------//e6
----- for(int v : adjU[g]) --------------------------------//f5
----- { ---------------------------------------------------//56
------- if(pairFromV[v] == -1) ----------------------------//dc
--------- minDist = min(minDist, d[g]); -------------------//af
------- else if(d[pairFromV[v]] > d[g] + 1) ---------------//f6
------- { -------------------------------------------------//20
--------- d[pairFromV[v]] = d[g] + 1; ---------------------//01
--------- q.push(pairFromV[v]); ---------------------------//19
------- } -------------------------------------------------//51
----- } ---------------------------------------------------//8a
--- } -----------------------------------------------------//2a
--- if(minDist == INF) ------------------------------------//a2
----- break; ----------------------------------------------//cc
 ----------------------------------------------------------//ba
--- bool visited[U] = {}; ---------------------------------//68
 ----------------------------------------------------------//c7
--- function<bool(int)> dfs = [&](int u) ------------------//b4
--- { -----------------------------------------------------//e0
----- if(visited[u]) --------------------------------------//57
------- return false; -------------------------------------//44
----- visited[u] = true; ----------------------------------//fa
----- for(int v : adjU[u]) --------------------------------//2d
----- { ---------------------------------------------------//46
------- if(pairFromV[v] == -1) ----------------------------//38
------- { -------------------------------------------------//94
--------- pairFromV[v] = u; -------------------------------//d0
--------- pairFromU[u] = v; -------------------------------//0a
--------- return true; ------------------------------------//4e
------- } -------------------------------------------------//51
------- else if ((d[pairFromV[v]] == d[u] + 1) ------------//7e
-------- && (dfs(pairFromV[v]))) --------------------------//cd
------- { -------------------------------------------------//a1
--------- pairFromU[u] = v; -------------------------------//5a
--------- pairFromV[v] = u; -------------------------------//74
--------- return true; ------------------------------------//e3
------- } -------------------------------------------------//ea
----- } ---------------------------------------------------//79
----- return false; ---------------------------------------//c6
--- }; ----------------------------------------------------//3e
 ----------------------------------------------------------//f0
 ----------------------------------------------------------//06
--- for(int i = 0; i < U; i++) ---------------------------//cb
--- { -----------------------------------------------------//d9
----- if(pairFromU[i] == -1) ------------------------------//82
------- dfs(i); -------------------------------------------//8b
--- } -----------------------------------------------------//cf
- } -------------------------------------------------------//41
 ----------------------------------------------------------//2c
- vii ans; ------------------------------------------------//e0
- for(int i = 0; i < V; i++) ------------------------------//d3
- { -------------------------------------------------------//d1
--- if(pairFromV[i] != -1) --------------------------------//2e
----- ans.push_back(mp(pairFromV[i], i)); -----------------//ae
- } -------------------------------------------------------//de
- return ans; ---------------------------------------------//6b
} ---------------------------------------------------------//53
```

3.2. **Hungarian Algorithm.** Finds min weight bipartite matching. $O(n^3)$ complexity.

```
int findMinAssignment(vector<vector<int> > C) -------------//d1
{ ---------------------------------------------------------//d2
```

```
- int n = C.size(), m = C[0].size(); ----------------------//73
- // cout << "n, m = " << n << ", " << m << endl; ---------//cc
- assert(n <= m); -----------------------------------------//a8
- int a[n + 1][m + 1]; ------------------------------------//ef
- for(int i = 1; i <= n; i++) -----------------------------//34
- { -------------------------------------------------------//4e
--- for(int j = 1; j <= m; j++) ---------------------------//06
--- { -----------------------------------------------------//87
----- a[i][j] = C[i - 1][j - 1]; --------------------------//69
--- } -----------------------------------------------------//17
- } -------------------------------------------------------//44
- const int INF = 1e9; ------------------------------------//47
 ----------------------------------------------------------//9d
- vector<int> u(n + 1, 0), v(m + 1, 0), p(m + 1), way(m + 1);
- // p[i] - corresponding row for column i in matching ----//78
- for(int i = 1; i <= n; i++) -----------------------------//cd
- { -------------------------------------------------------//6b
--- p[0] = i; ---------------------------------------------//95
--- int j0 = 0; // free column ----------------------------//94
--- vector<int> minv(m + 1, INF); -------------------------//04
--- vector<bool> used(m + 1, false); // jau panaudotas paieskoje
--- do ----------------------------------------------------//31
--- { -----------------------------------------------------//3d
----- used[j0] = true; ------------------------------------//53
----- int i0 = p[j0], delta = INF, j1; --------------------//08
----- for(int j = 1; j <= m; j++) -------------------------//bc
----- { ---------------------------------------------------//ac
------- if(!used[j]) --------------------------------------//6e
------- { -------------------------------------------------//3a
--------- int cur = a[i0][j] - u[i0] - v[j]; --------------//19
--------- if(cur < minv[j]) -------------------------------//e4
--------- { -----------------------------------------------//2a
----------- minv[j] = cur; --------------------------------//43
----------- way[j] = j0; ----------------------------------//d4
--------- } -----------------------------------------------//34
--------- if(minv[j] < delta) -----------------------------//aa
--------- { -----------------------------------------------//c0
----------- delta = minv[j]; ------------------------------//51
----------- j1 = j; ---------------------------------------//cf
--------- } -----------------------------------------------//36
------- } -------------------------------------------------//d2
----- } ---------------------------------------------------//6c
 ----------------------------------------------------------//be
----- for(int j = 0; j <= m; j++) -------------------------//71
----- { ---------------------------------------------------//09
------- if(used[j]) ---------------------------------------//60
------- { -------------------------------------------------//67
--------- u[p[j]] += delta; -------------------------------//28
--------- v[j] -= delta; ----------------------------------//cc
------- } -------------------------------------------------//d0
------- else ----------------------------------------------//96
--------- minv[j] -= delta; -------------------------------//27
----- } ---------------------------------------------------//18
 ----------------------------------------------------------//5a
----- j0 = j1; --------------------------------------------//73
--- } -----------------------------------------------------//f9
--- while(p[j0] != 0); ------------------------------------//59
 ----------------------------------------------------------//70
--- do ----------------------------------------------------//1c
--- { -----------------------------------------------------//59
----- int j1 = way[j0]; -----------------------------------//c5
----- p[j0] = p[j1]; --------------------------------------//4e
----- j0 = j1; --------------------------------------------//6d
--- } -----------------------------------------------------//f3
--- while(j0); --------------------------------------------//e0
- } -------------------------------------------------------//af
- ---------------------------------------------------------//de
- int cost = -v[0]; ---------------------------------------//91
- return cost; --------------------------------------------//55
 ----------------------------------------------------------//21
} ---------------------------------------------------------//d8
```

## 4. Miscellaneous

### 4.1. Round Robin.

```
vector<vector<pii> > roundRobin(int n) --------------------//52
{ ---------------------------------------------------------//6d
- if(n % 2 == 0) ------------------------------------------//a8
- { -------------------------------------------------------//48
--- vector<vector<pii> > partial = roundRobin(n-1); -------//e1
--- for(int i = 0; i < n - 1; ++i) -----------------------//53
----- partial[i].push_back(make_pair(i, n - 1)); ----------//39
--- return partial; ---------------------------------------//6f
- } -------------------------------------------------------//c2
- vector<vector<pii> > answer(n, vector<pair<int, int>>());//5a
- for(int i = 0; i < n; i++) ------------------------------//13
- { -------------------------------------------------------//09
--- for(int j = 1; 2 * j < n; j++) ------------------------//71
--- { -----------------------------------------------------//ae
----- int a = (i - j), b = i + j; -------------------------//9b
----- if(a < 0) -------------------------------------------//07
------- a += n; -------------------------------------------//ff
----- if(b >= n) ------------------------------------------//d2
------- b -= n; -------------------------------------------//be
----- answer[i].push_back(make_pair(a, b)); ---------------//38
--- } -----------------------------------------------------//51
- } -------------------------------------------------------//3d
- return answer; ------------------------------------------//df
} ---------------------------------------------------------//43
```

## 5. Mathematics

### 5.1. Extended Euclidean Algorithm.

Finds integer solutions $(x, y)$ to $ax + by = \gcd(a, b)$.

```
#include<bits/stdc++.h> -----------------------------------//84
void find(int a, int b) -----------------------------------//1c
{ ---------------------------------------------------------//c5
- const int max_log_a = 50; -------------------------------//a4
- int r[max_log_a], s[max_log_a], t[max_log_a]; -----------//63
- r[0] = a; -----------------------------------------------//db
- r[1] = b; -----------------------------------------------//ff
- s[0] = 1; -----------------------------------------------//d5
- s[1] = 0; -----------------------------------------------//58
- t[0] = 0; -----------------------------------------------//d6
- t[1] = 1; -----------------------------------------------//77
- int k = 1; ----------------------------------------------//9c
- for(int i=2; i<max_log_a; i++) --------------------------//72
- { -------------------------------------------------------//5d
--- if(r[i-1]==0) -----------------------------------------//fe
----- break; ----------------------------------------------//38
--- k = i-1; ----------------------------------------------//e8
--- int q = r[i-2]/r[i-1]; --------------------------------//09
--- s[i] = s[i-2] - q*s[i-1]; -----------------------------//bf
--- t[i] = t[i-2] - q*t[i-1]; -----------------------------//45
--- r[i] = r[i-2] - q*r[i-1]; -----------------------------//10
- } -------------------------------------------------------//3f
- assert(s[k] * a + t[k] * b == r[k]); --------------------//98
- assert(r[k] == __gcd(a, b)); ----------------------------//9e
} ---------------------------------------------------------//16
```

### 5.2. Shoelace's formula.

Finds the area of polygon, given it's vertices in clockwise or counterclockwise order.

```
double Shoelace(double X[], double Y[], int N) ------------//22
{ ---------------------------------------------------------//8c
- double ats = 0; -----------------------------------------//6d
- for(int i=0; i<N; i++) ----------------------------------//db
- { -------------------------------------------------------//24
--- ats += X[i]*Y[(i+1)%N] - X[(i+1)%N]*Y[i]; -------------//6a
- } -------------------------------------------------------//3b
- ats /= (double) 2; --------------------------------------//17
- return abs(ats); ----------------------------------------//a7
} ---------------------------------------------------------//0e
```