Northern University of Bussiness and Technology Khulna

NUBTK $(^w^)$

Sourav Mondal, Rahul Kumar Ghosh, Ayon Adikary Arko

NWU CSE FEST 2025

November 11, 2025

Contest (1)

template.cpp 13 lines

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

#ifdef LOCAL
#include "debug.h"
#else
#define dbg(...)
#endif

int32_t main() {
  cin.tie(0)->sync_with_stdio(0);
  cin.exceptions(cin.failbit);
}
```

.bashrc 5 lines

```
# write every function in single line
alias clr="printf '\33c'"
co() { g++ -std=c++23 -O2 -Wall -Wextra
 -Wshadow -Wconversion -o $1 $1.cpp; }
run() { co $1 && ./$1; }
```

hash.sh 3 lines

```
# Hash file ignoring whitespace and comments. Verifies that
# code was correctly typed. Usage: 'sh hash.sh < A.cpp'
cpp -dD -P -fpreprocessed|tr -d '[:space:]'|md5sum|cut -c-6
```

stress.sh 20 lines

```
#!/usr/bin/env bash

for ((testNum=0;testNum<$4;testNum++))
do
    "./$3".out > input
    "./$2".out < input > outSlow
    "./$1".out < input > outWrong
    if !(cmp -s "outWrong" "outSlow")
    then
        echo "Error found!"
        echo "Input:"
        cat input
        echo "Wrong Output:"
        cat outWrong
        echo "Slow Output:"
        cat outSlow
        exit
    fi
done
echo Passed $4 tests
```

troubleshoot.txt 75 lines

```
General:
* Write down most of your thoughts, even if you're not sure
whether they're useful.
* Give your variables (and files) meaningful names.
* Stay organized and don't leave papers all over the place!
* You should know what your code is doing ...

Pre-submit:
* Write a few simple test cases if sample is not enough.
* Are time limits close? If so, generate max cases.
* Is the memory usage fine?
* Could anything overflow?
* Remove debug output.
* Make sure to submit the right file.

Wrong answer:
* Print your solution! Print debug output as well.
* Read the full problem statement again.
* Have you understood the problem correctly?
* Are you sure your algorithm works?
* Try writing a slow (but correct) solution.
* Can your algorithm handle the whole range of input?
* Did you consider corner cases (ex. n=1)?
* Is your output format correct? (including whitespace)
* Are you clearing all data structures between test cases?
* Any uninitialized variables?
* Any undefined behavior (array out of bounds)?
* Any overflows or NaNs (or shifting ll by >=64 bits)?
* Confusing N and M, i and j, etc.?
* Confusing ++i and i++?
* Return vs continue vs break?
* Are you sure the STL functions you use work as you think?
* Add some assertions, maybe resubmit.
* Create some test cases to run your algorithm on.
* Go through the algorithm for a simple case.
* Go through this list again.
* Explain your algorithm to a teammate.
* Ask the teammate to look at your code.
* Go for a small walk, e.g. to the toilet.
* Rewrite your solution from the start or let a teammate do it.

Geometry:
* Work with ints if possible.
* Correctly account for numbers close to (but not) zero.
   ↪Related:
for functions like acos make sure absolute val of input is not
(slightly) greater than one.
* Correctly deal with vertices that are collinear, concyclic,
coplanar (in 3D), etc.
* Subtracting a point from every other (but not itself)?

Runtime error:
* Have you tested all corner cases locally?
* Any uninitialized variables?
* Are you reading or writing outside the range of any vector?
* Any assertions that might fail?
* Any possible division by 0? (mod 0 for example)
* Any possible infinite recursion?
* Invalidated pointers or iterators?
* Are you using too much memory?
* Debug with resubmits (e.g. remapped signals, see Various).

Time limit exceeded:
* Do you have any possible infinite loops?
* What's your complexity? Large TL does not mean that something
simple (like NlogN) isn't intended.
* Are you copying a lot of unnecessary data? (References)
* Avoid vector, map. (use arrays/unordered_map)
* How big is the input and output? (consider FastIO)
* What do your teammates think about your algorithm?
* Calling count() on multiset?

Memory limit exceeded:
* What is the max amount of memory your algorithm should need?
* Are you clearing all data structures between test cases?
* If using pointers try BumpAllocator.
```

Mathematics (2)

2.1 Trigonometry

$$\sin(v+w) = \sin v\cos w + \cos v\sin w$$
$$\cos(v+w) = \cos v\cos w - \sin v\sin w$$
$$\tan(v+w) = \frac{\tan v + \tan w}{1 - \tan v\tan w}$$
$$\sin v + \sin w = 2\sin\frac{v+w}{2}\cos\frac{v-w}{2}$$
$$\cos v + \cos w = 2\cos\frac{v+w}{2}\cos\frac{v-w}{2}$$

$$a\cos x + b\sin x = r\cos(x-\phi)$$
$$a\sin x + b\cos x = r\sin(x+\phi)$$

where $r = \sqrt{a^2+b^2}, \phi = \text{atan2}(b,a)$.

2.2 Geometry

2.2.1 Triangles

Side lengths: a, b, c

Semiperimeter: $s = \frac{a+b+c}{2}$

Area: $A = \sqrt{s(s-a)(s-b)(s-c)}$

Circumradius: $R = \frac{abc}{4A}$

Inradius: $r = \frac{A}{p}$

Length of median (divides triangle into two equal-area triangles): $m_a = \frac{1}{2}\sqrt{2b^2+2c^2-a^2}$

Length of bisector (divides angles in two):

$$s_a = \sqrt{bc\left[1 - \left(\frac{a}{b+c}\right)^2\right]}$$

Law of sines: $\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c} = \frac{1}{2R}$

Law of cosines: $a^2 = b^2 + c^2 - 2bc\cos\alpha$

Law of tangents: $\frac{a+b}{a-b} = \frac{\tan\frac{\alpha+\beta}{2}}{\tan\frac{\alpha-\beta}{2}}$

2.3 Derivatives/Integrals

$$\frac{d}{dx}\arcsin x = \frac{1}{\sqrt{1-x^2}} \qquad \frac{d}{dx}\arccos x = -\frac{1}{\sqrt{1-x^2}}$$

$$\frac{d}{dx}\tan x = 1+\tan^2 x \qquad \frac{d}{dx}\arctan x = \frac{1}{1+x^2}$$

$$\int \tan ax = -\frac{\ln|\cos ax|}{a} \qquad \int x\sin ax = \frac{\sin ax - ax\cos ax}{a^2}$$

$$\int e^{-x^2} = \frac{\sqrt{\pi}}{2}\text{erf}(x) \qquad \int xe^{ax}dx = \frac{e^{ax}}{a^2}(ax-1)$$

Integration by parts:

$$\int_a^b f(x)g(x)dx = [F(x)g(x)]_a^b - \int_a^b F(x)g'(x)dx$$

2.4 Sums/Series

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, (-1 < x \le 1)$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots, (-1 \le x \le 1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, (-\infty < x < \infty)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, (-\infty < x < \infty)$$

Data Structures (3)

3.1 STL

MapComparator.h

Description: example of function object (functor) for map or set
Usage: `set<int,cmp> s; map<int,int,cmp> m;`

5bfa6c, 1 lines

```
struct cmp{bool operator()(int l,int r)const{return l>r;}};
```

3.2 1D Range Queries

SparseTable.h

Description: Static 1D range (min/max/gcd/lcm/or/and) query. If TL is an issue, use arrays instead of vectors and store values instead of indices.
Memory: $\mathcal{O}(N \log N)$
Time: $\mathcal{O}(1)$

471d4b, 19 lines

```
template<class T> struct SparseTable {
  int N, LOG;
  vector<vector<T>> jmp;
  T cmb(T a, T b) { return min(a, b); }
  SparseTable(const vector<T>& v) {
    N = v.size(); LOG = __lg(N);
    jmp.resize(N, vector<T>(LOG + 1));
    for (int i = 0; i < N; ++i) jmp[i][0] = v[i];
    for (int j = 1; j <= LOG; ++j) {
      for (int i = 0; i + (1<<j) <= N; ++i) {
        jmp[i][j] = cmb(jmp[i][j-1], jmp[i+(1<<(j-1))][j-1]);
      }
    }
  }
  T query(int l, int r) {
    int d = __lg(r-l+1);
    return cmb(jmp[l][d], jmp[r-(1<<d)+1][d]);
  }
};
```

FenwickTree.h

Description: Computes partial sums a[0] + a[1] + ... + a[pos - 1], and updates single elements a[i], taking the difference between the old and new value. 0-Indexed.
Time: Both operations are $\mathcal{O}(\log N)$.

122df3, 23 lines

```
template<class T> struct FenwickTree {
  int N; vector<T> bit;
  FenwickTree(int _N) { N = _N; bit.resize(N); }
  void add(int p, T x) {
    for (++p; p <= N; p += p&-p) bit[p-1] += x;
  }
  T sum(int l, int r) { return sum(r+1)-sum(l); }
  T sum(int r) {
    T res = 0;
    for(; r; r -= r&-r) res += bit[r-1];
    return res;
  }
  int lower_bound(T sum) {
    if (sum <= 0) return -1;
    int pos = 0;
    for (int pw = 1<<25; pw; pw >>= 1) {
      int npos = pos+pw;
      if (npos <= N && bit[npos-1] < sum)
        pos = npos, sum -= bit[pos-1];
    }
    return pos;
  }
};
```

SegmentTree.h

Description: 1D point update and range query where `cmb` is any associative operation. `seg[1]==query(0,N-1)`.
Time: $\mathcal{O}(\log N)$

749f3e, 29 lines

```
template<class T> struct SegTree {
  const T ID{};
  T cmb(T a, T b) { return a + b; }
  int n; vector<T> seg;
  SegTree(int _n) {
    for (n = 1; n < _n; ) n *= 2;
    seg.assign(2*n, ID);
  }
  void pull(int p) { seg[p] = cmb(seg[2*p], seg[2*p+1]); }
  void update(int p, T val) { // set val at position p
    seg[p += n] = val;
    for (p /= 2; p; p /= 2) pull(p);
  }
  T query(int l, int r) { // zero-indexed, inclusive
    T ra = ID, rb = ID;
    for (l += n, r += n + 1; l < r; l /= 2, r /= 2) {
      if (l & 1) ra = cmb(ra, seg[l++]);
      if (r & 1) rb = cmb(seg[--r], rb);
    }
    return cmb(ra, rb);
  }
  // int first_at_least(int lo, int val, int ind, int l, int r)
    ↪ { // if seg stores max across range
  //   if (r < lo || val > seg[ind]) return -1;
  //   if (l == r) return l;
  //   int m = (l+r)/2;
  //   int res = first_at_least(lo,val,2*ind,l,m); if (res !=
    ↪-1) return res;
  //   return first_at_least(lo,val,2*ind+1,m+1,r);
  // }
};
```

LazySegmentTree.h

Description: 1D range increment and sum query.
Usage: `LazySeg<int64_t, 1<<20> T; T.update(l, r, val);`
Time: $\mathcal{O}(\log N)$

e0742f, 42 lines

```
template<class T, int SZ>struct LazySeg {
  // SZ must be power of 2
  static_assert(__builtin_popcount(SZ) == 1);
  const T ID{};
  T cmb(T a, T b) { return a + b; }
  T seg[2*SZ], lazy[2*SZ];
  LazySeg() {
    for (int i = 0; i < 2*SZ; ++i) seg[i] = lazy[i] = ID;
  }
  // modify values for current node
  void push(int ind, int L, int R) {
    seg[ind] += (R-L+1) * lazy[ind]; // dependent on operation
    if (L != R) for (int i = 0; i < 2; ++i) lazy[2*ind+i] +=
      ↪lazy[ind];
    lazy[ind] = 0;
  }
  void pull(int ind) {
    seg[ind] = cmb(seg[2*ind], seg[2*ind+1]);
  }
  void build() {
    for (int i = SZ-1; i >= 1; --i) pull(i);
  }
  void update(int lo, int hi, T inc, int ind = 1, int L = 0,
    ↪int R = SZ - 1) {
    push(ind, L, R);
    if (hi < L || R < lo) return;
    if (lo <= L && R <= hi) {
      lazy[ind] = inc;
      push(ind, L, R);
      return;
    }
    int M = (L + R) / 2;
    update(lo, hi, inc, 2*ind, L, M);
    update(lo, hi, inc, 2*ind+1, M+1, R);
    pull(ind);
  }
  T query(int lo, int hi, int ind = 1, int L = 0, int R = SZ -
    ↪1) {
    push(ind, L, R);
    if (lo > R || L > hi) return ID;
    if (lo <= L && R <= hi) return seg[ind];
    int M = (L + R) / 2;
    return cmb(query(lo, hi, 2*ind, L, M), query(lo, hi, 2*ind
      ↪+1, M+1, R));
  }
};
```

3.3 2D Range Queries

PrefixSum2D.h

Description: calculates rectangle sums in constant time

65b070, 17 lines

```
template<typename T> struct PrefixSum2D {
  vector<vector<T>> sum;
  PrefixSum2D(const vector<vector<T>> &v) {
    int n = v.size(), m = v[0].size();
    sum.assign(n+1, vector<T>(m+1, T{}));
    for (int i = 0; i < n; ++i) {
      for (int j = 0; j < m; ++j) {
        sum[i+1][j+1] = v[i][j]
        - sum[i][j] + sum[i+1][j] + sum[i][j+1];
      }
```

```
    }
  }
  T query(int x1, int y1, int x2, int y2) {
    return sum[x2][y2] + sum[x1-1][y1-1]
    - sum[x1-1][y2] - sum[x2][y1-1];
  }
};
```