



Chulalongkorn University

What Name ?

Pakapim E., Pasin P., Sarit P.

template from KACTL

2025-07-20

Template

(1)

template.cpp

31 lines

```
#pragma once
#include <bits/stdc++.h>
#define sz(x) (int)(x).size()
#define all(x) (x).begin(), (x).end()

using namespace std;

typedef long long ll;
typedef double db;
typedef long double ld;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

template<typename T> bool ckmin(T &a, const T &b) { return b <
    a ? a = b, 1 : 0; }
template<typename T> bool ckmax(T &a, const T &b) { return a <
    b ? a = b, 1 : 0; }

mt19937 rng(chrono::steady_clock::now().time_since_epoch().
    count());

const char nl = '\n';
const int INF = 0x3fffffff;
const int MOD=1000000007;
// const int MOD = 998244353;
const ll LINF = 0x1fffffffffffffff;
const db DINF = numeric_limits<db>::infinity();
const db EPS = 1e-9;
const db PI = acos(db(-1));

signed main(){
    ios_base::sync_with_stdio(0); cin.tie(NULL);
    return 0;
}
```

c.sh

2 lines

```
g++ -std=gnu++2a -Wall $1 -o a.out
./a.out
```

Data Structures

(2)

- OrderedSet.hpp
- SparseTable.hpp
- FenwickTree.hpp
- 2DFenwickTree.hpp
- BinaryTrie.hpp
- StaticTopTree.hpp
- Treap.hpp

Competitive Programming Topics

(A)

topics.txt

159 lines

Recursion

Divide and conquer

Finding interesting points in $N \log N$

Algorithm analysis

Master theorem

Amortized time complexity

Greedy algorithm

Scheduling

Max contiguous subvector sum

Invariants

Huffman encoding

Graph theory

Dynamic graphs (extra book-keeping)

Breadth first search

Depth first search

* Normal trees / DFS trees

Dijkstra's algorithm

MST: Prim's algorithm

Bellman-Ford

Konig's theorem and vertex cover

Min-cost max flow

Lovasz toggle

Matrix tree theorem

Maximal matching, general graphs

Hopcroft-Karp

Hall's marriage theorem

Graphical sequences

Floyd-Warshall

Euler cycles

Flow networks

* Augmenting paths

* Edmonds-Karp

Bipartite matching

Min. path cover

Topological sorting

Strongly connected components

2-SAT

Cut vertices, cut-edges and biconnected components

Edge coloring

* Trees

Vertex coloring

* Bipartite graphs (\Rightarrow trees)

* 3^n (special case of set cover)

Diameter and centroid

K'th shortest path

Shortest cycle

Dynamic programming

Knapsack

Coin change

Longest common subsequence

Longest increasing subsequence

Number of paths in a dag

Shortest path in a dag

Dynprog over intervals

Dynprog over subsets

Dynprog over probabilities

Dynprog over trees

3^n set cover

Divide and conquer

Knuth optimization

Convex hull optimizations

RMQ (sparse table a.k.a 2^k -jumps)

Bitonic cycle

Log partitioning (loop over most restricted)

Combinatorics

Computation of binomial coefficients

Pigeon-hole principle

Inclusion/exclusion

Catalan number

Pick's theorem

Number theory

Integer parts

Divisibility

Euclidean algorithm

Modular arithmetic

* Modular multiplication

* Modular inverses

* Modular exponentiation by squaring

Chinese remainder theorem

Fermat's little theorem

Euler's theorem

Phi function

Frobenius number

Quadratic reciprocity

Pollard-Rho

Miller-Rabin

Hensel lifting

Vieta root jumping

Game theory

Combinatorial games

Game trees

Mini-max

Nim

Games on graphs

Games on graphs with loops

Grundy numbers

Bipartite games without repetition

General games without repetition

Alpha-beta pruning

Probability theory

Optimization

Binary search

Ternary search

Unimodality and convex functions

Binary search on derivative

Numerical methods

Numeric integration

Newton's method

Root-finding with binary/ternary search

Golden section search

Matrices

Gaussian elimination

Exponentiation by squaring

Sorting

Radix sort

Geometry

Coordinates and vectors

* Cross product

* Scalar product

Convex hull

Polygon cut

Closest pair

Coordinate-compression

Quadtrees

KD-trees

All segment-segment intersection

Sweeping

Discretization (convert to events and sweep)

Angle sweeping

Line sweeping

Discrete second derivatives

Strings

Longest common substring

Palindrome subsequences

Knuth-Morris-Pratt

Tries

Rolling polynomial hashes

Suffix array

Suffix tree

Aho-Corasick

Manacher's algorithm

Letter position lists

Combinatorial search

Meet in the middle

Brute-force with pruning

Best-first (A*)

Bidirectional search

Iterative deepening DFS / A*

Data structures

LCA (2^k -jumps in trees in general)

Pull/push-technique on trees

Heavy-light decomposition

Centroid decomposition

Lazy propagation

Self-balancing trees

Convex hull trick (wcipeg.com/wiki/Convex_hull_trick)

Monotone queues / monotone stacks / sliding queues

Sliding queue using 2 stacks

Persistent segment tree

troubleshooting.txt

52 lines

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?

Make sure to submit the right file.

Wrong answer:

Print your solution! Print debug output, as well.

Are you clearing all data structures between test cases?

Can your algorithm handle the whole range of input?

Read the full problem statement again.

Do you handle all corner cases correctly?

Have you understood the problem correctly?

Any uninitialized variables?

Any overflows?

Confusing N and M, i and j, etc.?

Are you sure your algorithm works?

What special cases have you not thought of?

Are you sure the STL functions you use work as you think?

Add some assertions, maybe resubmit.

Create some testcases 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.

Is your output format correct? (including whitespace)

Rewrite your solution from the start or let a teammate do it.

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 is the complexity of your algorithm?
Are you copying a lot of unnecessary data? (References)
How big is the input and output? (consider scanf)
Avoid vector, map. (use arrays/unordered_map)
What do your teammates think about your algorithm?

Memory limit exceeded:
What is the max amount of memory your algorithm should need?
Are you clearing all data structures between test cases?