1 Setup

1.1 header.h

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

#define ll long long
#define pl pair<ll, ll>
#define pi pair<int, int>
#define vl vector<ll>
#define vi vector<int>
#define vpl vector<pl>
#define vpi vector<pi>
#define in(el, cont) (cont.find(el) != cont.end())

constexpr int INF  = 2000000010;
constexpr ll LLINF = 9000000000000000010LL;

template <typename T, template <typename ELEM, typename ALLOC = std::
    allocator<ELEM> > class Container>
std::ostream& operator<<(std::ostream& o, const Container<T>& container) {
  typename Container<T>::const_iterator beg = container.begin();
    o << *beg++;
  while (beg != container.end()) {
    o << " " << *beg++;
  }
  return o;
}

// int main() {
//  ios::sync_with_stdio(false);  // do not use cout + printf
//  cin.tie(NULL);
//     cout << fixed << setprecision(12);
//  return 0;
// }
```

1.2 Bash for c++ compile with header.h

```
#!/bin/bash
if [ $# -ne 1 ];then echo "Usage: $0 <input_file>"; exit 1;fi
f="$1";d=code/;o=a.out
[ -f $d/$f ] || { echo "Input file not found: $f"; exit 1; }
g++ -I$d $d/$f -o $o && echo "Compilation successful. Executable '$o'
    created." || echo "Compilation failed."
```

1.3 Bash for run tests c++

```
g++ $1/$1.cpp -o $1/$1.out
for file in $1/*.in; do diff <($1/$1.out < "$file") "${file%.in}.ans"; done
```

1.4 Bash for run tests python

```
for file in $1/*.in; do diff <(python3 $1/$1.py < "$file") "${file%.in}.ans
    "; done
```

2 Python

2.1 Graphs

2.2 Dynamic Programming

2.3 Trees

2.4 Number Theory

2.5 Strings

2.6 Geometry

2.7 Combinatorics

2.8 Other Data Structures

2.9 Other Mathematics

3 C++

3.1 Graphs

3.1.1 BFS

```cpp
#include "header.h"
#define graph unordered_map<ll, unordered_set<ll>>
vi bfs(int n, graph& g, vi& roots) {
    vi parents(n+1, -1); // nodes are 1..n
    unordered_set<int> visited;
    queue<int> q;
    for (auto x: roots) {
        q.emplace(x);
        visited.insert(x);
    }
    while (not q.empty()) {
        int node = q.front();
        q.pop();

        for (auto neigh: g[node]) {
            if (not in(neigh, visited)) {
                parents[neigh] = node;
                q.emplace(neigh);
                visited.insert(neigh);
            }
        }
    }
    return parents;
}
vi reconstruct_path(vi parents, int start, int goal) {
    vi path;
    int curr = goal;
    while (curr != start) {
        path.push_back(curr);
        if (parents[curr] == -1) return vi(); // No path, empty vi
        curr = parents[curr];
    }
    path.push_back(start);
    reverse(path.begin(), path.end());
    return path;
}
```

3.2 Dynamic Programming

3.3 Trees

3.4 Number Theory

3.5 Strings

3.6 Geometry

3.7 Combinatorics

3.8 Other Data Structures

3.9 Other Mathematics