

1	Setup	1	2.8	Other Data Structures . . .	1
1.1	header.h . . . . .	1	2.9	Other Mathematics . . . . .	1
1.2	Bash for c++ compile with header.h . . . . .	1	3	C++	1
1.3	Bash for run tests c++ . .	1	3.1	Graphs . . . . .	1
1.4	Bash for run tests python .	1	3.1.1	BFS . . . . .	1
2	Python	1	3.2	Dynamic Programming . .	2
2.1	Graphs . . . . .	1	3.3	Trees . . . . .	2
2.2	Dynamic Programming . .	1	3.4	Number Theory . . . . .	2
2.3	Trees . . . . .	1	3.5	Strings . . . . .	2
2.4	Number Theory . . . . .	1	3.6	Geometry . . . . .	2
2.5	Strings . . . . .	1	3.7	Combinatorics . . . . .	2
2.6	Geometry . . . . .	1	3.8	Other Data Structures . . .	2
2.7	Combinatorics . . . . .	1	3.9	Other Mathematics . . . . .	2

## 1 Setup

### 1.1 header.h

```
1 #pragma once
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 #define ll long long
6 #define pl pair<ll, ll>
7 #define pi pair<int, int>
8 #define vl vector<ll>
9 #define vi vector<int>
10 #define vpl vector<pl>
11 #define vpi vector<pi>
12 #define in(e1, cont) (cont.find(e1) != cont.end())
13
14 constexpr int INF = 2000000010;
15 constexpr ll LLINF = 9000000000000000010LL;
16
17 template <typename T, template <typename ELEM, typename ALLOC = std::
    allocator<ELEM> > class Container>
18 std::ostream& operator<<(std::ostream& o, const Container<T>& container) {
19     typename Container<T>::const_iterator beg = container.begin();
20     o << *beg++;
21     while (beg != container.end()) {
22         o << " " << *beg++;
23     }
24     return o;
25 }
26
27 // int main() {
28 //     ios::sync_with_stdio(false); // do not use cout + printf
29 //     cin.tie(NULL);
30 //     cout << fixed << setprecision(12);
31 //     return 0;
32 // }
```

### 1.2 Bash for c++ compile with header.h

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

### 1.3 Bash for run tests c++

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

### 1.4 Bash for run tests python

```
1 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

---

```
1 #include "header.h"
2 #define graph unordered_map<ll, unordered_set<ll>>
3 vi bfs(int n, graph& g, vi& roots) {
4     vi parents(n+1, -1); // nodes are 1..n
5     unordered_set<int> visited;
6     queue<int> q;
7     for (auto x: roots) {
8         q.emplace(x);
9         visited.insert(x);
10    }
11    while (not q.empty()) {
12        int node = q.front();
13        q.pop();
14
15        for (auto neigh: g[node]) {
16            if (not in(neigh, visited)) {
17                parents[neigh] = node;
18                q.emplace(neigh);
19                visited.insert(neigh);
20            }
21        }
22    }
23    return parents;
24 }
25 vi reconstruct_path(vi parents, int start, int goal) {
26     vi path;
27     int curr = goal;
28     while (curr != start) {
29         path.push_back(curr);
30         if (parents[curr] == -1) return vi(); // No path, empty vi
31         curr = parents[curr];
32     }
33     path.push_back(start);
34     reverse(path.begin(), path.end());
35     return path;
36 }
```

---

## 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