COMPETITIVE PROGRAMMING

Why it helps you become a better software engineer

TASK

- 500K numbers, each up to 10K.
- Output the sum of these numbers, as quickly as you can.

500 000 * 10 000

5 BILLION

INTEGER *OVERFLOW*

COMPETITIVE PROGRAMMING

Why it helps you become a better software engineer

```
#include <bits/stdc++.h>
#define int long long
int a[1000005],b;
int main() {
  scanf(“%lld\n”, &b);
  FOR(i,b)printf(“%lld\n”,a[i]);
}
```

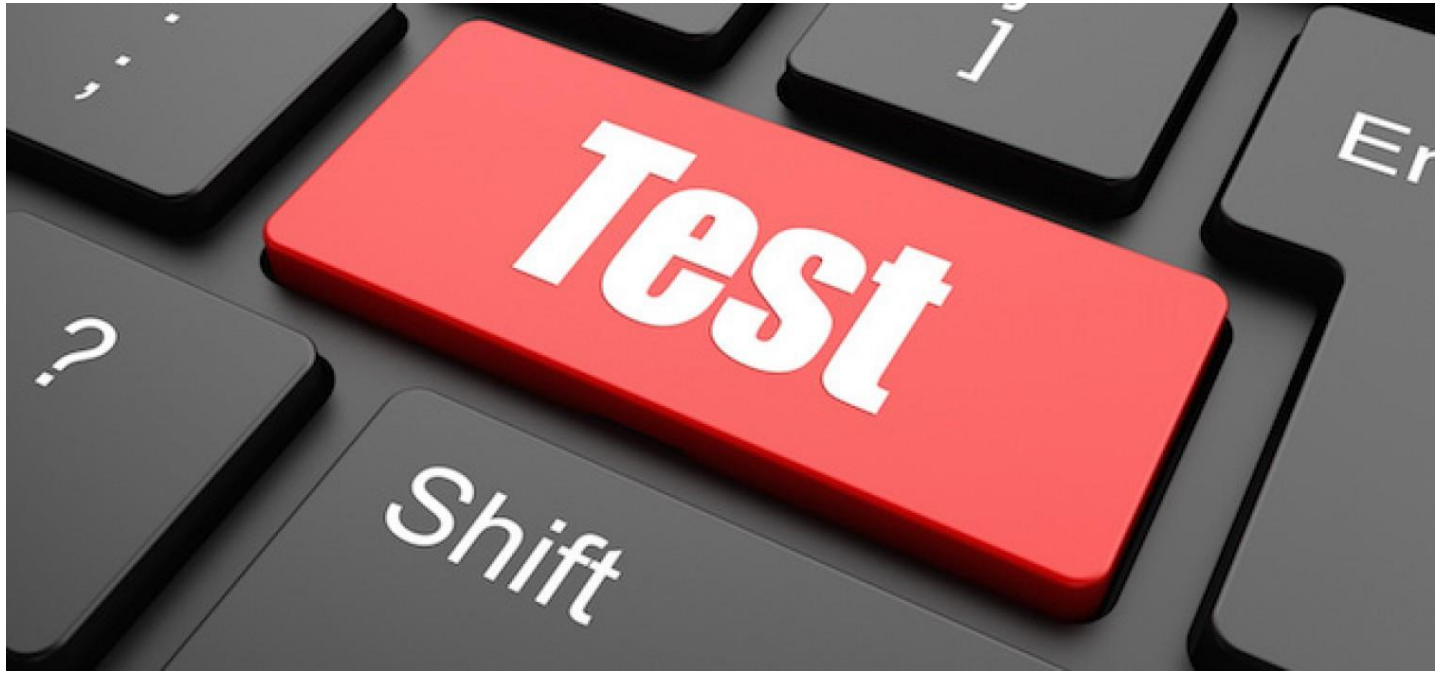
*The above is written by a current student of CS3281/82.

GOAL

- Become a better software engineer.
- Make more reliable products.

AGENDA

- How do we do better testing for our products?
- How do we engineer efficient solutions?
- How do we do better debugging?



HOW DO WE DO BETTER TESTING?

WHY IS GOOD TESTING IMPORTANT?

- The software is as good as its tests
- Ineffective tests
  - Waste of computing resources, time
  - False sense of security

CONSIDER ALL EDGE CASES

EDGE CASE CHECKLIST

- Negative answers
- Overflow
- Uniqueness
- Large cases
- “What if N > …”

```
#include <bits/stdc++.h>
#define int long long
int a[1000005],b;
int main() {
  scanf(“%lld\n”, &b);
  FOR(i,b)printf(“%lld\n”,a[i]);
}
```

*The above is written by a current student of CS3281/82.

HOW DO WE ENGINEER EFFICIENT SOLUTIONS?

WE DON’T

COMPLICATED, 100 LINE
$O(N \log \log N)$

EASY, 10 LINE
$O(N^2)$

COMPLICATED, 100 LINE
$O(N \log \log N)$

EASY, 10 LINE
$O(N^2)$

AS EFFICIENT AS NECESSARY

DATA STRUCTURES AND ALGORITHMS

- Using (Hash)maps instead of Arrays
- Optimising database queries
  - DB queries are usually faster than application-side queries.
- Generally implemented as libraries for you
  - C++: STD / Algorithms / Boost Library
  - Java: Apache Commons
  - NodeJS: just NPM it

DATA STRUCTURES AND ALGORITHMS

- Learning them is important, but cannot be taught in a 5-minute talk.
  - Just master common ones. No need for "Fenwick Tree" or "KMP String Matching algorithm"
  - Alternatively, find people in the ICPC lab.

HOW DO WE DEBUG BETTER?

YOU ARE THE DEBUGGER

DEBUGGING

- No luxury of debuggers

THINK ABOUT THE CODE

DEBUGGING

- No luxury of debuggers
  - Check only the "high risk" things
    - Memory/array access: check the index
    - Variables: check the value (does it make sense?)
- Assertions and print statements
  - **Confirm** our assumptions.

TAKEAWAYS

- How do we do better testing for our products?
- How do we engineer efficient solutions?
- How do we do better debugging?

DO COMPETITIVE PROGRAMMING

YOU'LL BE A BETTER SOFTWARE ENGINEER.