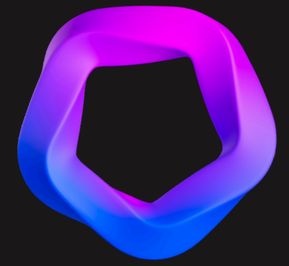
An Intro To

Competitive Programming



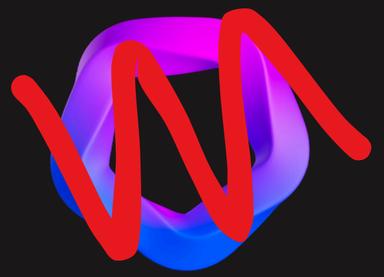
LUCIDS

Computing, Informatics & Data Society

Presented by:
Harshit Mittal
h.mittal@lancaster.ac.uk

An Intro To

Competitive Programming



LUCIDS

Computing, Informatics & Data Society

Presented by:
Harshit Mittal
h.mittal@lancaster.ac.uk

An Intro To

Competitive Programming



LUCID

CompSoc

Computing, Informatics & Data Society

Presented by:
Harshit Mittal
h.mittal@lancaster.ac.uk

What...?

Competitive Programming?



> The point of the competition is to see who can write BETTER code

> BETTER code runs faster, and/or uses less memory

> Your code is judged by.. well.. a judge (an automated third party program)



Why Should I Care?

- Improve problem solving and programming skills
- Learning useful algorithms in a practical way
- Great practice for technical interviews
- Thinking and problem solving is fun! and competitive!



Where Do I Start?

There are some very useful and popular resources to get started. There are problems for everyone, whether a complete beginner or a hardened veteran.

- LeetCode (leetcode.com/problemset/)
- Codeforces (codeforces.com/problemset)
- CodeChef (www.codechef.com/practice)
- CP-Algorithms (cp-algorithms.com)
- Competitive Programmer’s Handbook (cses.fi/book)



Lets Try Something...

You are given an array of integers. You will be given pairs of indexes, say i and j. For each pair you must return the sum of all the numbers in the index interval [i,j] from the array.

Try to think of the logic, code is not relevant for this part of the talk

Feel free to ask for clarifications

How Well Did We Do?



How do we know how good our solution was?
We cant really count up the bytes of memory used, or the milliseconds or processor time...

So what metric do we use?

- Time complexity
- Memory complexity

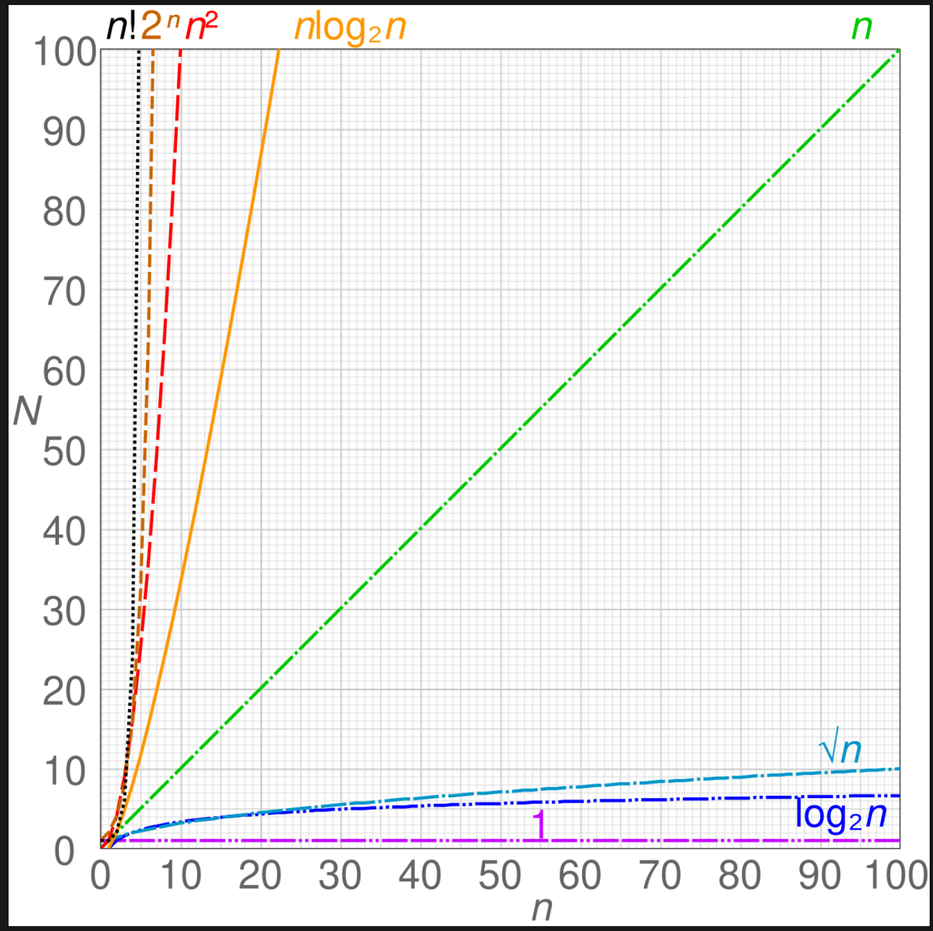
Time Complexity



- Time complexity describes how the running time of an algorithm grows as the input size n increases. We usually express it using Big-O notation.
- We ignore all constants.
- We always assume the worst case input whilst calculating time complexity.

$O(n!) > O(2^n) > O(n^2) > O(n * \log n) > O(n) > O(\sqrt{n}) > O(\log n) > O(1)$

*logN always refers to $\log_2 N$ unless specified otherwise



Example of O(n)



- Runs n times
- Does constant operation every time

```
n = ...

for i in range(n):
    print(i)

print("done!")
```

Total operations: n*c = nc

Total time complexity: O(n)

(constant ignored!)

Example of O(n)



- Runs n times
- Runs n more times

Total operations: nc + nc = 2nc

Total time complexity: O(n)

(constants ignored!)

```
n = ...

for i in range(n):
    print(i)
for j in range(n):
    print(j)

print("done!")
```

Example of $O(n^2)$



- Runs n times
- Does n operations every time
- Does c operations n*n times

Total operations : n*n*c

Time complexity: $O(n^2)$

(constant ignored!)

```
n = ...

for i in range(n):
    for j in range(n):
        print(i, j)

print("done!")
```



How Well Did We Do?

Part 2: Electric Boogaloo

(The speaker will now explain how good our solution was)



Quick Intro To Leetcode

(The speaker will now switch windows)

Please open Leetcode and go to problem #204



Try to come up with a solution all by yourself, you are allowed to use C/C++/Java/Python3 for writing the code.

Try your best to first lay out a logical solution, and only then start writing code.

Generative AI or peeping at the solutions tab is not allowed.
You can look at documentation for your respective language.
Feel free to ask the speaker for clarifications or help



Discussion Time!

Well, This is it.



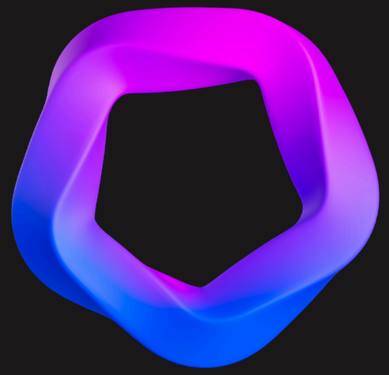
This is it! This was your intro to Competitive Programming.

Now go get started on your own with the aforementioned resources, and remember, the only way to improve is to practice.

THANK YOU FOR YOUR TIME

(The speaker will now attempt to do a smooth outro)

THANK YOU FOR YOUR TIME



LUCIDS

Computing, **I**nformatics & **D**ata **S**ociety

WhatsApp

Discord

Presented by:
Harshit Mittal
h.mittal@lancaster.ac.uk