UQCS Competitive Programming Group Week 3

UQ Computing Society

Hosted by Amy Hu and Roxie Cunningham

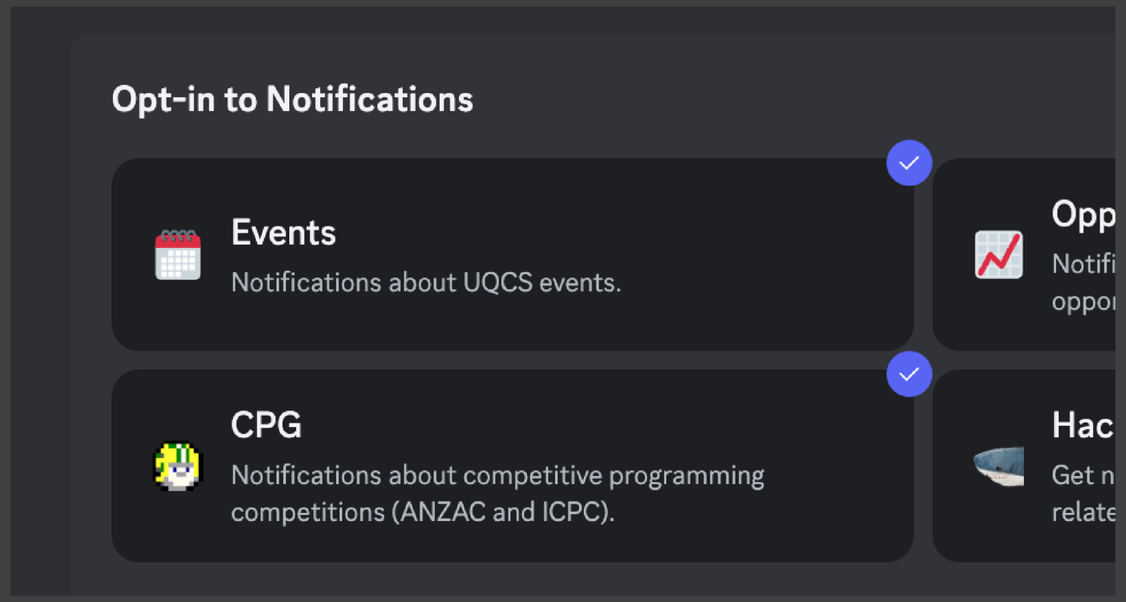
UQ Computing Society

We shall start at 4:15, before that, feel free to talk to people around you and try form a team of 3-5 people, we are getting competitive later!

Also come up with a team name, I will collect them later

Useful Resources

- GitHub - https://github.com/UQComputingSociety/cpg
- Past Year Questions https://prog4fun.csse.canterbury.ac.nz/course/view.php?id=2
- LeetCode
- Go sign up for the @cpg role in our discord channel!
- Related courses - eg. COMP3506 - Data Structure and Algorithms



Quick explanation of Big-O complexity (For the first years)

- if you’re already familiar with it, great! Take a nap, chat a little, or do some exercise on prog4fun, this won’t take long. (hopefully)

So, what is Big-O complexity?

Quick explanation of Big-O complexity (For the first years)

- if you’re already familiar with it, great! Take a nap, chat a little, or do some exercise on prog4fun, this won’t take long. (hopefully)

So, what is Big-O complexity?

- Well, It’s about time -

Quick explanation of Big-O complexity (For the first years)

- if you're already familiar with it, great! Take a nap, chat a little, or do some exercise on prog4fun, this won't take long. (hopefully)

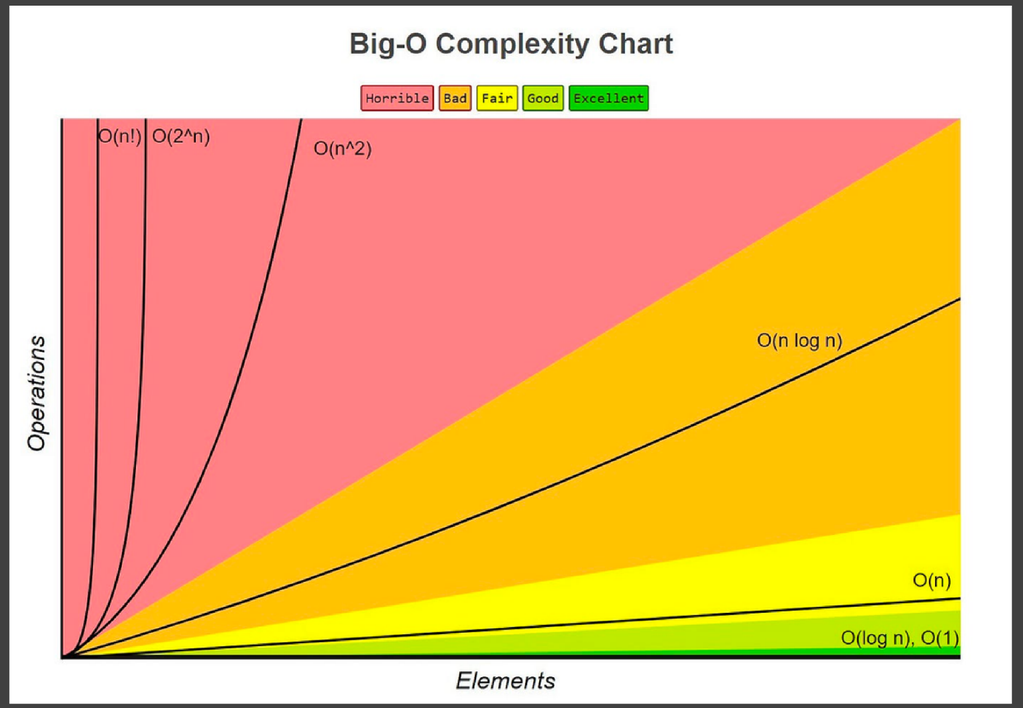
So, what is Big-O complexity?

- Well, It's about time - and space!
- It is what we use to describe how much time/memory is required for our program to run based on the size of our input
- Big-O complexity is about finding the asymptopic upper bound for an algorithm.

There are also big-θ complexity and big-Ω complexity but we're not talking about them here - this is, after all, not an algorithm class.

Quick explanation of Big-O complexity (For the first years)

- Here are some of the most common levels of big-O complexity:
  - O(1), aka constant time
  - O(n)
  - O(n log n)
  - $O(n^2)$
  - $O(2^n)$
  - O(n!)
- Selection sort (from last week's example) have a time complexity of $O(n^2)$, while merge sort have time complexity of O(nlog(n))



Didn’t Understand? Don’t worry! Here are some articles about complexity

- Khan Academy - Asymptotic Notation - Big-O notation
- GeeksforGeeks - Worst, Average and Best Case Analysis of Algorithms
- https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt-1674cfa8a23c/
- https://towardsdatascience.com/understanding-time-complexity-with-python-examples-2bda6e8158a7

And I assure you while these articles are about complexity, they are not complex at all ;)

This week’s topic: Common Data Structures

In cpg, we often use a lot of different data structures to help us solve problems. Today we are going to focus on three of the most basic and common ones, which are:

- Stack
- Queue
- Priority Queue

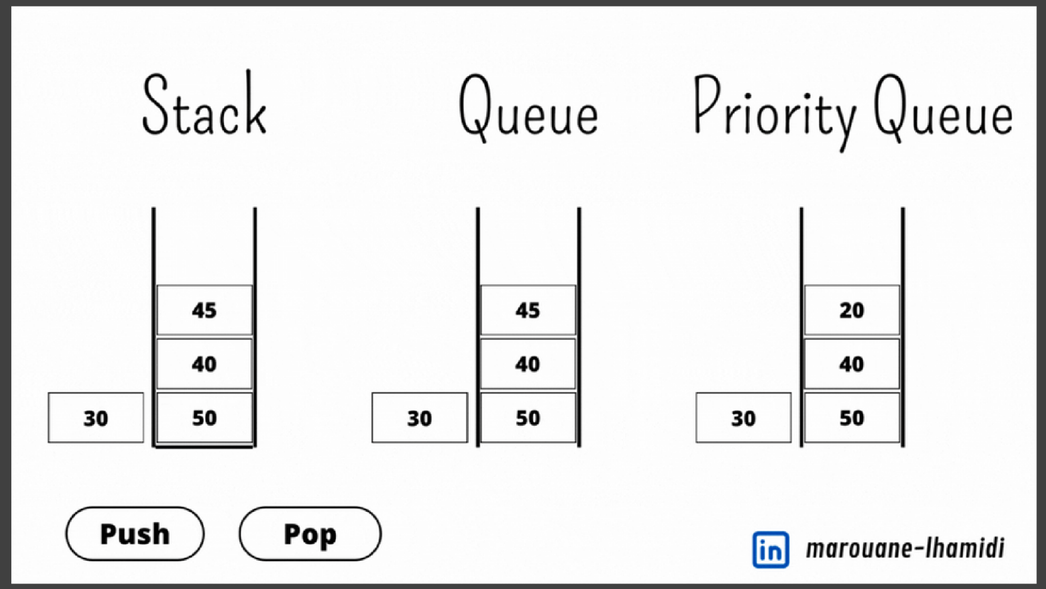


Image stolen from https://medium.com/@marouane.lhamidi/data-structures-and-algorithms-cf46913d5b18, they have a nice animation, go check it out if you want

DEMO WITH ✨CARDS✨

Water Break

THEN THERE WILL BE SOME PROBLEM SETS WITH AMY'S UGLY CODES

Finally, The Problem Sets! - This one is about stack

Samir String

Samir got a string from his friend and string contains only two characters *'I'* and *'D'*, *'I'* represent the Increasing sequence and *'D'* represent the Decreasing sequence. His friend asks to make the minimum number without repetition of the numbers such that it follows the sequence given in the string. Samir is solving this problem from the last 33 days. You can help him to solve this problem.

Expected Time Complexity O(n)

Note: String contain only Uppercase Letter *'I'* and *'D'*.

Input Format

The first line of input is an integer *T* denoting the number of the test case. Then *T* test cases follow. Each test case contains a string.

Output Format

For each test case print output in the new line. The minimum number without repetition of numbers.

Constraints

1 <= T <= 10

$1<=|S|<=2*10^4$, |S| denote the length of the given string.

Time Limit

1 second

Example

Input

1

IDIDI

Output

1 3 2 5 4 6

Yoinked from: https://mycode.prepbytes.com/problems/stacks/MINNUMSTR

We didn’t get to do this, you can try them at home!

And the next one is about Queue AND ALSO Stack

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.
The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack. At each step:

- If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue.
- Otherwise, they will leave it and go to the queue's end.

This continues until none of the queue students want to take the top sandwich and are thus unable to eat.
You are given two integer arrays students and sandwiches where sandwiches[i] is the type of the i th sandwich in the stack (i = 0 is the top of the stack) and students[j] is the preference of the jth student in the initial queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

https://leetcode.com/problems/number-of-students-unable-to-eat-lunch/description/