Quiz 03 Review Session

COMP 210 / 2024 Summer Session I

Ajay Gandecha

Quiz 03 Format

- 30 minutes at the start of class.
- Should be shorter :)
- *On paper* - bring a pencil!
- **Question Types:**
  - Multiple choice, T/F, select all that apply, fill in the blank, diagramming (?)
  - *No code writing on this quiz - but be able to trace given Java code!*

Exercise Check-In Question

- Similar format to the exercise question on Quiz 02.
- Review **`Ex05` - Linked List pt. 1**. → isSymmetrical
  → reverse
  → multiply
  → isEqual
- *Questions?*

On Quiz 03

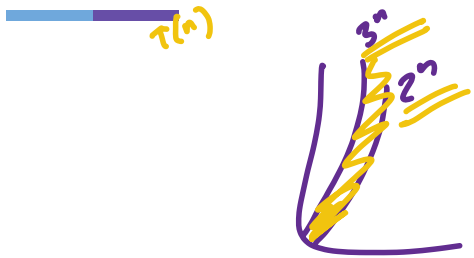
- Stacks (LIFO data structure)
- Queues (FIFO data structure)
- Basic Sorting, Big-O Analysis

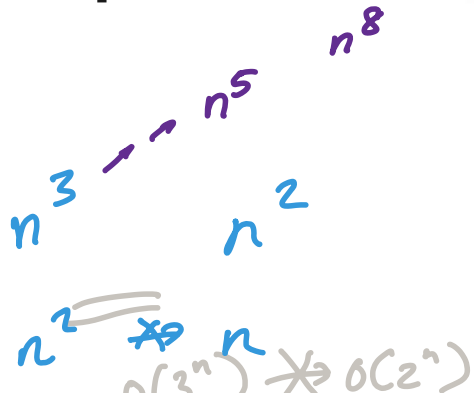
Review: Big-O Analysis

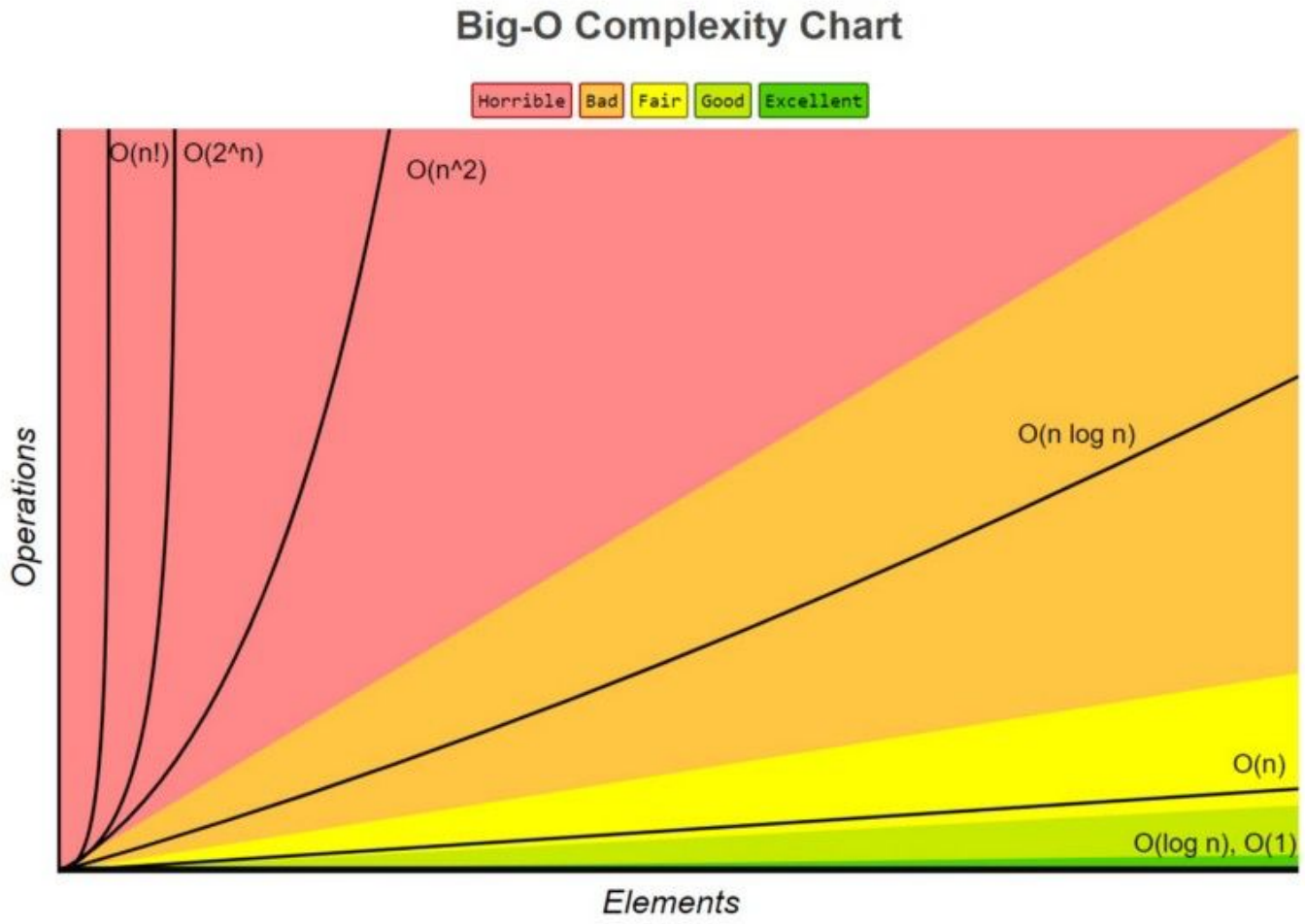
- **We need a way to determine *how efficiently* algorithms run.**
  - We need notation to be able to compare the *efficiency* of algorithms.
  - This is called Big-O Notation.

- **We can tell how efficient algorithms run by comparing *how many operations* an algorithm performs compared to the *number of inputs we supply to it*.**

Big-O Graph Comparisons

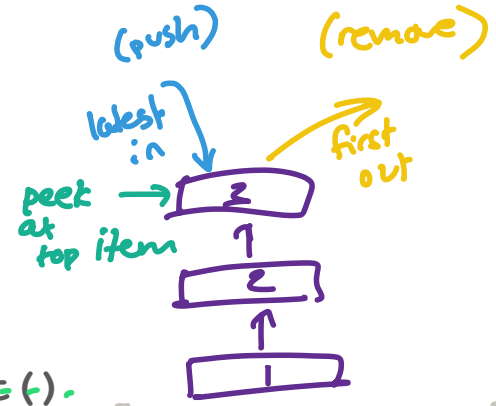


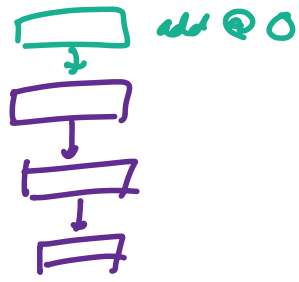




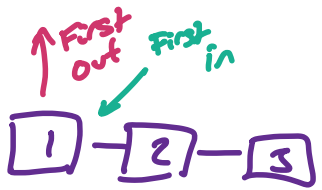
Stack Data Structure

- LIFO (last in, first out)
- Operations: `.push()`, `.pop()`, ~~`.get()`~~ `.peek()`





Queue Data Structure



- FIFO (first in, first out)
- Operations: `.enqueue()`, `.dequeue()`, `.get()`

=add =remove peek

	add(end)	remove(0)	get(0)
LL: head+tail	O(1)	O(1)	O(1)
≠			
LL: head only (no tail)	O(N)	O(1)	O(1)
AL:	O(N)	O(N)	O(1)

Sorting

- We can create **algorithms** to convert an unsorted list into a sorted one.
- Many different approaches… (some not covered in class):
  - Quick sort
  - Merge sort
  - Bubble sort
  - *… and more!*
- We can compare these different methods by comparing their **time complexity**.

Sorting

- `Comparable<T>`
  - Has method: `a.compareTo(b)`
    - *= 1 if a>b*
    - *= 0 if a == b*
    - *= -1 if a < b*
  - *TLDR: Helps us **sort items** (to sort, we need to compare items to each other).*

Big-O Sorting Example - LinkedList<T>

Selection Sort

← LL!

● = if done with an ArrayList instead.

```
List<T> sort(List<T> list) {

  for(int i = 0; i < list.size() - 1; i++) {

    int minIndex = i;

    for(int j = i; j < list.size(); j++) {

      if(list.get(j).compareTo(list.get(minIndex)) == -1) minIndex = j;

    }

    list.swapAt(i, minIndex); // O(?)

  }

  return list;

}
```

O(N)

O(1)

O(N)

O(N) O(1) O(N) O(1) O(1) O(N)

*O(1) *AL: O(1) *AL: O(1) *AL: O(1)

O(N) *AL: O(1)

AL: O(1)

LL: O(N)

$$n\left[1 + n\left[O(N)\right] + n\right]$$

$\sum_{i=0}^{n}$ n times

AL:

$n[1+n[1]+1]$

$n[n+2] \rightarrow O(n^2)$

w/ ArrayList

$n[1+n^2+n]$

$n+n^3+n^2 \Rightarrow O(n^3)$

w/ LinkedList