Quiz 02 Review Session

COMP 210 / 2024 Summer Session I

Ajay Gandecha

Quiz 02 Format

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

Exercise Check-In Question

- ...

On Quiz 02

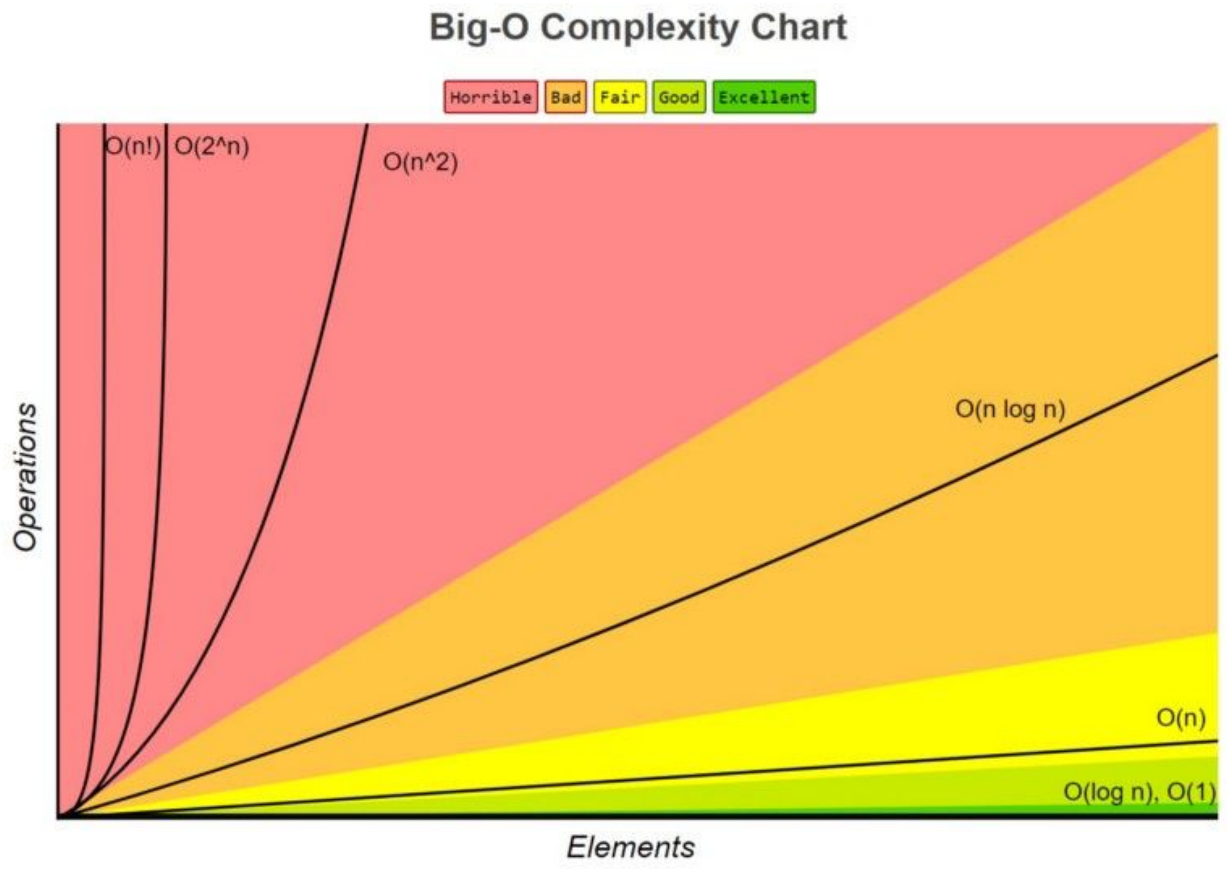
- Big-O Analysis
  - Analyzing code snippets for runtime analysis, including recursive code
- The `List` Abstract Data Type
  - Understand `ArrayList` and `LinkedList` on the heap
  - Explain trade-offs between both, justified using big-O notation

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



Recursive Example 1

```
void foo(int n) {
    if(n<=0) return 1;
    return 1 + foo(n-1);
}
```

Recursive Example 2

```
void fib(int n) {

    if(n<2) return n;

    return fib(n-1) + fib(n-2);

}
```

Recursive Example 3

```
void fib(int n) {
    if(n<=1) return n;
    return fib(n/2) + fib(n/2);
}
```

`ArrayList` Representation

- Recall that `List` is an abstract data type.
- **`ArrayList`** is one implementation of the `List` interface.

`ArrayList` Representation

`LinkedList` Representation

- **`LinkedList`** is another implementation of the `List` interface.

`LinkedList` Representation

Deriving `List` Time Complexities

	get(0)	get(i)	get(n)	insert(0)	insert(i)	insert(n)	remove(0)	remove(i)
ArrayList								
LinkedList (Head only)								
LinkedList (Head and Tail)								