LEC 07

CSE 123

Runtime Analysis

BEFORE WE START

Talk to your neighbors:

What was the last book you read?

Music: 123 24su Lecture Tunes

Instructor: Joe Spaniac

TAs: Andras, Daniel, Eric, Nicole, Sahej, Trien, Zach

Questions during Class?

Raise hand or send here

sli.do #cse123

Lecture Outline

- **Announcements**
- Finishing up LinkedIntList
- Runtime Analysis
  - Complexity Classes
  - Big-Oh Notation
- Analyzing List Implementations

Announcements

- Quiz 1 grades out!
  - Please check your grades before the next quiz, practice **metacognition**
- Quiz 2 in section on Tuesday, 7/16
  - Topics: Abstract classes, ArrayIntList, LinkedIntList
  - Same policies as last time: One sheet of 8.5” x 11” notes (double-sided, printed or handwritten), 50mins, etc.
- Programming assignment 2 released last night, due in 2 weeks
- Resubmission Period 2 closes tonight, 7/12 at 11:59pm
- Resubmission Period 3 opens tonight, due next Friday 7/19 at 11:59pm

Lecture Outline

- Announcements
- **Finishing up LinkedIntList**
- Runtime Analysis
  - Complexity Classes
  - Big-Oh Notation
- Analyzing List Implementations

Lecture Outline

- Announcements
- Finishing up LinkedIntList
- **Runtime Analysis**
    - Complexity Classes
    - Big-Oh Notation
- Analyzing List Implementations

Runtime Analysis

- What’s the “best” way to write code?
  - Depends on how you define best: Code quality, memory usage, speed, etc.

- Runtime = most popular way of analyzing solutions
  - Slow code = bad for business
- How do we figure out how long execution takes?
  - Stopwatch = human error
  - Computers = computer error (artifacts, operating systems, language)
  - Need a way to formalize abstractly…

Runtime Analysis

- We'll count simple operations as 1 unit
  - variable initialize / update `int x = 0;`
  - array accessing `arr[0] = 10;`
  - conditional checks `if (x < 10) {`

- Goal: determine how the number of operations scales w/ input size
  - Don't care about the difference between 2 and 4
  - Find the appropriate **complexity class**

- Result: evaluation tactic independent of OS, language, compiler, etc.
  - Simple operation = constant regardless of if it is truly 1

Complexity Classes

- Input will always be an array `arr` of length *n*
- <u>Constant (1)</u>
  - # Ops doesn't relate to *n*

```
return arr[0];
```

- <u>Linear (n)</u>
  - # Ops proportional to *n*

```
for (int i = 0; i < arr.length; i++)
```

- <u>Quadradic (n^2)</u>
  - # Ops proportional to *n^2*

```
for (int j = 0; j < arr.length; j++)
    for (int j = 0; j < arr.length; j++)
```

- Lets say # Ops = n^2 + 100000n
  - If n was really, really, really big, which term matters more?
  - Only care about the **dominating term** for complexity!

Complexity Classes

What’s the complexity class of the following?

```
      public static void mystery(int[] arr) {
        1   if (arr.length == 0) {
   2                throw new IllegalArgumentException();
            }
        1   return arr[arr.length – 1];
      }
```

Constant Complexity (1)

Complexity Classes

What's the complexity class of the following?

```
             public static int mystery(int[] arr) {
          1      int sum = 0;
                 for (int i = 0; i < arr.length; i++) {
3n + 2   3n        3   sum += arr[i];
                 }
          1      return sum;
             }
```

Linear Complexity (n)

Complexity Classes

What's the complexity class of the following?

```
public static int mystery(int[] arr) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = 0; j < arr.length; j++) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}
```

2

2n

1

n(2n + 1)
= 2n^2 + n

Quadratic Complexity (n^2)

Big-Oh Notation

- Programmers… are pessimists (or maybe realists)
  - Case in point: dominating term
- In the real world, best-case complexity isn't super useful
  - Want to make sure solutions work well in the worst possible situations

- We use Big-Oh notation to demonstrate worst-case complexity!

```
public static int indexOf(int[] arr, int x) {
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == x) return i;
    }
    return -1;
}
```

Worst-case linear

<u>O(n)</u>

Lecture Outline

- Announcements
- Finishing up LinkedIntList
- Runtime Analysis
  - Complexity Classes
  - Big-Oh Notation
- **Analyzing List Implementations**