

# CS 113 – Computer Science I

## Lecture 25 – Runtime

Tuesday 12/12/2023

# Announcements

## HW10:

- Due Thursday 12/12 (shorter side)
- Building 2 Fancy classes that keeps track of Strings
  - SuperDuperArray
  - SuperDuperSortedArray
- fully autograded

# Announcements

## Final

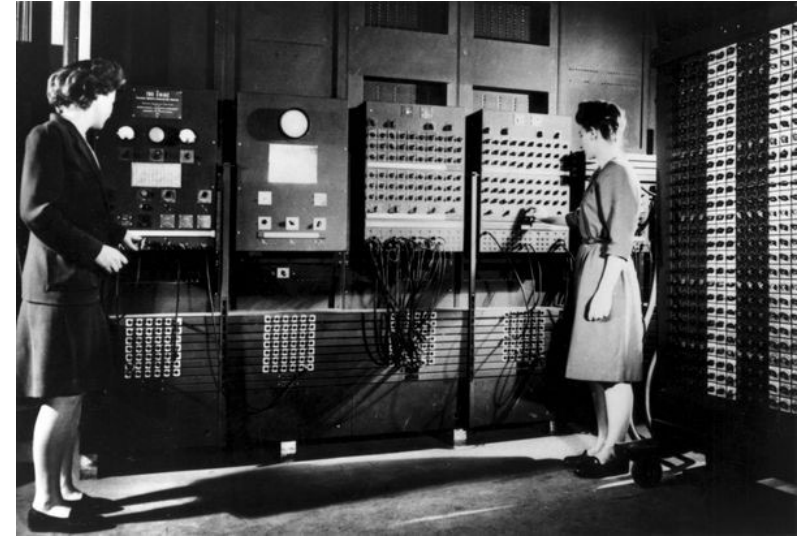
- Self-scheduled
- Closed-notes and books
- Practice exam will be released this week
- Cumulative
- A bit longer than midterms (but not 2x)

## Thursdays class:

- Review
- AMA
- On Zoom

# Measuring performance

How do we quantify performance?



# Computing the speed of your programs

Compute the time needed to execute a function

```
import java.lang.System.*;

public static void main(String[] args) {
    ....

    double start = System.currentTimeMillis()/1000.0; // converts to seconds
    bubbleSort(L);
    double end = System.currentTimeMillis()/1000.0;

    System.out.printf("Time: %.10f", (end-start));
```

# Runtime analysis: Big-O notation

Quantifies worse-case performance

theoretical measure of how performance changes with input size

Advantages

Hardware-independent measure

Allows us to analyze different approaches without implementing the algorithm first

# Big-O counting

- defining, assigning variables (1 step)
- printing, reading input (built-in function calls: “k” steps)
- mul, divide, sub, add, mod, etc (1 step)
- testing conditions (1 step)

# Big-O Example – Compute a sum

```
int sum = 0;
int i = 0;
while (i < n) {
    sum = sum + i;
    i++;
}
System.out.println(sum);
```



# Runtime practice

a)

```
int n = getInputSize();  
for (int i = 0; i < n; i++) {  
    System.out.println(i);  
}
```

{

# Runtime practice

b)

```
int n = getInputSize();  
for (int i = 0; i < 100; i++) {  
    System.out.println(i*n);  
}
```

{

# Runtime practice

c)

```
int n = getInputSize();  
for (int i = 0; i < n; i++) {  
    System.out.println(i);  
}
```

```
for (int j = 0; j < n; j++) {  
    System.out. println(j);  
}
```

# Runtime practice

d)

```
int n = getInputSize();  
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < n; j++) {  
        System.out.println(i, j);  
    }  
}
```

# Runtime practice

e)

```
int n = getInputSize();  
for (int i = 0; i < n; i++) {  
    for (int j = i; j < n; j++) {  
        System.out.println(i, j);  
    }  
}
```

# Runtime practice

h)

```
int[] lst = {1,2,3,5,7,12,19,34,55,67,99,101};
```

```
int n = lst.length;
```

```
int mid = floor(n/2);
```

```
System.out.println(lst[mid]);
```

# Runtime practice

```
i)
int n = getInputSize();
for (int i = 0; i < n; i++) {
    k = n;
    while (k > 1) {
        System.out.println(i, k);
        k = k/2;
    }
}
```

# Runtime practice

```
f)
int n = getInputSize();
for (int i = 0; i < n; i++) {
    for (int j = 0; j < 10; j++) {
        println(i, j);
    }
}
```



# Runtime practice

```
g)
int n = getInputSize();
while (n > 1) {
    println(n);
    n = n/2;
}
```

# Linear Search - revisited

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

# Binary Search – what is the runtime?

```
public static int search(int x, int[] L) {  
    int low = 0;  
    int high = L.length-1;  
    while (low <= high) {  
        int mid = (low + high)/2;  
        if (x > L[mid]) {  
            low = mid+1;  
        }  
        else if (x < L[mid]) {  
            high = mid-1;  
        }  
        else {  
            return mid;  
        }  
    }  
    return -1;  
}
```

# BubbleSort Revisited – What is the runtime?

```
public static void bubbleSort(int[] L) {  
  
    for (int n = 0; n < L.length; n++) {  
        for (int j = 1; j < L.length-n; j++) {  
            if (L[j-1] > L[j]) {  
                swap(j-1, j, L);  
            }  
        }  
    }  
}
```

# Comparisons in BubbleSort

How many comparisons do we need to make to sort a list of size  $n$ ?

$$\frac{n(n-1)}{2}$$

Why? (draw on the board)

$$\frac{n(n-1)}{2} = \frac{n^2 - n}{2}$$

So what's the Big-Oh?

$$O(n^2)$$

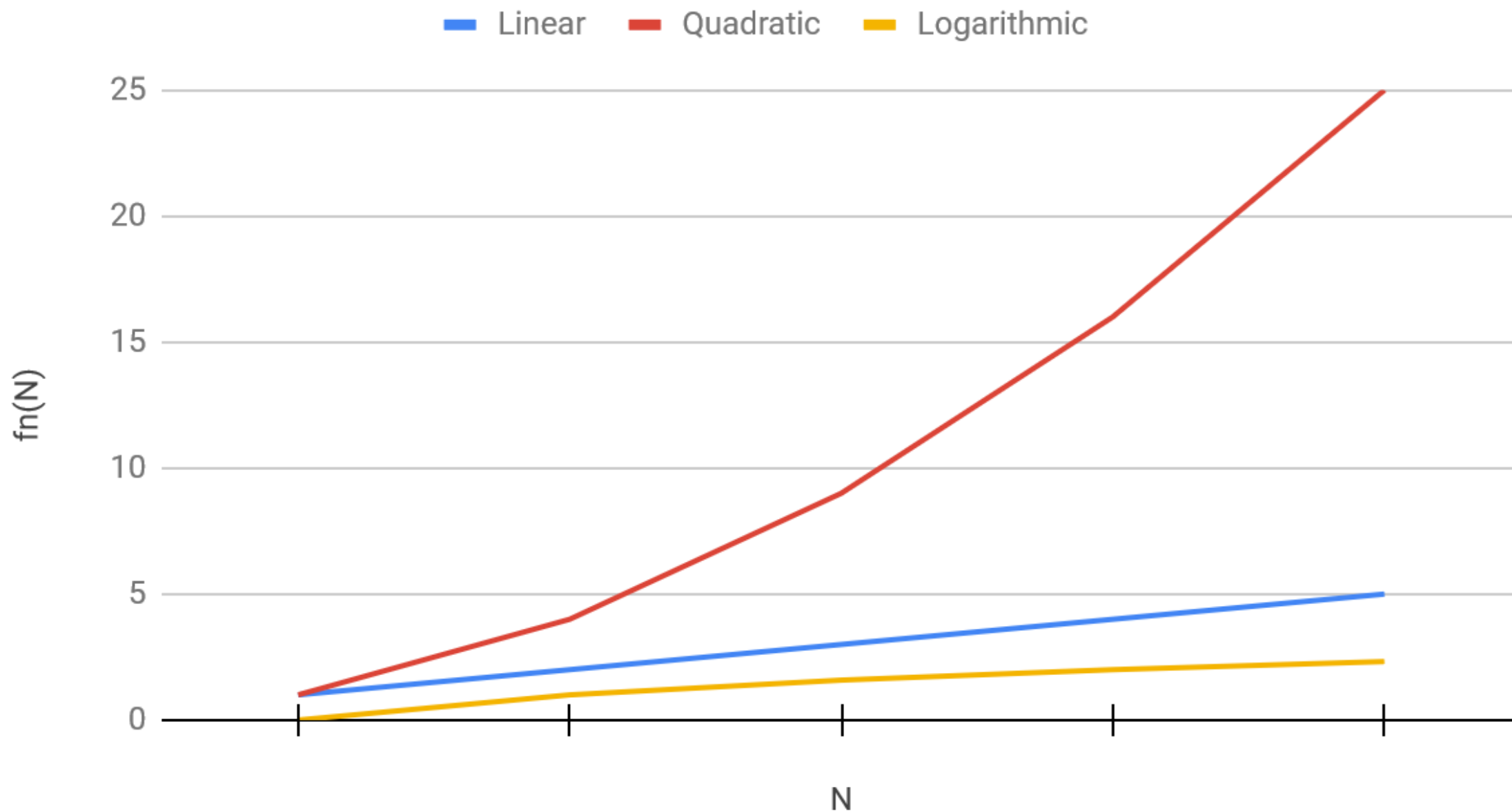
# BubbleSort Revisited – What is the runtime?

```
public static void bubbleSort(int[] L) {  
  
    for (int n = 0; n < L.length; n++) {  
        for (int j = 1; j < L.length-n; j++) {  
            if (L[j-1] > L[j]) {  
                swap(j-1, j, L);  
            }  
        }  
    }  
}
```

# SelectionSort Revisited – What is the runtime?

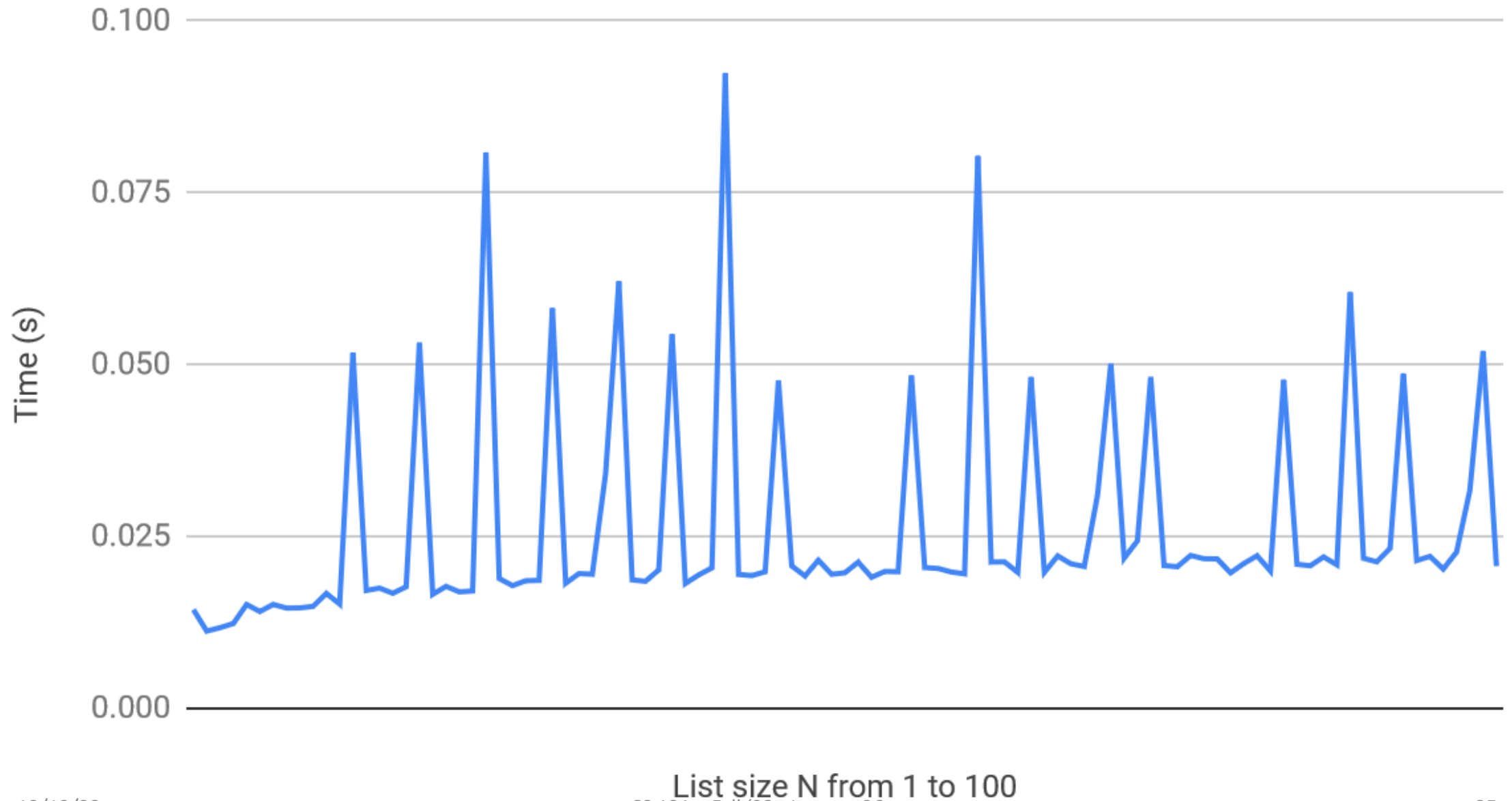
```
public static void selectionSort(int[] L) {  
  
    for (int i = 0; i < L.length; i++) {  
        int minIdx = i;  
        for (int j = i+1; j < L.length; j++) {  
            if (L[j] < L[minIdx]) {  
                minIdx = j;  
            }  
        }  
        swap(i, minIdx, L);  
    }  
}
```

# Comparison of runtimes





# Binary search



# Linear Search

