CSE 250
Data Structures

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Lec 07: Analyzing Code

Announcements

- PA1 Testing due Sunday

Recap from Last Class

f and ***g*** are in the same complexity class, denoted $g(n) \in \Theta(f(n))$, iff:

$g(n) \in O(f(n))$: Exists $n_0 \geq 0$, $c > 0$ s.t. for all $n \geq n_0$, $g(n) \leq c \cdot f(n)$

and

$g(n) \in \Omega(f(n))$: Exists $n_0 \geq 0$, $c > 0$ s.t. for all $n \geq n_0$, $g(n) \geq c \cdot f(n)$

In Practice

Most documentation uses Big-O (upper, ‘worst-case’) bounds…

- There’s always a Big-O bound
- The best case usually doesn’t bring down production servers

Quick Demo

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &= \theta(g(N) \cdot f(N)) \quad \text{(if } \theta(g(N)) \text{ exists)} \\
\theta(g(N)) + \theta(f(N)) &= \theta(g(N) + f(N)) \\
&= \text{The greater of } \theta(f(N)) \textbf{ or } \theta(g(N))
\end{aligned}
$$

Algebra with Big-Θ

Example

```
1  public void countDuplicates(Data[] data) {
2    System.out.println("Counting duplicates");
3    int count = 0;
4    for (int i = 0; i < data.length; i++) {
5      for (int j = i+1; j < data.length; j++) {
6        if (data[i] == data[j]) {
7          count++;
8        }
9      }
10   }
11 }
```

Example

```
1  public void countDuplicates(Data[] data) {
2    System.out.println("Counting duplicates");
3    int count = 0;
4    for (int i = 0; i < data.length; i++) {
5      for (int j = i+1; j < data.length; j++) {
6        if (data[i] == data[j]) {
7          count++;
8        }
9      }
10   }
11 }
```

1 step ∈ Θ(1)

1 step ∈ Θ(1)

Example

```
1  public void countDuplicates(Data[] data) {
2    Θ(1)
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        Θ(1)
6      }
7    }
8  }
```

Example

```
1  public void countDuplicates(Data[] data) {
2    Θ(1)
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        Θ(1)
6      }
7    }
8  }
```

$$\sum_{j=i+1}^{n} 1 = (n - i + 2)$$

Example

```
1  public void countDuplicates(Data[] data) {
2    Θ(1)
3    for (int i = 0; i < data.length; i++) {
4      Θ(n)
5    }
6  }
```

Example

```
1  public void countDuplicates(Data[] data) {
2    Θ(1)
3    for (int i = 0; i < data.length; i++) {
4      Θ(n)
5    }
6  }
```

$$\sum_{i=1}^{n} n = n^2$$

Example

```
1  public void countDuplicates(Data[] data) {
2    Θ(1)
3    Θ(n^2)
4  }
```

Example

```
1  public void countDuplicates(Data[] data) {
2    Θ(1 + n²) = Θ(n²)
3  }
```

Example

```
1  public void countDuplicates(Data[] data) {
2    System.out.println("Counting duplicates");
3    int count = 0;
4    for (int i = 0; i < data.length; i++) {
5      for (int j = i+1; j < data.length; j++) {
6        if (data[i] == data[j]) {
7          count++;
8        }
9      }
10   }
11 }
```

$$1 + \sum_{i=1}^{n} \sum_{j=i+1}^{n} 1 = \frac{n^2}{2} + \frac{3n}{2} + 1 \in \theta(n^2)$$

Tip

If you know the complexity of a piece of code, you can use that instead of an exact number of steps

Example 2

```
1  public void updateCells(Data[] data) {
2    System.out.println("Updating our data...");
3    int num_neighbors = 8;
4    for (Data d : data) {
5      System.out.println("Processing element " + d);
6      for (int i = 0; i < num_neighbors; i++) {
7        data.weight += data.neighbor[i] / num_neighbors;
8      }
9    }
10 }
```

Example 2

```
1  public void updateCells(Data[] data) {
2    System.out.println("Updating our data...");
3    int num_neighbors = 8;
4    for (Data d : data) {
5      System.out.println("Processing element " + d);
6      for (int i = 0; i < num_neighbors; i++) {
7        data.weight += data.neighbor[i] / num_neighbors;
8      }
9    }
10 }
```

Θ(1)

Θ(1)

Example 2

```
1  public void updateCells(Data[] data) {
2    Θ(1)
3    for (Data d : data) {
4      Θ(1)
5    }
6  }
```

Example 2

```
1  public void updateCells(Data[] data) {
2    Θ(1)
3    for (Data d : data) {
4      Θ(1)
5    }
6  }
```

$n \cdot \Theta(1) = \Theta(n)$

Example 2

```
1  public void updateCells(Data[] data) {
2    Θ(1)
3    Θ(n)
4  }
```

Example 2

```
1  public void updateCells(Data[] data) {
2    Θ(1 + n) = Θ(n)
3  }
```

Example 2

```
1  public void updateCells(Data[] data) {
2    System.out.println("Updating our data...");
3    int num_neighbors = 8;
4    for (Data d : data) {
5      System.out.println("Processing element " + d);
6      for (int i = 0; i < num_neighbors; i++) {
7        data.weight += data.neighbor[i] / num_neighbors;
8      }
9    }
10 }
```

$$\theta(1) + \sum_{d \in data} \theta(1) \in \theta(n)$$

Tip

You are not counting "lines of code"

A single line of code does not necessarily mean a single step

Conversely, a loop doesn't guarantee a non-constant runtime

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          data.remove(j--);
7        }
8      }
9    }
10 }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          data.remove(j--);
7        }
8      }
9    }
10 }
```

Is this still a single "step"?

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#remove-int-

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          data.remove(j--);
7        }
8      }
9    }
10 }
```

Is this still a single "step"?

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html#remove-int-

Could have to move all ***n*** elements in the worst case, so upper bound is ***O*(*n*)**

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          O(n)
7        }
8      }
9    }
10 }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          O(n)
7        }
8      }
9    }
10 }
```

The body is only executed if the condition is true…

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          O(n)
7        }
8      }
9    }
10 }
```

$$\begin{cases} O(1) & \text{if condition is false} \\ O(n) & \text{if condition is true} \end{cases}$$

The body is only executed if the condition is true…

$$
\begin{aligned}
c \cdot O(f(N)) &= O(f(N)) \\
N \cdot O(f(N)) &= O(N \cdot f(N)) \\
g(N) \cdot O(f(N)) &= O(g(N) \cdot f(N)) \\
O(g(N)) + O(f(N)) &= O(g(N) + f(N)) \\
&= \text{the greater of } O(f(N)) \textbf{ or } O(g(N)) \\
\begin{cases} O(g(N)) & \textbf{if one thing} \\ O(f(N)) & \textbf{otherwise} \end{cases} &= \text{the greater of } O(f(N)) \textbf{ or } O(g(N)) \\
&= O(g(N) + f(N))
\end{aligned}
$$

Algebra with Big-O

$$
\begin{aligned}
c \cdot \Omega(f(N)) &= \Omega(f(N)) \\
N \cdot \Omega(f(N)) &= \Omega(N \cdot f(N)) \\
g(N) \cdot \Omega(f(N)) &= \Omega(g(N) \cdot f(N)) \\
\Omega(g(N)) + \Omega(f(N)) &= \Omega(g(N) + f(N)) \\
&= \text{the greater of } \Omega(f(N)) \textbf{ or } \Omega(g(N))
\end{aligned}
$$

$$
\begin{cases}
\Omega(g(N)) & \textbf{if one thing} \\
\Omega(f(N)) & \textbf{otherwise}
\end{cases}
= \text{Smaller of f(N) or g(N)}
$$

Algebra with Big-Ω

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          O(n)
7        }
8      }
9    }
10 }
```

$$\begin{cases} O(1) & \text{if condition is false} \\ O(n) & \text{if condition is true} \end{cases}$$

The body is only executed if the condition is true…

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          O(n)                                    O(n)
7        }
8      }
9    }
10 }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        O(n)
6      }
7    }
8  }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      O(n^2)
5    }
6  }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    O(1)
3    O(n^3)
4  }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    O(1 + n^3) = O(n^3)
3  }
```

Example 3

```
1  public void makeUnique(ArrayList<Data> data) {
2    System.out.println("Making all elements in data unique");
3    for (int i = 0; i < data.length; i++) {
4      for (int j = i+1; j < data.length; j++) {
5        if (data.get(i) == data.get(j)) {
6          data.remove(j--);
7        }
8      }
9    }
10 }
```

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} O(n) \in O(n^3)$$

Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        if (list.get(j) < list.get(j+1)) {
5          Integer tmp = list.get(j);
6          list.set(j, list.get(j+1));
7          list.set(j+1, tmp);
8        }
9      }
10   }
11 }
```

Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        if (list.get(j) < list.get(j+1)) {
5          Integer tmp = list.get(j);
6          list.set(j, list.get(j+1));
7          list.set(j+1, tmp);
8        }
9      }
10   }
11 }
```

Is this implementation of bubble sort an $\mathbf{O}(n^2)$ algorithm?

Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        if (list.get(j) < list.get(j+1)) {
5          Integer tmp = list.get(j);
6          list.set(j, list.get(j+1));
7          list.set(j+1, tmp);
8        }
9      }
10   }
11 }
```

Is this implementation of bubble sort an $O(n^2)$ algorithm?

What is the runtime of `get()/set()`?

Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        O(?)
5      }
6    }
7  }
```

Is this implementation of bubble sort an $\mathbf{O}(n^2)$ algorithm?

What is the runtime of `get()/set()`?

Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        O(?)
5      }
6    }
7  }
```

$$\sum_{i=0}^{n-2} \sum_{j=i}^{n-1} O(?)$$

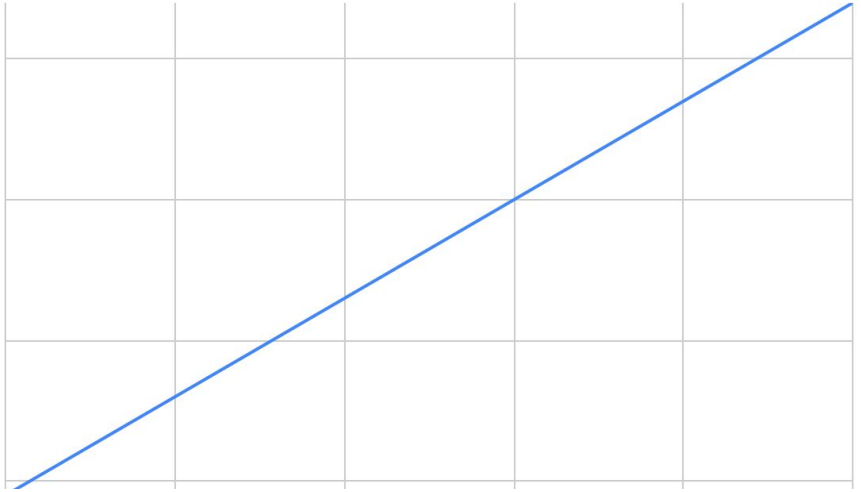
Is this implementation of bubble sort an **O(n^2)** algorithm?

What is the runtime of **`get()/set()`**?

Comparing Random Access for Array vs List

Array

List



Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        O(?)
5      }
6    }
7  }
```

$$\sum_{i=0}^{n-2} \sum_{j=i}^{n-1} O(?)$$

Is this implementation of bubble sort an $O(n^2)$ algorithm?

What is the runtime of `get()/set()`?

If our list is an array, $O(?) = O(1)$, so overall runtime is $O(n^2)$

Real Example

```
1  public void bubbleSort(List<Integer> list) {
2    for (int i = list.size() - 2; i >= 0; i--) {
3      for (int j = i; j < list.size() - 1; j++) {
4        O(?)
5      }
6    }
7  }
```

$$\sum_{i=0}^{n-2} \sum_{j=i}^{n-1} O(?)$$

Is this implementation of bubble sort an $O(n^2)$ algorithm?

What is the runtime of `get()/set()`?

If our list is an array, $O(?) = O(1)$, so overall runtime is $O(n^2)$

If our list is a LinkedList, $O(?) = O(n)$, so overall runtime is $O(n^3)$