

CSE 250

Data Structures

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

Lec 08: 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

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 }
```

So what is the complexity and/or tight upper bound on the runtime of this code?

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 }
```

So what is the complexity and/or tight upper bound on the runtime of this code?

First we must determine the runtime growth function

Control Flow

Remember the different types of control flow:

- Sequential (statements executed one after another)
 - Add the number of steps together
 - If I do **A** then **B**, the total cost is the **cost to do A** plus the **cost to do B**

Control Flow

Remember the different types of control flow:

- Sequential (statements executed one after another)
 - Add the number of steps together
 - If I do **A** then **B**, the total cost is the **cost to do A** plus the **cost to do B**
- Selection (conditional execution of statements)
 - Our growth function will be a piecewise function

Control Flow

Remember the different types of control flow:

- Sequential (statements executed one after another)
 - Add the number of steps together
 - If I do **A** then **B**, the total cost is the **cost to do A** plus the **cost to do B**
- Selection (conditional execution of statements)
 - Our growth function will be a piecewise function
- Repetition (repeating execution of one or more statements)
 - Add up the total number of steps...with summations

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 }
```

**Let's break this
function down into
sequential pieces**

Example

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

**Let's break this
function down into
sequential pieces**

Example

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

Let's break this
function down into
sequential pieces

$$T(n) = T_A(n) + T_B(n) + T_C(n)$$

Example

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

Let's break this
function down into
sequential pieces

$$T(n) = 1 + 1 + T_c(n)$$

Example

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

Now how many steps
does C take in total?

Let's isolate it

$$T(n) = 1 + 1 + T_c(n)$$

Example

```
1  for (int i = 0; i < data.length; i++) {
2      for (int j = i+1; j < data.length; j++) {
3          if (data[i] == data[j]) {
4              count++;
5          }
6      }
7  }
```

The outer loop starts at **$i = 0$** , and goes up to **data.length (n)** stepping by **1**

Each iteration it does a certain amount of work (which may or may not depend on i)

Example

```
1  for (int i = 0; i < data.length; i++) {  
2    for (int j = i+1; j < data.length; j++) {  
3      if (data[i] == data[j]) {  
4        count++;  
5      }  
6    }  
7  }
```

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


The outer loop starts at $i = 0$, and goes up to **data.length (n)** stepping by 1

Each iteration it does a certain amount of work (which may or may not depend on i)

Example

```
1  for (int i = 0; i < data.length; i++) {  
2      for (int j = i+1; j < data.length; j++) {  
3          if (data[i] == data[j]) {  
4              D      count++;  
5          }  
6      }  
7  }
```

$$\sum_{i=0}^n T_D(n, i)$$

The outer loop starts at $i = 0$, and goes up to **data.length (n)** stepping by 1

Each iteration it does a certain amount of work (which may or may not depend on i)

Example

```
1  for (int i = 0; i < data.length; i++) {  
2      for (int j = i+1; j < data.length; j++) {  
3          if (data[i] == data[j]) {  
4              D      count++;  
5          }  
6      }  
7  }
```

$$\sum_{i=0}^n T_D(n, i)$$

Apply the same approach to determine T_D

Example

```
1  for (int i = 0; i < data.length; i++) {  
2      for (int j = i+1; j < data.length; j++) {  
3          if (data[i] == data[j]) {  
4              count++;  
5          }  
6      }  
7  }
```

D

E

$$\sum_{i=0}^n T_D(n, i)$$

Apply the same approach to determine T_D

Example

```
1  for (int i = 0; i < data.length; i++) {  
2      for (int j = i+1; j < data.length; j++) {  
3          if (data[i] == data[j]) {  
4              count++;  
5          }  
6      }  
7  }
```

D **E**

Apply the same approach to determine T_D

$$\sum_{i=0}^n \sum_{j=i+1}^n T_E(n, i, j)$$

Example

```
1  for (int i = 0; i < data.length; i++) {  
2      for (int j = i+1; j < data.length; j++) {  
3          if (data[i] == data[j]) {  
4              count++;  
5          }  
6      }  
7  }
```

D **E**

Finally let's isolate **E**

$$\sum_{i=0}^n \sum_{j=i+1}^n T_E(n, i, j)$$

Example

1	<code>if (data[i] == data[j]) {</code>
2	<code> count++;</code>
3	<code>}</code>

How many steps does this code take?

Example

1	<code>if (data[i] == data[j]) {</code>
2	<code> count++;</code>
3	<code>}</code>

How many steps does this code take?

What if `data[i] == data[j]`?

Example

```
1   if (data[i] == data[j]) {  
2       count++;  
3   }
```

How many steps does this code take?

What if `data[i] == data[j]`?

1 step to check the condition plus 1 step to increment count = 2

Example

1	<code>if (data[i] == data[j]) {</code>
2	<code> count++;</code>
3	<code>}</code>

How many steps does this code take?

What if `data[i] != data[j]`?

Example

```
1   if (data[i] == data[j]) {  
2       count++;  
3   }
```

How many steps does this code take?

What if `data[i] != data[j]`?

1 step to check the condition

Example

```
1   if (data[i] == data[j]) {  
2       count++;  
3   }
```

How many steps does this code take?

$$T_E(n, i, j) = \begin{cases} 2 & \text{if } \text{data}[i] == \text{data}[j] \\ 1 & \text{otherwise} \end{cases}$$

Putting It All Together

```
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 }
```

$$T(n) = 1 + 1 + \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} T_E(n, i, j)$$
$$T_E(n, i, j) = \begin{cases} 2 & \text{if data}[i] == \text{data}[j] \\ 1 & \text{otherwise} \end{cases}$$

Tip #1

Tip: If you know the complexity/bound of a growth function, you can use the complexity/bound instead of the growth function

What is the complexity of T_E ?

$$T_E(n, i, j) = \begin{cases} 2 & \text{if data}[i] == \text{data}[j] \\ 1 & \text{otherwise} \end{cases}$$

Tip #1

Tip: If you know the complexity/bound of a growth function, you can use the complexity/bound instead of the growth function

What is the complexity of T_E ? $\Theta(1)$

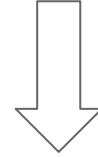
$$T_E(n, i, j) = \begin{cases} 2 & \text{if data}[i] == \text{data}[j] \\ 1 & \text{otherwise} \end{cases}$$

Tip #1

$$T(n) = 1 + 1 + \sum_{i=0}^n \sum_{j=i+1}^n T_E(n, i, j)$$

Tip #1

$$T(n) = 1 + 1 + \sum_{i=0}^n \sum_{j=i+1}^n T_E(n, i, j)$$

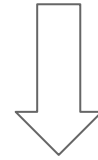


$$T(n) = 1 + 1 + \sum_{i=0}^n \sum_{j=i+1}^n \Theta(1)$$

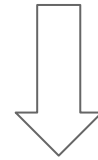
Tip #1

Now the summations are much easier to deal with

$$T(n) = 1 + 1 + \sum_{i=0}^n \sum_{j=i+1}^n T_E(n, i, j)$$



$$T(n) = 1 + 1 + \sum_{i=0}^n \sum_{j=i+1}^n \Theta(1)$$



$$T(n) = \Theta(1) + \sum_{i=0}^n \sum_{j=i+1}^n \Theta(1)$$

Tip #1

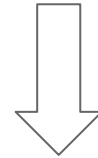
Now the summations are much easier to deal with

$$T(n) = \Theta(1) + \sum_{i=0}^n \sum_{j=i+1}^n \Theta(1)$$

Tip #1

Now the summations are much easier to deal with

$$T(n) = \Theta(1) + \sum_{i=0}^n \sum_{j=i+1}^n \Theta(1)$$

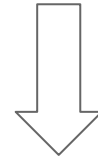


$$T(n) = \Theta(1) + \sum_{i=0}^n (n - i)\Theta(1)$$

Tip #1

Now the summations are much easier to deal with

$$T(n) = \Theta(1) + \sum_{i=0}^n \sum_{j=i+1}^n \Theta(1)$$



$$T(n) = \Theta(1) + \sum_{i=0}^n (n - i)\Theta(1)$$



$$T(n) = \Theta(1) + \Theta(n^2)$$

$$c \cdot \theta(f(N)) = \theta(f(N))$$

$$N \cdot \theta(f(N)) = \theta(N \cdot f(N))$$

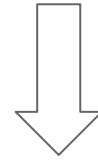
$$g(N) \cdot \theta(f(N)) = \theta(g(N) \cdot f(N)) \quad (\text{if } \theta(g(N)) \text{ exists})$$

$$\begin{aligned} \theta(g(N)) + \theta(f(N)) &= \theta(g(N) + f(N)) \\ &= \text{The greater of } \theta(f(N)) \text{ or } \theta(g(N)) \end{aligned}$$

Tip #1

Finish off the simplification using the rules of Big-Theta algebra

$$T(n) = \Theta(1) + \Theta(n^2)$$



$$T(n) = \Theta(n^2)$$

Tip #2

Tip: Start by identifying the parts of the code that have a constant runtime

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 }
```

$\Theta(1)$

$\Theta(1)$

Example 2

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

Example 2

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

$$\sum_{i=1}^n \Theta(1) = n \cdot \Theta(1) = \Theta(n)$$

Tip #3: If the cost of an iteration does not depend on the loop variable, you can just multiple the cost of an iteration by the number of iterations
(*use with caution...*)

Example 2

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

Example 2

```
1 public void updateCells(Data[] data) {  
2      $\Theta(1 + n) = \Theta(n)$   
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 }
```

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

Common Pitfall

Remember: 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 single line 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 single line 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 }
```

$\left\{ \begin{array}{l} O(1) \text{ if condition is false} \\ O(n) \text{ if condition is true} \end{array} \right.$

The body is only executed if the condition is true...

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) = O(N \cdot f(N))$$

$$g(N) \cdot O(f(N)) = O(g(N) \cdot f(N))$$

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

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

$$c \cdot \Omega(f(N)) = \Omega(f(N))$$

$$N \cdot \Omega(f(N)) = \Omega(N \cdot f(N))$$

$$g(N) \cdot \Omega(f(N)) = \Omega(g(N) \cdot f(N))$$

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

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

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 }
```

The body is only executed if the condition is true...

$\left\{ \begin{array}{l} O(1) \text{ if condition is false} \\ O(n) \text{ if condition is true} \end{array} \right.$

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 }
```

$O(n)$

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              $O(n)$   
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 }
```

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

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 }
```

Remember: Plugging in $O(n)$ here is a shortcut...we may be able to be more specific once we understand how remove depends on j

When in doubt, do the full summation

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

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 $O(n^2)$ 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 $O(n^2)$ algorithm?
What is the runtime of `get()/set()`?

Real Example

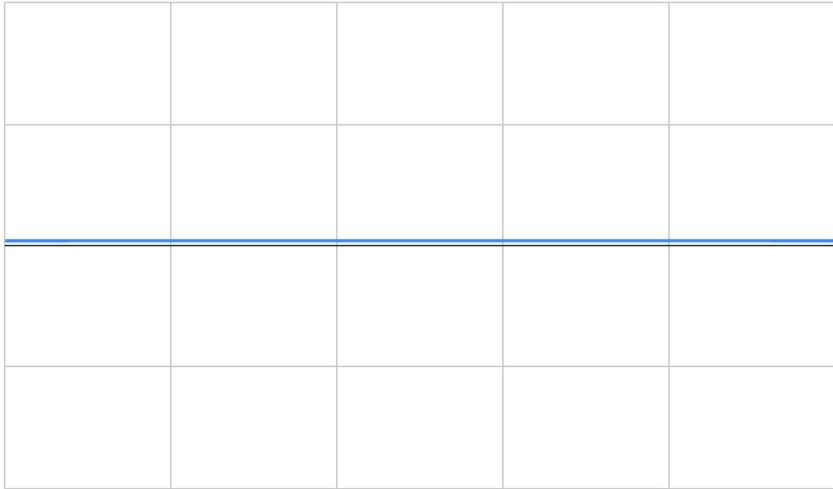
```
1 public void bubbleSort(List<Integer> list) {  
2     for (int i = list.size() - 2; i >= 0; i--) {  
3         for (int j = i; j < list.size() - 1; j++) {  
4             O(?)  
5         }  
6     }  
7 }
```

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

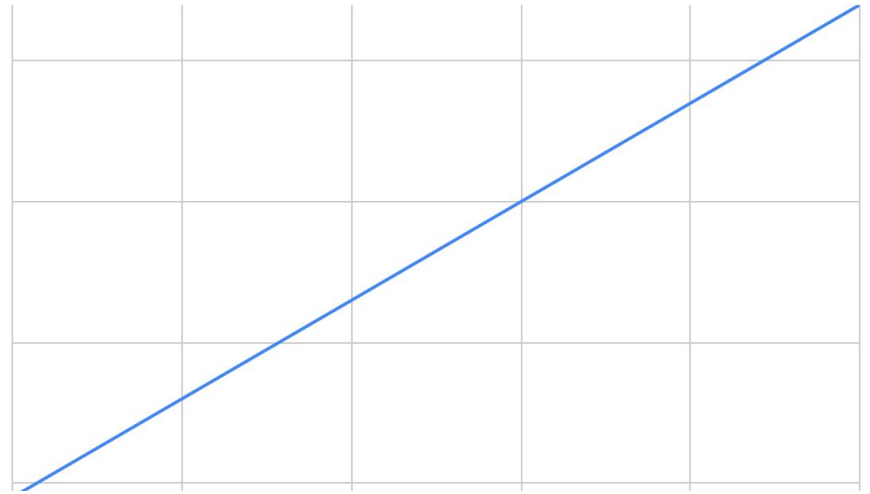
Is this implementation of bubble sort an $O(n^2)$ algorithm?
What is the runtime of `get()/set()`?

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 $O(n^3)$