CS 600.226: Data Structures

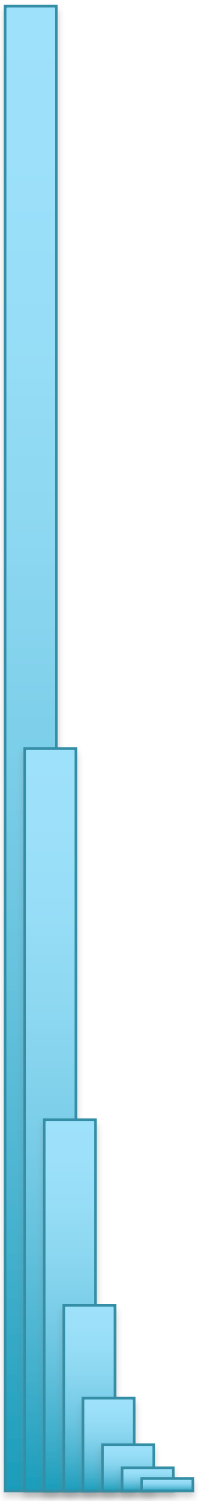
Michael Schatz

Sept 14 2018

Lecture 7. More Complexity

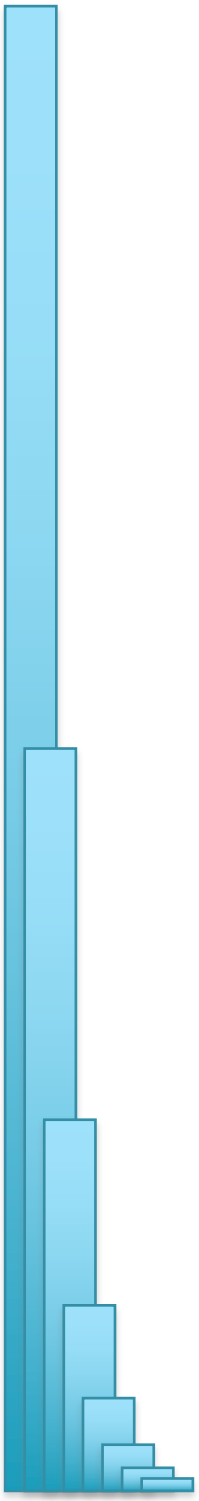
Agenda

1. *Review HW1*
2. *Introduce HW 2*
3. *Recap & continuation on complexity*



Assignment 1: Due Friday Sept 14 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment01/assignment01.md



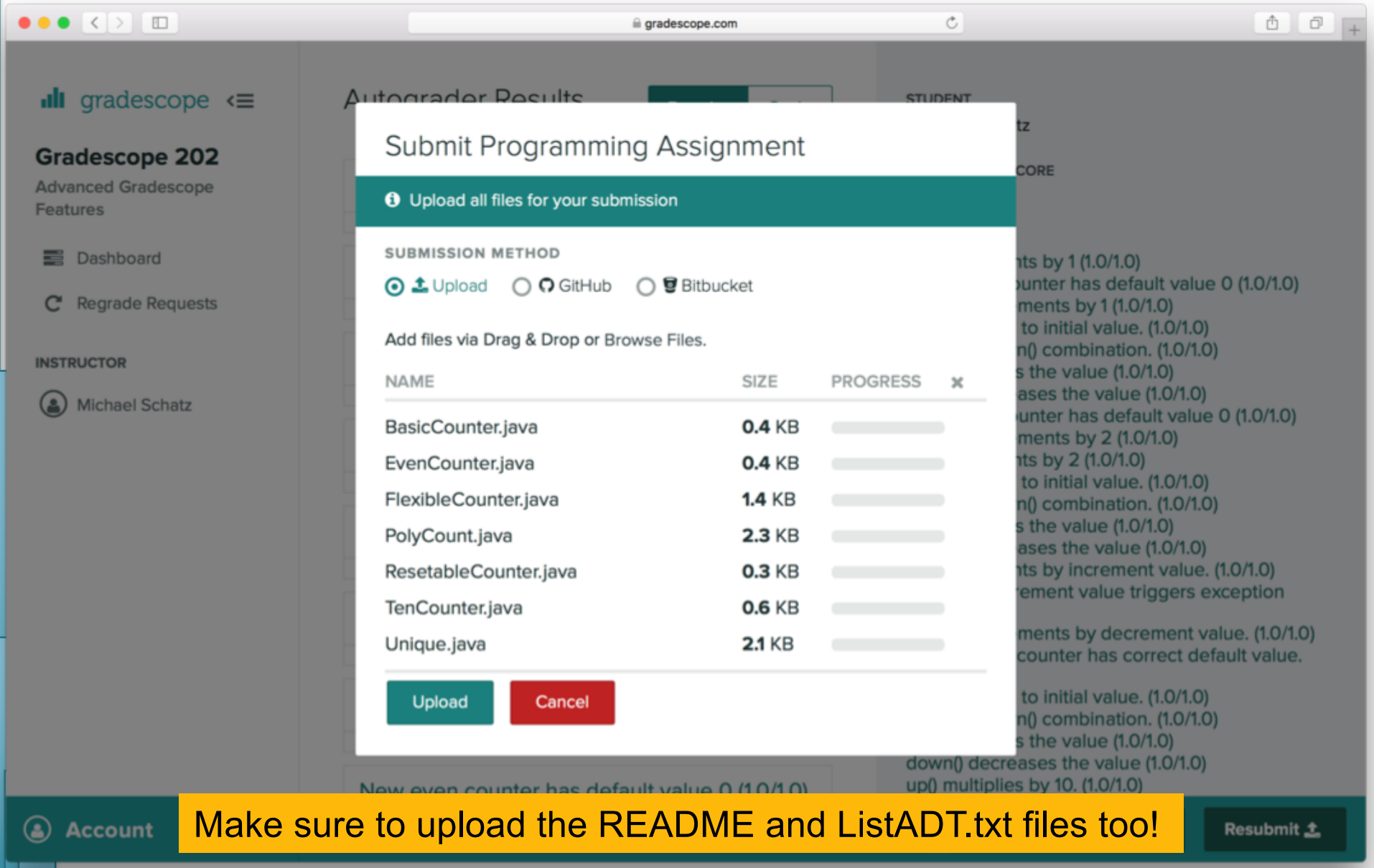
Assignment 1: Warming Up

- **Out on:** September 7, 2016
- **Due by:** September 14, 2016 before 10:00 pm
- **Collaboration:** None
- **Grading:**
  - Functionality 65%
  - ADT Solution 30%
  - Solution Design and READMDE 5%
  - Style 0%

Overview

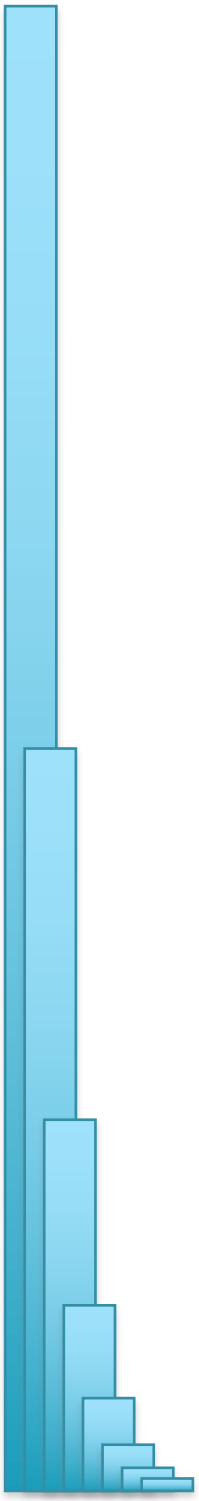
The first assignment is mostly a warmup exercise to refresh your knowledge of Java and an ADT problem to start you thinking more abstractly about your data.

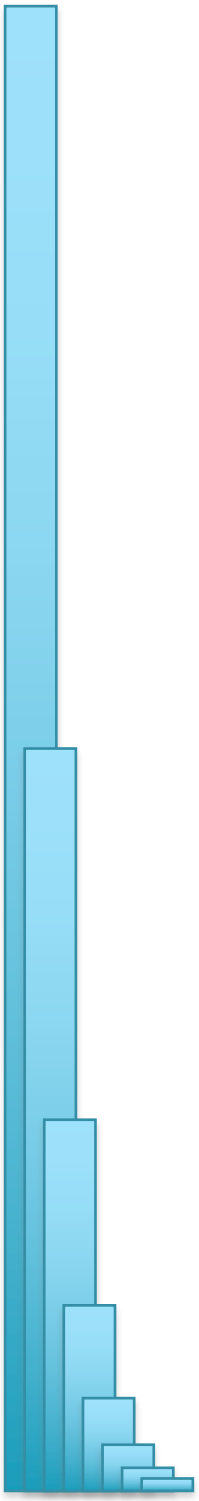
GradeScope.com
Entry Code: MDJYER



Agenda

1. *Review HW1*
2. *Introduce HW 2*
3. *Recap & continuation on complexity*





Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Assignment 2: Arrays of Doom!

Out on: September 14, 2018
Due by: September 21, 2018 before 10:00 pm
Collaboration: None
Grading:

- Functionality 65%
- ADT Solution 20%
- Solution Design and README 5%
- Style 10%

Overview
The second assignment is mostly about arrays, notably our own array specifications and implementations, not just the built-in Java arrays. Of course we also once again snuck a small ADT problem in there...

Note: The grading criteria now include **10% for programming style**. Make sure you use Checkstyle with the correct configuration file from Github!

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Problem 1: Revenge of Unique (30%)

You wrote a small Java program called Unique for Assignment 1. The program accepted any number of command line arguments (each of which was supposed to be an integer) and printed each unique integer it received back out once, eliminating duplicates in the process.

For this problem, you will implement a new version of Unique called ***UniqueRevenge*** with two major changes:

- First, you are no longer allowed to use Java arrays (nor any other advanced data structure), but you can use our Array interface and our SimpleArray implementation from lecture (also available on github)

- Second, you're going to modify the program to read the integers from standard input instead of processing the command line.

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Example:

```
$ java UniqueRevenge
1 9 2
3
1 4
9 5 3
6 0
<ctrl-d> or <ctrl-z>
1
9
2
3
4
5
6
0
```

Hints

- Reading numbers from standard input can be accomplished using a java.util.Scanner object that has been wrapped around System.in which is Java's name for the standard input stream.
- Make sure you hit return one last time at the end of your input and only then signal end-of-file with the appropriate key-combination for your operating system (this restriction doesn't apply when you use I/O redirection to give input to the program, a highly recommended practice for testing).
- You will have to process an unbounded number of inputs, which requires that you keep track of how "full" the array is. When nothing fits into the array anymore, you'll have to "grow" it somehow. The best approach is to double the size of the array when you are out of space. (We'll talk about the reasons for this in lecture next week.)
- Do not try to change everything at once, there are too many "moving parts" to get things right that way. Instead, choose one thing to change,for example just the way input is given to the program, finish that, test it, and only then move on to the next thing. **Remember: Baby steps!**

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

```
import java.util.Scanner;

public class PrintInts {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        while (s.hasNextInt()) {
           int i = s.nextInt();
           System.out.println("found: " + i);
        }
    }
}
```

```
$ java PrintInts
1 2 3 4 5
found: 1
found: 2
found: 3
found: 4
found: 5
6 7 8
found: 6
found: 7
found: 8
9
found: 9
000
found: 0
<ctrl-d>
```

```
$ seq 1 5 > nums
$ cat nums
1
2
3
4
5
$ cat nums | java PrintInts
found: 1
found: 2
found: 3
found: 4
found: 5
```

```
$ seq 1 1000 > nums
$ head -2 nums
1
2
$ tail -2 nums
999
1000
$ cat nums | java PrintInts > results
$ head -2 results
found: 1
found: 2
$ tail -2 results
found: 999
found: 1000
$ wc -l results
    1000 results
```

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

- First, you are no longer allowed to use Java arrays (nor any other advanced data structure), but you can use our Array interface and our SimpleArray implementation from lecture (also available on github)

Wait a second, Im only allowed to use SimpleArrays, but the constructor requires giving a size…. How do I know how big to make it???

☺ Call the constructor with an initial buffer size, and then grow the buffer as needed. Make sure to keep track of how many slots are really used.

Wait a second, how do I grow a buffer????

☺ Make a new array that is larger, copy everything over.

Wait a second, how much bigger???

☺ Doubling is usually a good rule (wait a few weeks for a formal analysis)

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Problem 2: Flexible Arrays (20%)

Develop an algebraic specification for the abstract data type FlexibleArray which works like the existing Array ADT for the most part **except** that both its **lower** and its **upper** index bound are set when the array is created. The lower as well as upper bound can be **any** integer, provided the lower bound is **less than or equal** the upper bound.

Write up the specification for FlexibleArray in the format we used in lecture and **comment** on the design decisions you had to make. Also, tell us what kind of array **you** prefer and why.

Hints

- A FlexibleArray for which the lower bound equals the upper bound has exactly one slot.
- Your FlexibleArray is **not** the Array ADT we did in lecture; it doesn't have to support the exact same set of operations.

Array ADT

```
adt Array
  uses Any, Integer
  defines Array<T: Any>

  operations
      new: Integer x T ---> Array<T>
      get: Array<T> x Integer ---> T
      put: Array<T> x Integer x T ---> Array<T>
      length: Array<T> ---> Integer

  axioms
      get(new(n, t), i) = t
      get(put(a, i, t), j) = (if i = j then t else get(a, j))
      length(new(n, t)) = n
      length(put(a, i, t)) = length(a)

  preconditions
      new(n, t): 0 < n
      get(a, i): 0 <= i < length(a)
      put(a, i, t): 0 <= i < length(a)
```

Uses two related ADTs

Defines method signatures

Enforced by asserts

Enforced by exceptions

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Problem 3: Sparse Arrays (35%)

A **sparse** array is an array in which **relatively few** positions have values that differ from the initial value set when the array was created. For sparse arrays, it is wasteful to store the value of **all** positions explicitly since **most of them never change** and take the default value of the array. Instead, we want to store positions that **have actually been changed**.

For this problem, write a class SparseArray that implements the Array interface we developed in lecture (the same interface you used for Problem 1 above). **Do not modify the Array interface in any way!** Instead of using a plain Java array like we did for SimpleArray, your SparseArray should use a **linked list** of Node objects to store values, similar to the ListArray from lecture (and available in github). However, your nodes no longer store just the **data** at a certain position, they also store **the position itself**!

Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Here's a rough outline of how your implementation could work:

- Start with an empty list (instead of the complete list we built in the constructor of ListArray).
- For put, check if the relevant position has been modified before (meaning a Node object exists for that position); if not, add a Node to the list for the position and its new value; otherwise update the correct Node to the new value.
- For get, check if the relevant position has been modified before; if not, return the default value; otherwise, return the value found in the relevant Node object.

Important: Your Node class must be nested inside your SparseArray class with private visibility! Clients should not be able to "touch" Node objects in any way!

```
new SparseArray(10, “Mike”)
```

SparseArray

int length: 10
String default: Mike
Node list: null

```
sa.put(5, “Peter”)
```

SparseArray

int length: 10
String default: Mike
Node list:

Node

int pos: 5
String data: Peter
Node next: null

```
sa.put(8, “Kelly”)
```

Assignment 2: Due Friday Sept 21 @ 10pm

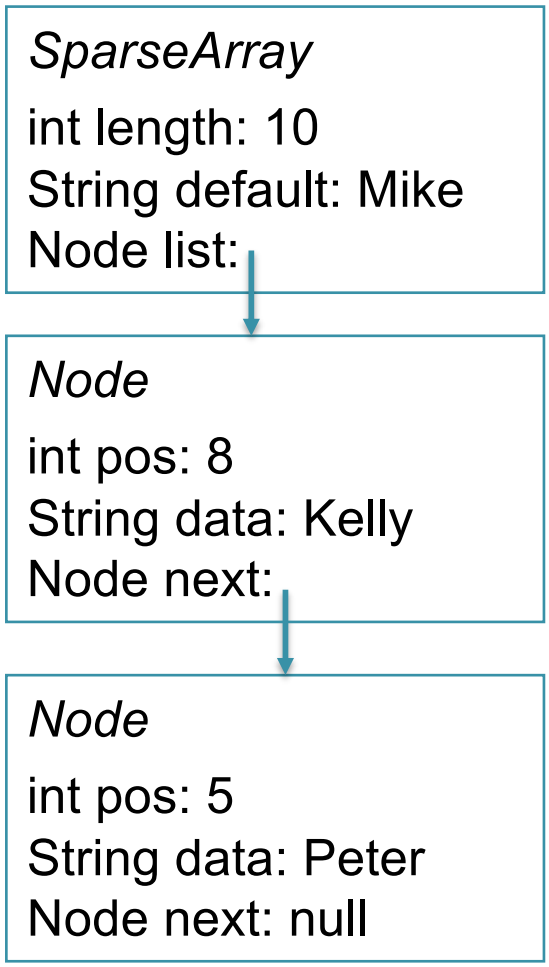
https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Here's a rough outline of how your implementation could work:

- Start with an empty list (instead of the complete list we built in the constructor of ListArray).
- For put, check if the relevant position has been modified before (meaning a Node object exists for that position); if not, add a Node to the list for the position and its new value; otherwise update the correct Node to the new value.
- For get, check if the relevant position has been modified before; if not, return the default value; otherwise, return the value found in the relevant Node object.

Important: Your Node class must be nested inside your SparseArray class with private visibility! Clients should not be able to "touch" Node objects in any way!

```
sa.put(8, "Kelly")
```




```
sa.put(5, "James")
```

Assignment 2: Due Friday Sept 21 @ 10pm

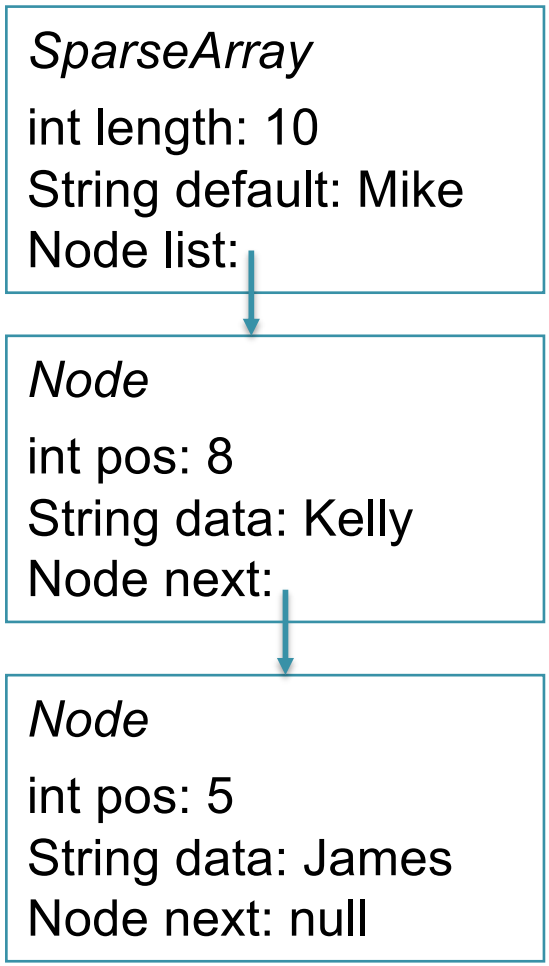
https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Here's a rough outline of how your implementation could work:

- Start with an empty list (instead of the complete list we built in the constructor of ListArray).
- For put, check if the relevant position has been modified before (meaning a Node object exists for that position); if not, add a Node to the list for the position and its new value; otherwise update the correct Node to the new value.
- For get, check if the relevant position has been modified before; if not, return the default value; otherwise, return the value found in the relevant Node object.

Important: Your Node class must be nested inside your SparseArray class with private visibility! Clients should not be able to "touch" Node objects in any way!

```
sa.put(5, "James")
```



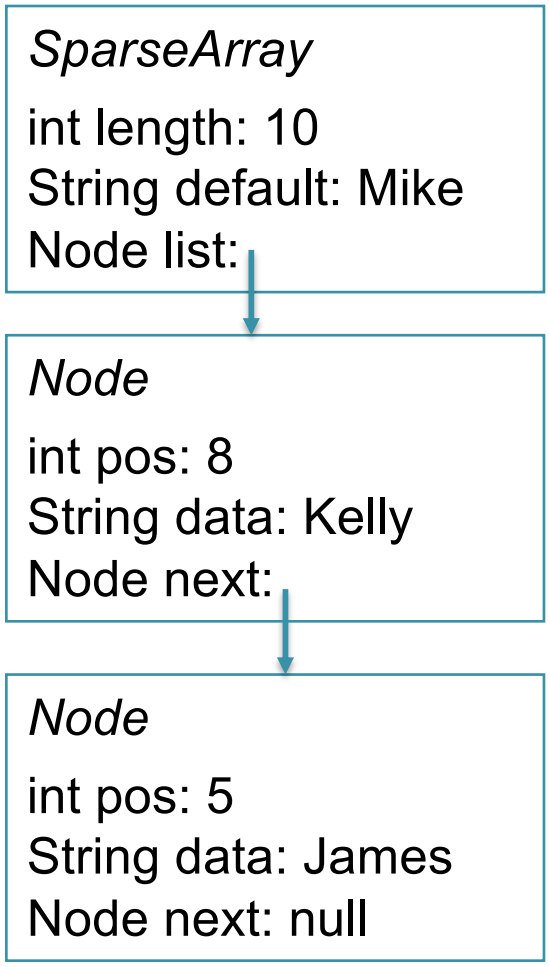
Assignment 2: Due Friday Sept 21 @ 10pm

https://github.com/schatzlab/datastructures2018/blob/master/assignments/assignment02/README.md

Here's a rough outline of how your implementation could work:

- Start with an empty list (instead of the complete list we built in the constructor of ListArray).
- For put, check if the relevant position has been modified before (meaning a Node object exists for that position); if not, add a Node to the list for the position and its new value; otherwise update the correct Node to the new value.
- For get, check if the relevant position has been modified before; if not, return the default value; otherwise, return the value found in the relevant Node object.

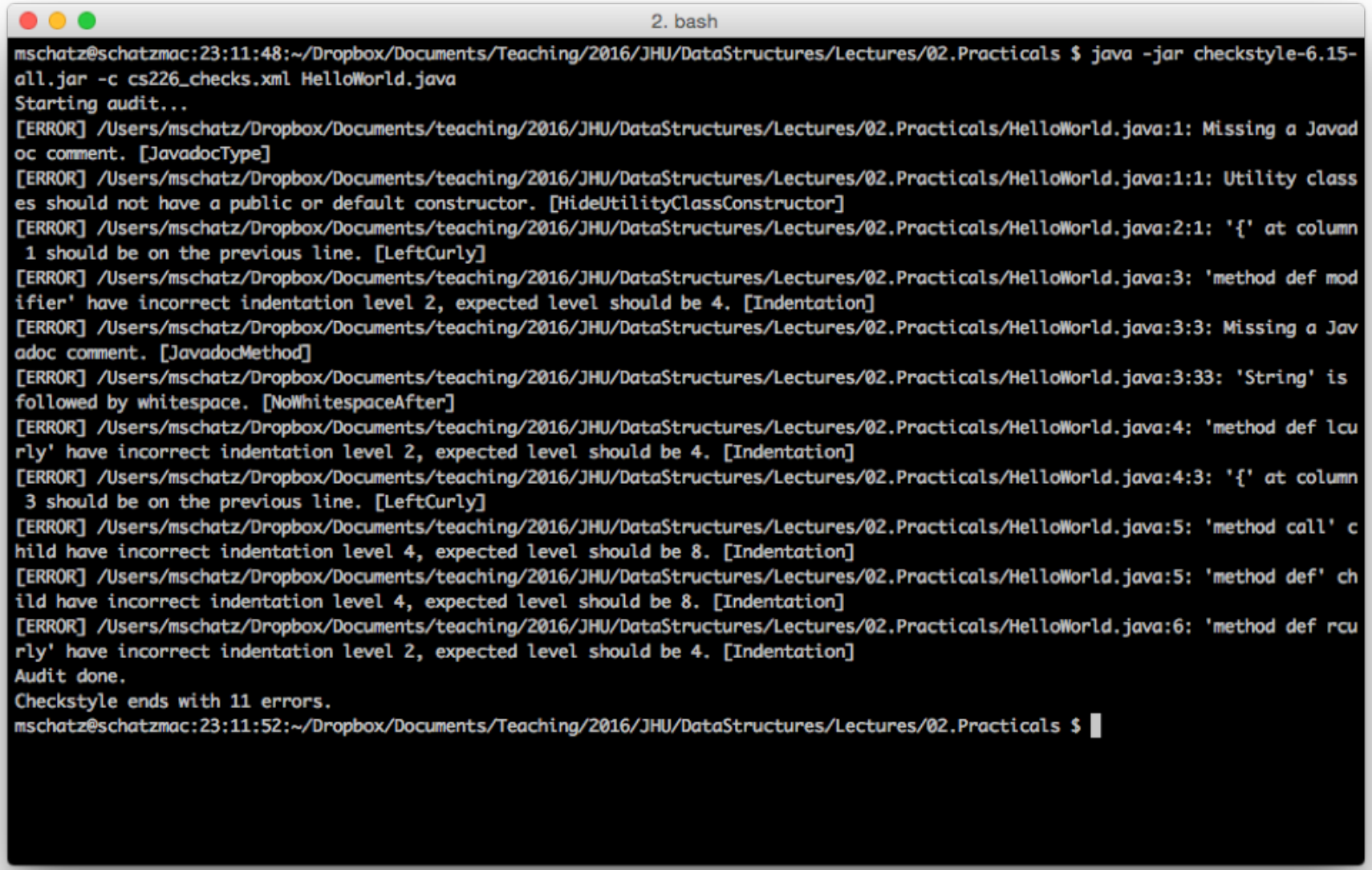
Important: Your Node class must be nested inside your SparseArray class with private visibility! Clients should not be able to "touch" Node objects in any way!




```
sa.get(5) => "James"
sa.get(8) => "Kelly"
sa.get(3) => "Mike"
```

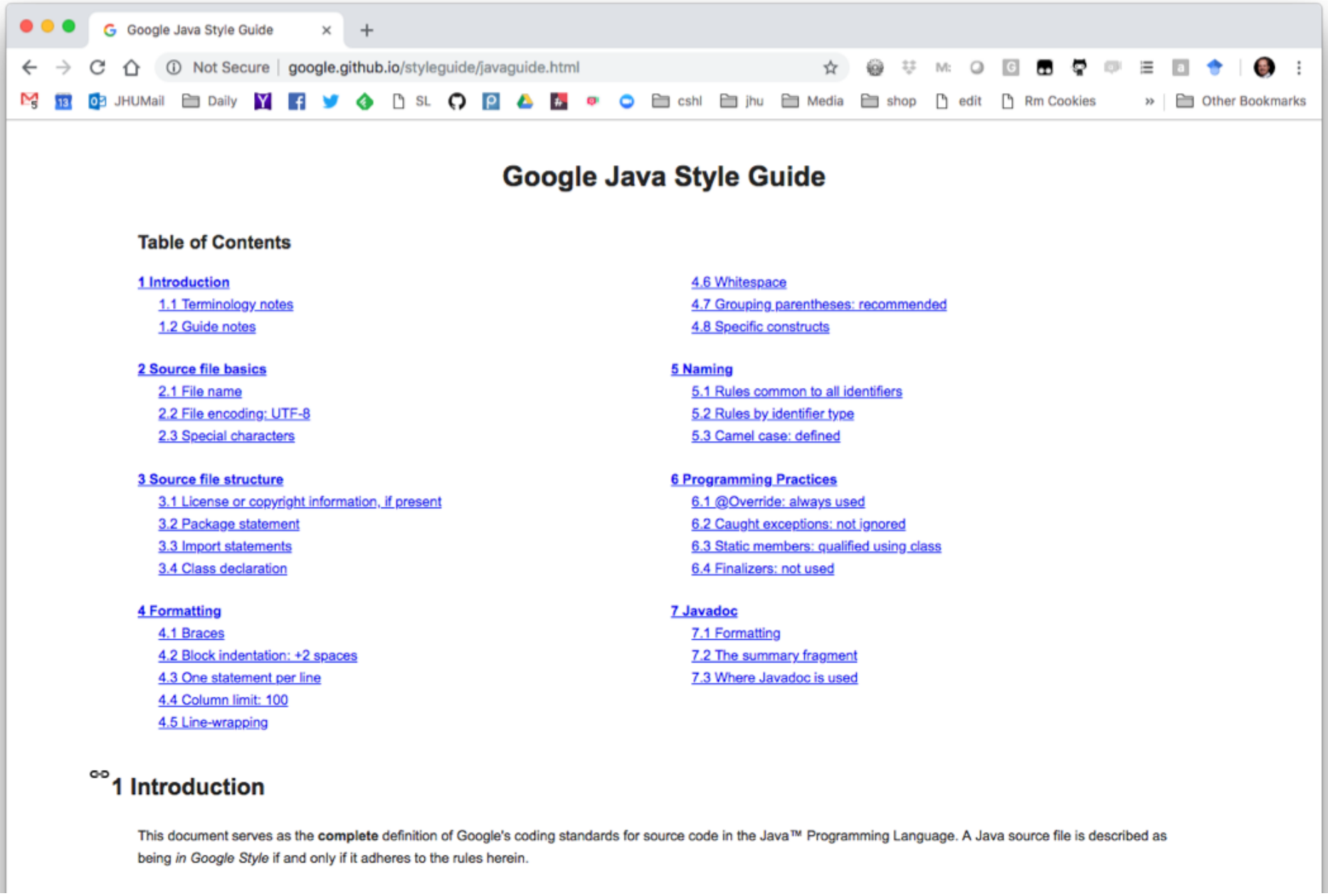
Introduction to Checkstyle

http://checkstyle.sourceforge.net/




```
$ java -jar datastructures2018/resources/checkstyle-8.12-all.jar \
       -c datastructures2018/resources/cs226_checks.xml HelloWorld.java
```

Google's Java Style Guide



Google Java Style Guide

Table of Contents

1 Introduction

This document serves as the **complete** definition of Google's coding standards for source code in the Java™ Programming Language. A Java source file is described as being *in Google Style* if and only if it adheres to the rules herein.

cs226_checks.xml (1)

```
    <!-- maximum 2000 lines by default -->
    <module name="FileLength"/>

    <!-- tabs are not popular in Java -->
    <module name="FileTabCharacter"/>

    <!-- no trailing whitespace, evil -->
    <module name="RegexpSingleline">
       <property name="format" value="\s+$"/>
       <property name="message" value="Line has trailing whitespace."/>
    </module>

    <module name="TreeWalker">
        <!-- enforce Javadoc but not for private stuff -->
        <module name="JavadocMethod">
           <property name="scope" value="protected"/>
        </module>
        <module name="JavadocType">
           <property name="scope" value="protected"/>
        </module>
        <module name="JavadocVariable">
           <property name="scope" value="protected"/>
        </module>
        <module name="JavadocStyle">
           <property name="scope" value="protected"/>
           <!-- empty tags are not okay -->
           <property name="checkEmptyJavadoc" value="true"/>
        </module>
        <!-- being super-picky here, like Google -->
        <module name="JavadocTagContinuationIndentation"/>
...
```

cs226_checks.xml (2)

```
        <!-- various naming conventions -->
        <module name="ConstantName"/>
        <module name="LocalFinalVariableName"/>
        <module name="LocalVariableName"/>
        <module name="MemberName"/>
        <module name="MethodName"/>
        <module name="PackageName"/>
        <module name="ParameterName"/>
        <module name="StaticVariableName"/>
        <module name="TypeName"/>
        <module name="CatchParameterName"/>
        <module name="ClassTypeParameterName"/>
        <module name="InterfaceTypeParameterName"/>
        <module name="MethodTypeParameterName"/>

        <!-- enforce sane imports -->
        <module name="AvoidStarImport"/>
        <module name="IllegalImport"/>
        <module name="RedundantImport"/>
        <module name="UnusedImports"/>

        <!-- size violations -->
        <module name="AnonInnerLength"/>  <!-- default 20 lines -->
        <module name="LineLength"/>       <!-- default 80 chars -->
        <module name="MethodLength"/>     <!-- default 150 lines -->
        <module name="ParameterNumber"/>  <!-- default 7 parameters -->
        <module name="OuterTypeNumber"/>  <!-- default 1 per file -->
...
```

cs226_checks.xml (3)

```
        <!-- whitespace checks -->
        <module name="EmptyForInitializerPad"/>
        <module name="EmptyForIteratorPad"/>
        <module name="EmptyLineSeparator">
            <property name="allowNoEmptyLineBetweenFields" value="true" />
        </module>
        <module name="GenericWhitespace"/>
        <module name="MethodParamPad"/>
        <module name="NoLineWrap"/>
        <module name="NoWhitespaceAfter"/>
        <module name="NoWhitespaceBefore"/>
        <module name="OperatorWrap"/>
        <module name="ParenPad"/>
        <module name="TypecastParenPad"/>
        <module name="WhitespaceAfter"/>
        <module name="WhitespaceAround">
           <!-- empty methods look better this way -->
           <property name="allowEmptyMethods" value="true" />
           <property name="allowEmptyConstructors" value="true" />
        </module>

        <!-- sane use of modifiers (sane is a relative term) -->
        <module name="ModifierOrder"/>
        <module name="RedundantModifier"/>

        <!-- block checks -->
        <module name="AvoidNestedBlocks"/>
        <module name="EmptyBlock"/>
        <module name="EmptyCatchBlock"/>
        <module name="LeftCurly"/>
        <module name="NeedBraces"/>
        <module name="RightCurly"/>
...
```

cs226_checks.xml (4)

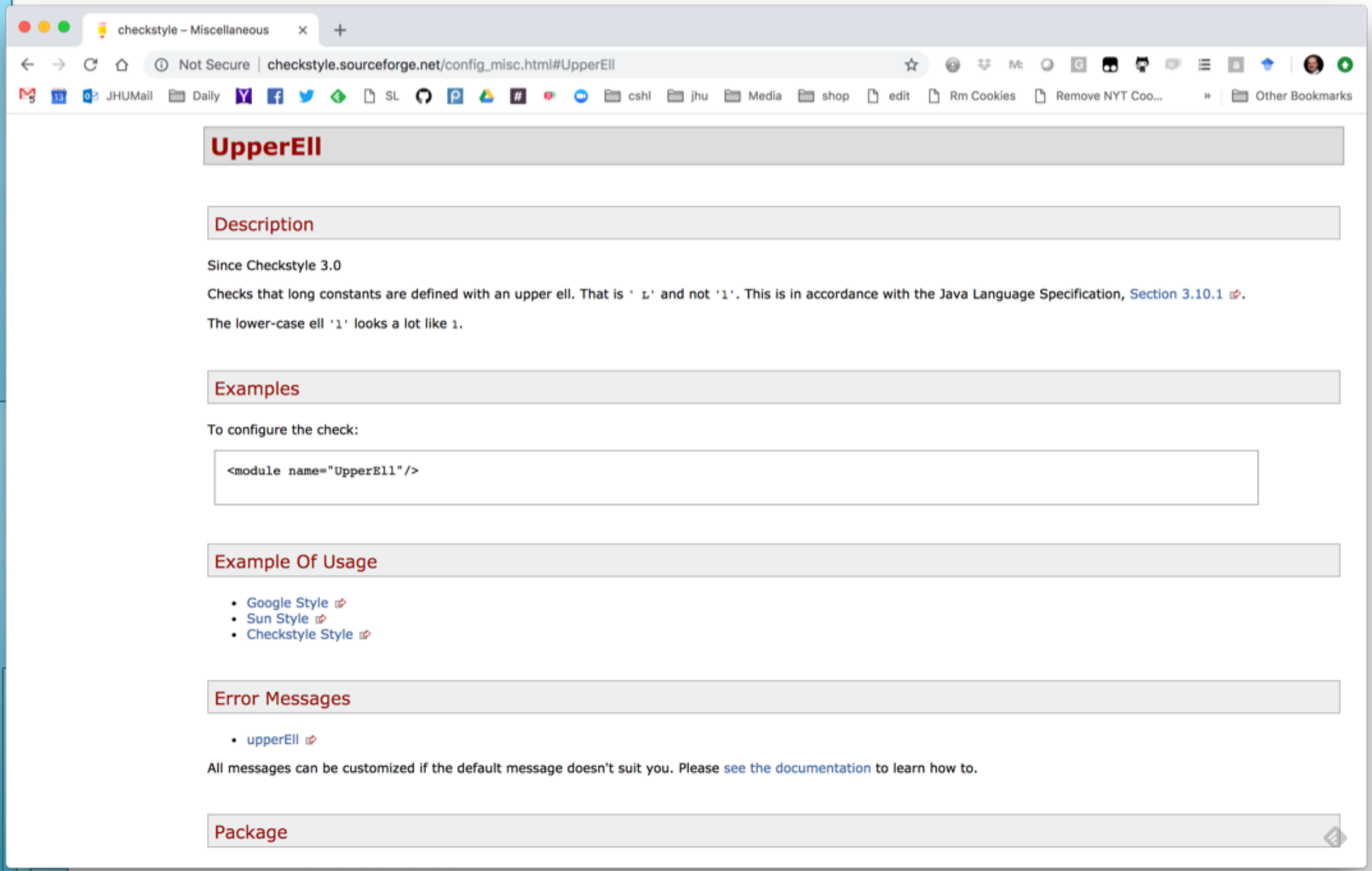
```
        <module name="ArrayTrailingComma"/>
        <module name="CovariantEquals"/>        <!-- avoid accidental overload -->
        <module name="DeclarationOrder"/>       <!-- standardize classes -->
        <module name="DefaultComesLast"/>       <!-- standardize switch -->
        <module name="EmptyStatement"/>
        <module name="EqualsAvoidNull"/>
        <module name="EqualsHashCode"/>
        <module name="ExplicitInitialization"/> <!-- avoid initializing twice -->
        <module name="FallThrough"/>            <!-- avoid forgetting breaks -->
        <module name="HiddenField">             <!-- softened for constructors -->
           <property name="ignoreConstructorParameter" value="true"/>
        </module>
        <module name="IllegalCatch"/>           <!-- avoid overly generic catch -->
        <module name="IllegalThrows"/>          <!-- avoid overly generic throw -->
        <module name="InnerAssignment"/> <!-- avoid assignments as expressions -->
        <!--<module name="MagicNumber"/>--> <!-- more trouble than it's worth, we
can still grade them down but we don't have to force ridiculous constant
declarations -->
        <module name="MissingSwitchDefault"/>   <!-- standardize switch -->
        <module name="ModifiedControlVariable"/>
        <module name="MultipleVariableDeclarations"/>
        <module name="NestedTryDepth"/>         <!-- no try inside a try -->
        <module name="NoClone"/>
        <module name="NoFinalizer"/>
        <module name="OneStatementPerLine"/>
        <module name="OverloadMethodsDeclarationOrder"/>
        <module name="RequireThis"/>            <!-- emphasize non-local stuff -->
        <module name="SimplifyBooleanExpression"/>
        <module name="SimplifyBooleanReturn"/>
        <module name="StringLiteralEquality"/>  <!-- reminder to use equals() -->
...
```

cs226_checks.xml (5)

```
        <!-- annotation checks -->
        <module name="AnnotationLocation"/>    <!-- standardize classes -->

        <!-- design checks -->
        <module name="FinalClass"/>
        <module name="HideUtilityClassConstructor"/>
        <module name="InterfaceIsType"/>
        <module name="MutableException"/>
        <module name="OneTopLevelClass"/>
        <module name="ThrowsCount">
            <property name="ignorePrivateMethods" value="false"/>
        </module>

        <!-- code complexity checks -->
        <module name="ClassFanOutComplexity"/>
        <module name="CyclomaticComplexity"/>
        <module name="NPathComplexity"/>

        <!-- miscellaneous checks -->
        <module name="ArrayTypeStyle"/>
        <module name="CommentsIndentation"/>
        <module name="Indentation"/>           <!-- standardize indentation -->
        <module name="OuterTypeFilename"/>
        <module name="TodoComment"/>
        <module name="UpperEll"/>
```

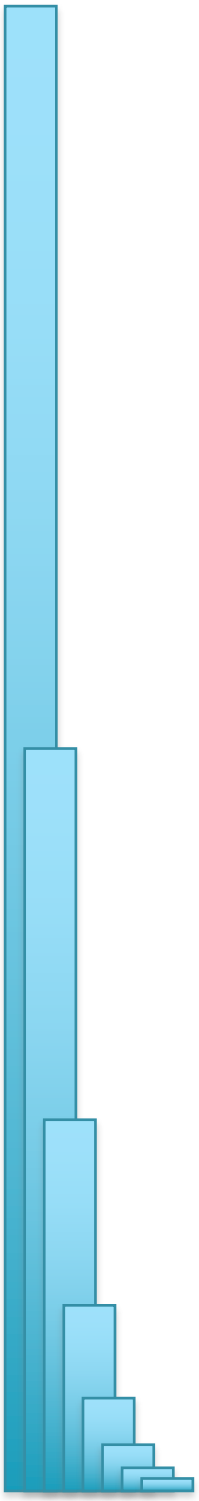
```
$ java -jar datastructures2018/resources/checkstyle-8.12-all.jar \
       -c datastructures2018/resources/cs226_checks.xml HelloWorld.java
```

checkstyle.sourceforge.org



Agenda

1. *Review HW1*
2. *Introduce HW 2*
3. *Recap & continuation on complexity*



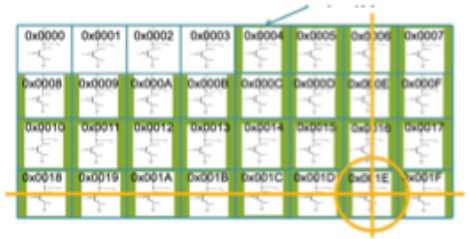
Complexity Analysis

How long will the algorithm take when run on inputs of different sizes:

- If it takes X seconds to process 1000 items, how long will it take to process twice as many (2000 items) or ten times as many (10,000 items)?

Generally looking for an order of magnitude estimate:

Constant time

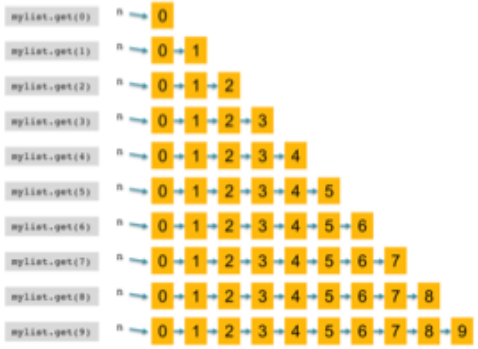


Accessing 1st or 1 billionth entry from an array takes same amount of time

Linear time

Takes 10 times longer to scan a list that has 10 times as many values

Quadradic time



Nested loops grows with the square of the list length

Also very important for space characterization:

Sometimes doubling the number of elements will more than double the amount of space needed

Find Max: Linear Search (1)

```
import java.text.NumberFormat;

public class ArrayFind {
  final static int MAXINT = 100000000;

  // return the biggest int in the array
  public static int findMaximum(int [] myarray){
   …
  }

  public static void main(String[] args) {
    if (args.length == 0) {
      System.out.println("USAGE: ArrayFind <array size>");
      return;
    }

    int arraysize = Integer.parseInt(args[0]);

    System.out.println("Scanning the array of size: ” +
                        NumberFormat.getInstance().format(arraysize));

    int [] myarray = new int[arraysize];

    ...
```

Find Max: Linear Search (2)

```
...

    int [] myarray = new int[arraysize];

    // initialize with random valuas
    for (int i = 0; i < myarray.length; i++) {
      int random = (int)(Math.random() * MAXINT);
      myarray[i] = random;
    }

    long startTime = System.nanoTime();
    int max = findMaximum(myarray);
    long endTime = System.nanoTime();
    long duration = endTime - startTime;

    System.out.println("The max is: " + max);
    System.out.println("Search took: " +
        NumberFormat.getInstance().format(duration) + " nanoseconds");
  }
}
```

FindMax Analysis

```
public static int findMaximum(int [] myarray) {
   int max = myarray[0];
   for (int i = 1; i < myarray.length; i++) {
     if (myarray[i] > max) {
       max = myarray[i];
     }
   }

   return max;
 }
```

```
$ java ArrayFind 10000000
Scanning the array of size: 10,000,000
The max is: 99999989
Search took: 11,666,963 nanoseconds
```

```
$ java ArrayFind 100000000
Scanning the array of size: 100,000,000
The max is: 99999999
Search took: 71,270,945 nanoseconds
```

Why isnt ArrayFind 100M exactly 10 times longer than ArrayFind 10M?

FindMax Analysis

```
public static int findMaximum(int [] myarray) {
   int max = myarray[0];
   for (int i = 1; i < myarray.length; i++) {
     if (myarray[i] > max) {
       max = myarray[i];
     }
   }

   return max;
 }
```

How many comparisons are done?

i < myarray.length	*n*
myarray[i] > max	*n*

C(n) = 2n

How many assignments are done (worst case)?

max = myarray[0]	*1*
for i =1; i < myarray.length; i++	*n*
val = myarray[i]	*n-1*
max = myarray[i]	*n-1*

A(n) = 1+ n + 2(n-1) = 3n-1

FindMax Analysis

```
public static int findMaximum(int [] myarray) {
   int max = myarray[0];
   for (int i = 1; i < myarray.length; i++) {
     if (myarray[i] > max) {
       max = myarray[i];
     }
   }

   return max;
 }
```

What is the total amount of work done?

T(n) = C(n) + A(n) = (2n) + (3n – 1) = 5n-1

Should we worry about the "-1"?

Nah, for sufficiently large inputs will make a tiny difference

Should we worry about the 5n?

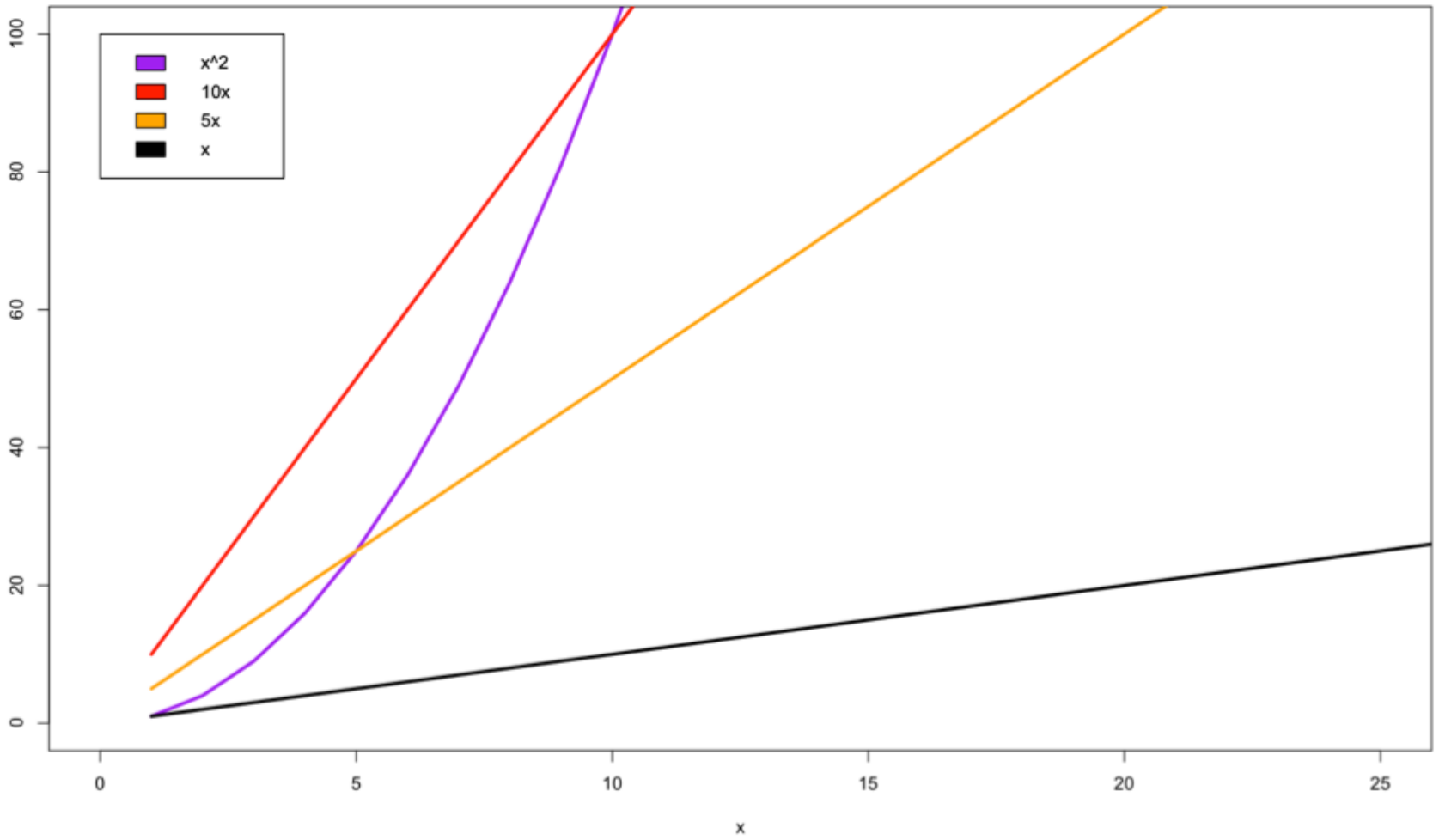
Nah, the runtime is linearly proportional to the length of the array

Big-O Notation

- ***Formally, algorithms that run in O(X) time means that the total number of steps (comparisons and assignments) is a polynomial whose largest term is X, aka asymptotic behavior***
  - ***f(x) ∈ O(g(x)) if there exists c > 0 (e.g., c = 1) and x_0 (e.g., x_0 = 5) such that f(x) ≤ cg(x) whenever x ≥ x_0***
    - $T(n) = 33$ => $O(1)$
    - $T(n) = 5n-2$ => $O(n)$
    - $T(n) = 37n^2 + 16n - 8$ => $O(n^2)$
    - $T(n) = 99n^3 + 12n^2 + 70000n + 2$ => $O(n^3)$
    - $T(n) = 127n \log(n) + \log(n) + 16$ => $O(n \lg n)$
    - $T(n) = 33 \log(n) + 8$ => $O(\lg n)$
    - $T(n) = 900*2^n + 12n^2 + 33n + 54$ => $O(2^n)$

- ***Informally, you can read Big-O(X) as "On the order of X"***
  - *O(1) => On the order of constant time*
  - *O(n) => On the order of linear time*
  - *$O(n^2)$ => On the order of quadratic time*
  - *$O(n^3)$ => On the order of cubic time*
  - *O(lg n) => On the order of logarithmic time*
  - *O(n lg n) => On the order of n log n time*

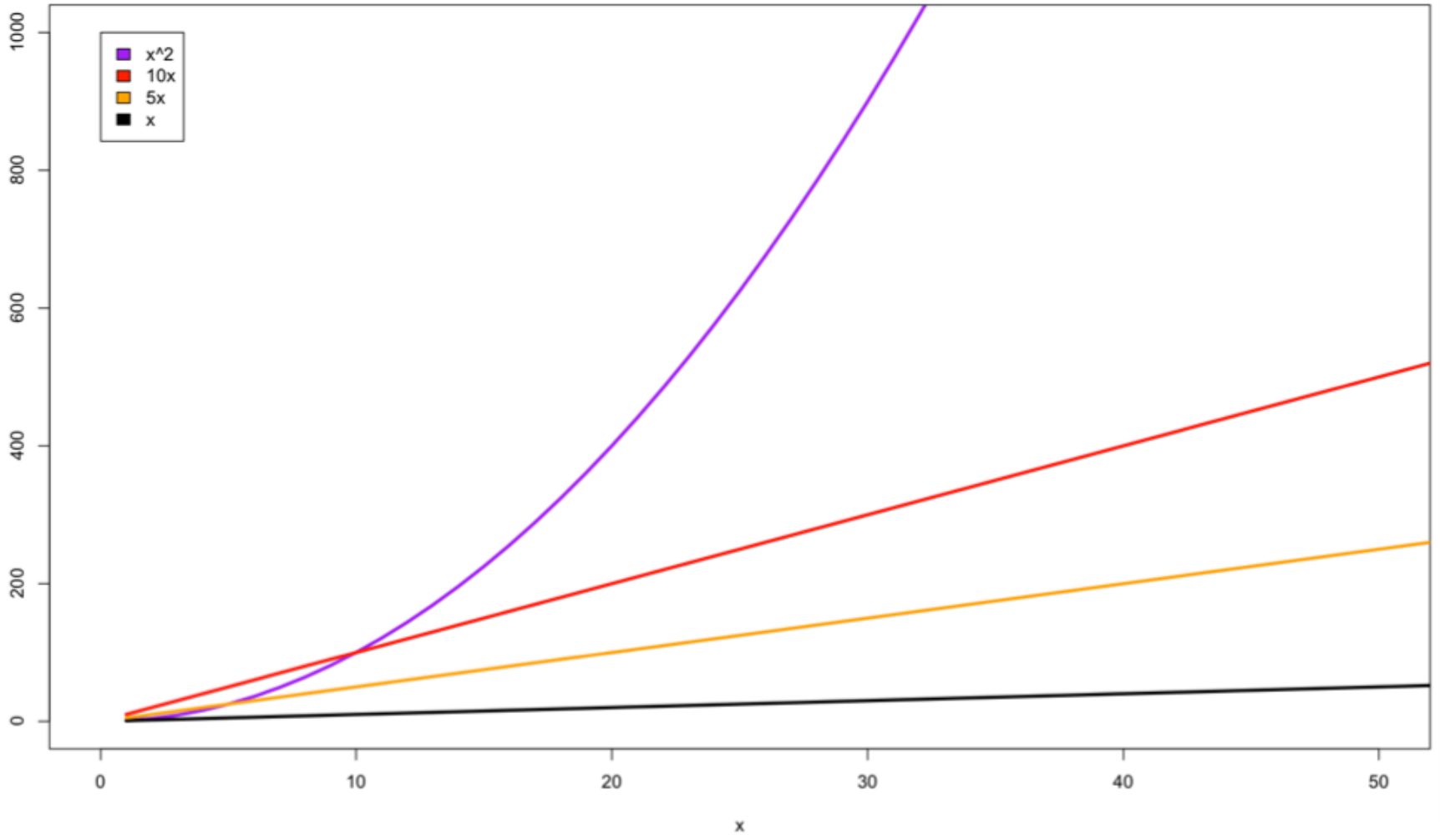
Growth of functions



A quadratic function isnt necessarily larger than a linear function for all possible inputs, but eventually will be

That largest polynomial term defines the Big-O complexity

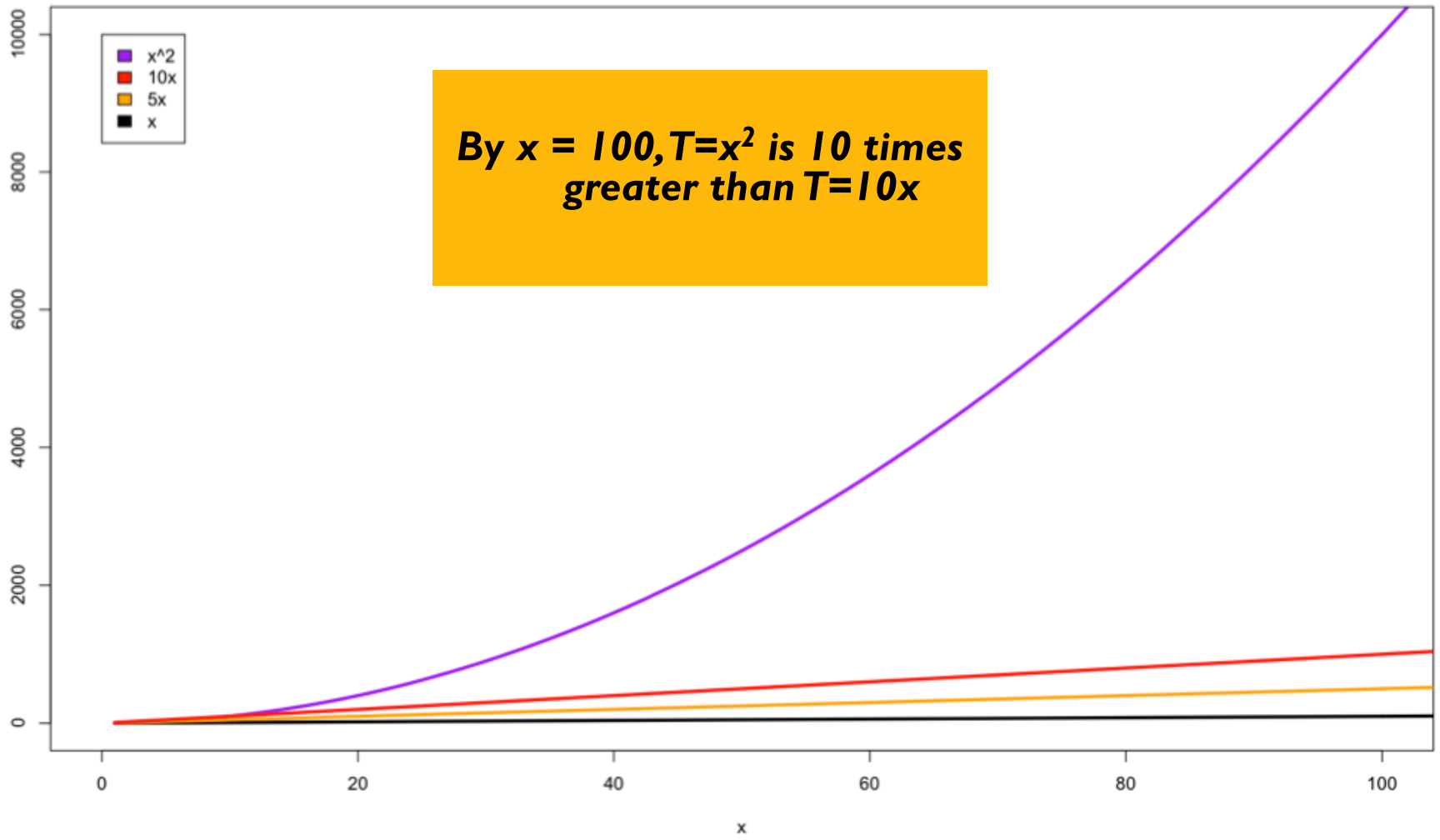
Growth of functions



A quadratic function isnt necessarily larger than a linear function for all possible inputs, but eventually will be

That largest polynomial term defines the Big-O complexity

Growth of functions



By x = 100, $T=x^2$ is 10 times greater than T=10x

A quadratic function isnt necessarily larger than a linear function for all possible inputs, but eventually will be

That largest polynomial term defines the Big-O complexity

Pop quiz!

```
public static int foo(int n) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
      sum = sum + i;
  }

  return sum;
}
```

What is foo(10)?

What is the complexity?

```
public static int bar(int n) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
      for (int k = 0; k < j; k++) {
        sum = sum + i + k + k;
      }
    }
  }

  return sum;
}
```

What is bar(10)?

What is the complexity?

Pop quiz!

```
public static int baz(int n) {
  if (n <= 0) {
    return 0;
  }

  int sum = 0;
  for (int i = 0; i < n; i++) {
    sum = sum + i;
  }

  return sum + baz(n/2) + baz(n/2-1);
}
```

What is baz(10)?

What is the complexity?

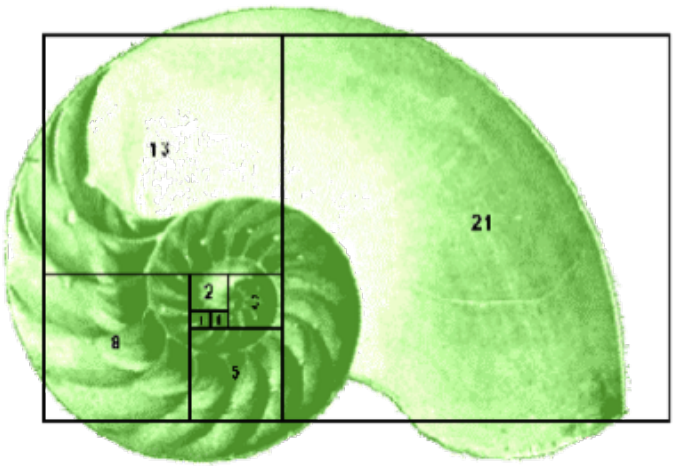
```
public static int buz(int n) {
  if (n <= 1) {
    return 1;
  }

  return buz(n-1) + buz(n-2);
}
```

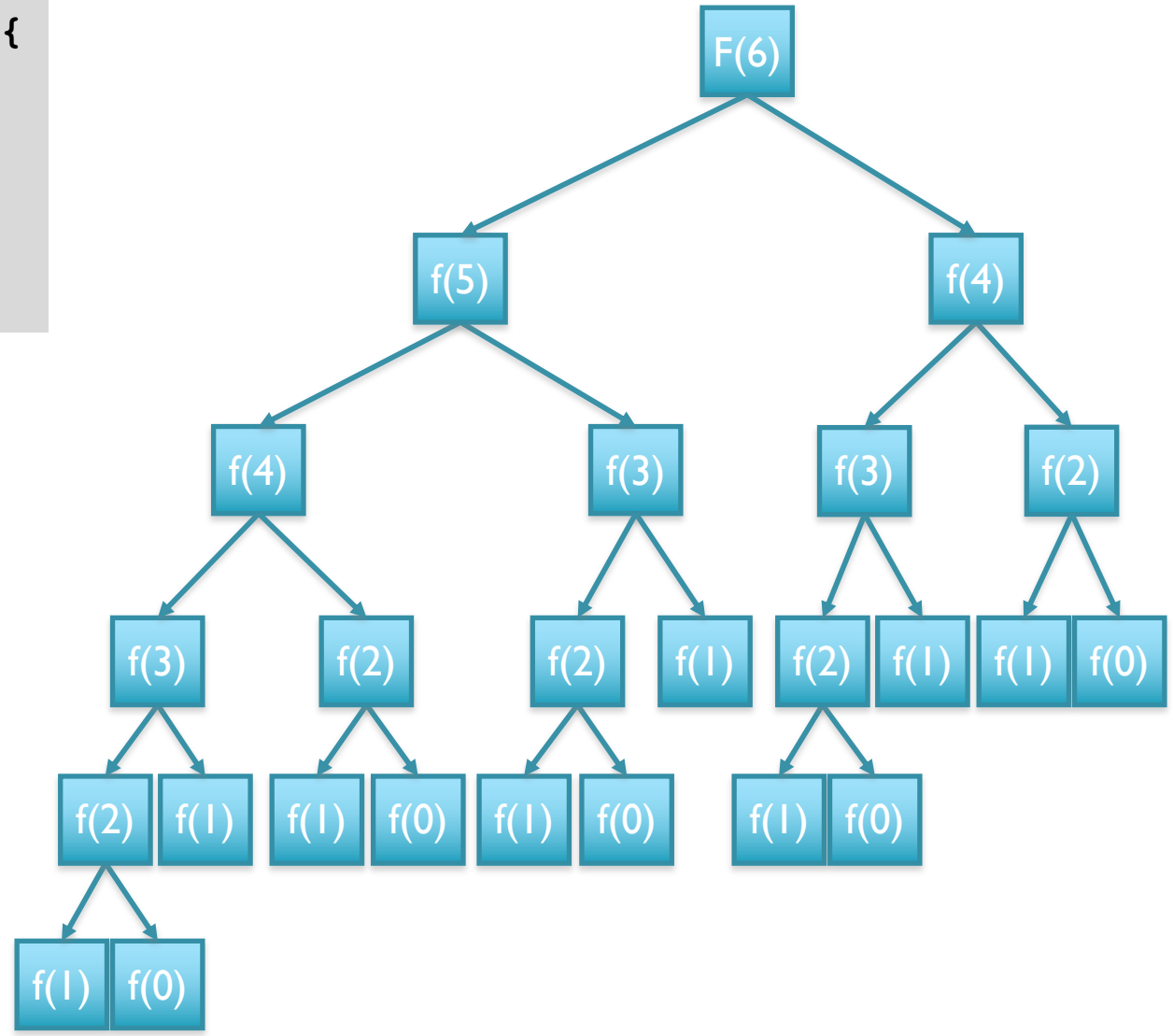
What is buz(10)?

What is the complexity?

Fibonacci Sequence

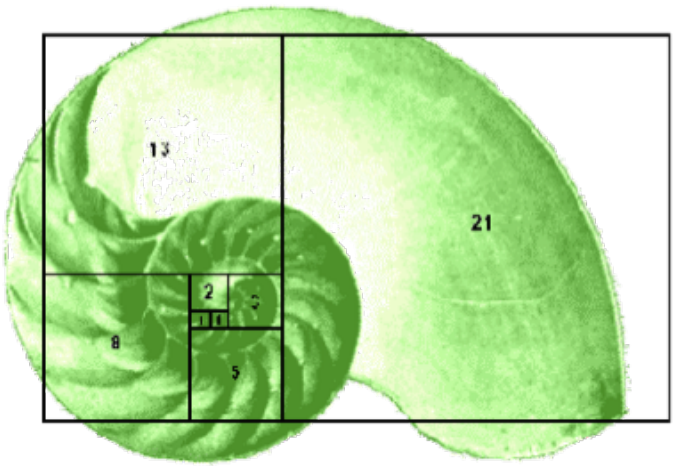
```
public static int fib(int n) {
  if (n <= 1) {
    return 1;
  }

  return buz(n-1) + buz(n-2);
}
```

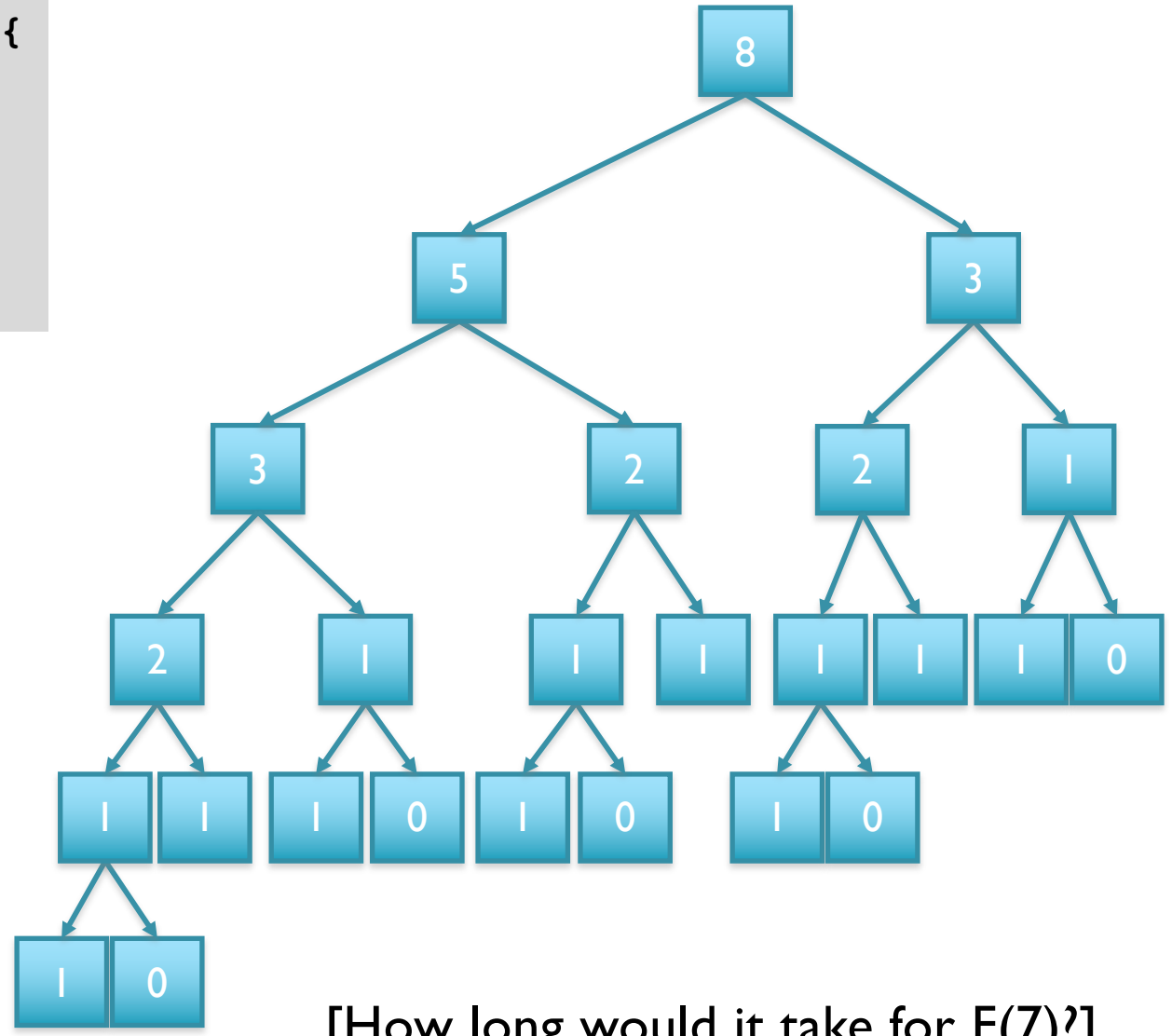




Fibonacci Sequence

```
public static int fib(int n) {
  if (n <= 1) {
    return 1;
  }

  return buz(n-1) + buz(n-2);
}
```



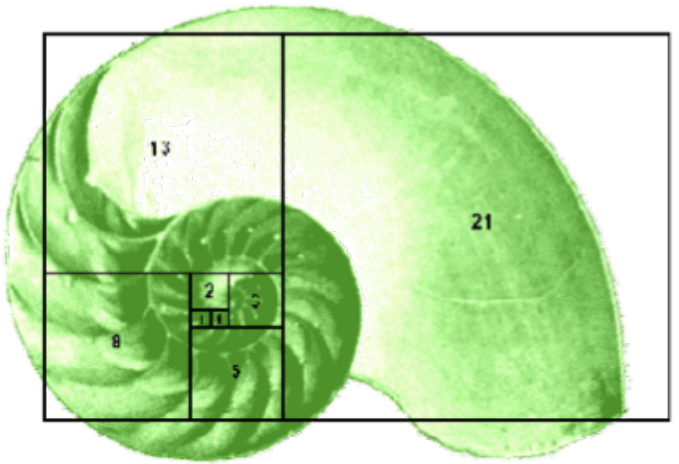


[How long would it take for F(7)?]
[What is the running time?]

Bottom-up Fibonacci Sequence

```
public static int fastbuz(int n) {
    int [] s = new int[n+1];
    s[0] = 1; s[1] = 1;

    for (int i = 2; i <= n; i++) {
        s[i] = s[i-1] + s[i-2];
    }

    return s[n];
 }
```

[How long will it take for F(7)?]
[What is the running time?]



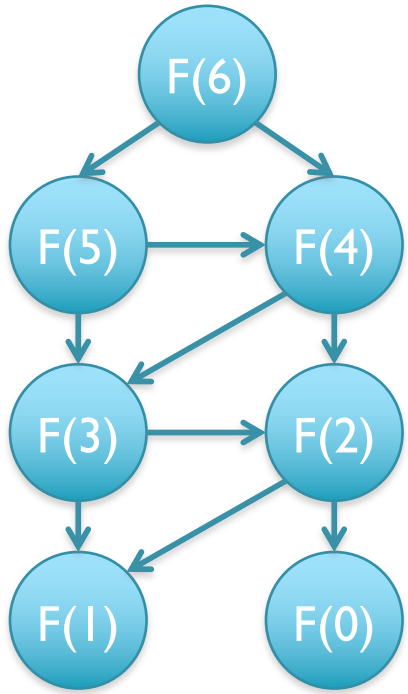
Fib vs FastFib

```
$ for i in `seq 1 50`;
  do echo $i; java Buz $i; done
1
Scanning the array of size: 1
The value is: 1
Search took: 3,515 nanoseconds
2
Scanning the array of size: 2
The value is: 2
Search took: 3,849 nanoseconds
3
Scanning the array of size: 3
The value is: 3
Search took: 4,034 nanoseconds
...
47
Scanning the array of size: 47
The value is: 512559680
Search took: 11,723,622,912 nanoseconds
48
Scanning the array of size: 48
The value is: -811192543
Search took: 19,283,637,425 nanoseconds
49
Scanning the array of size: 49
The value is: -298632863
Search took: 33,963,346,264 nanoseconds
50
Scanning the array of size: 50
The value is: -1109825406
Search took: 51,185,363,592 nanoseconds
```

```
$ for i in `seq 1 50`;
  do echo $i; java FastBuz $i; done
1
Scanning the array of size: 1
The value is: 1
Search took: 4,116 nanoseconds
2
Scanning the array of size: 2
The value is: 2
Search took: 4,286 nanoseconds
3
Scanning the array of size: 3
The value is: 3
Search took: 4,600 nanoseconds
...
47
Scanning the array of size: 47
The value is: 512559680
Search took: 9,140 nanoseconds
48
Scanning the array of size: 48
The value is: -811192543
Search took: 10,143 nanoseconds
49
Scanning the array of size: 49
The value is: -298632863
Search took: 9,212 nanoseconds
50
Scanning the array of size: 50
The value is: -1109825406
Search took: 9,662 nanoseconds
```

Dynamic Programming

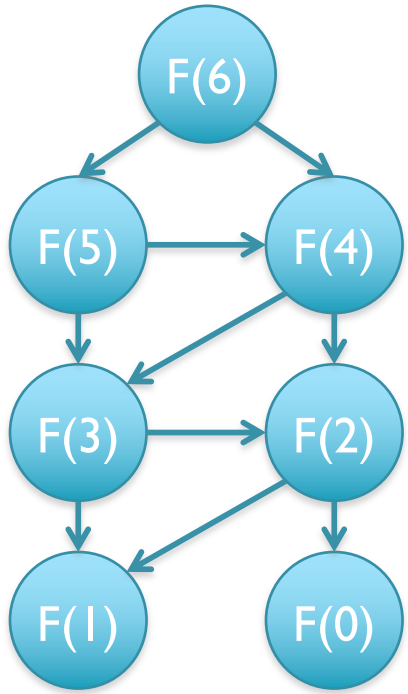
- General approach for solving (some) complex problems
  - When applicable, the method takes far less time than naive methods.
    - Polynomial time ($O(n)$ or $O(n^2)$ instead of exponential time ($O(2^n)$ or $O(3^n)$)
- Requirements:
  - **Overlapping subproblems**
  - **Optimal substructure**
- Applications:
  - Fibonacci
  - Longest Increasing Subsequence
  - Sequence alignment, Dynamic Time Warp, Viterbi



- Not applicable:
  - Traveling salesman problem, Clique finding, Subgraph isomorphism, …
  - The cheapest flight from airport A to airport B involves a single connection through airport C, but the cheapest flight from airport A to airport C involves a connection through some other airport D.

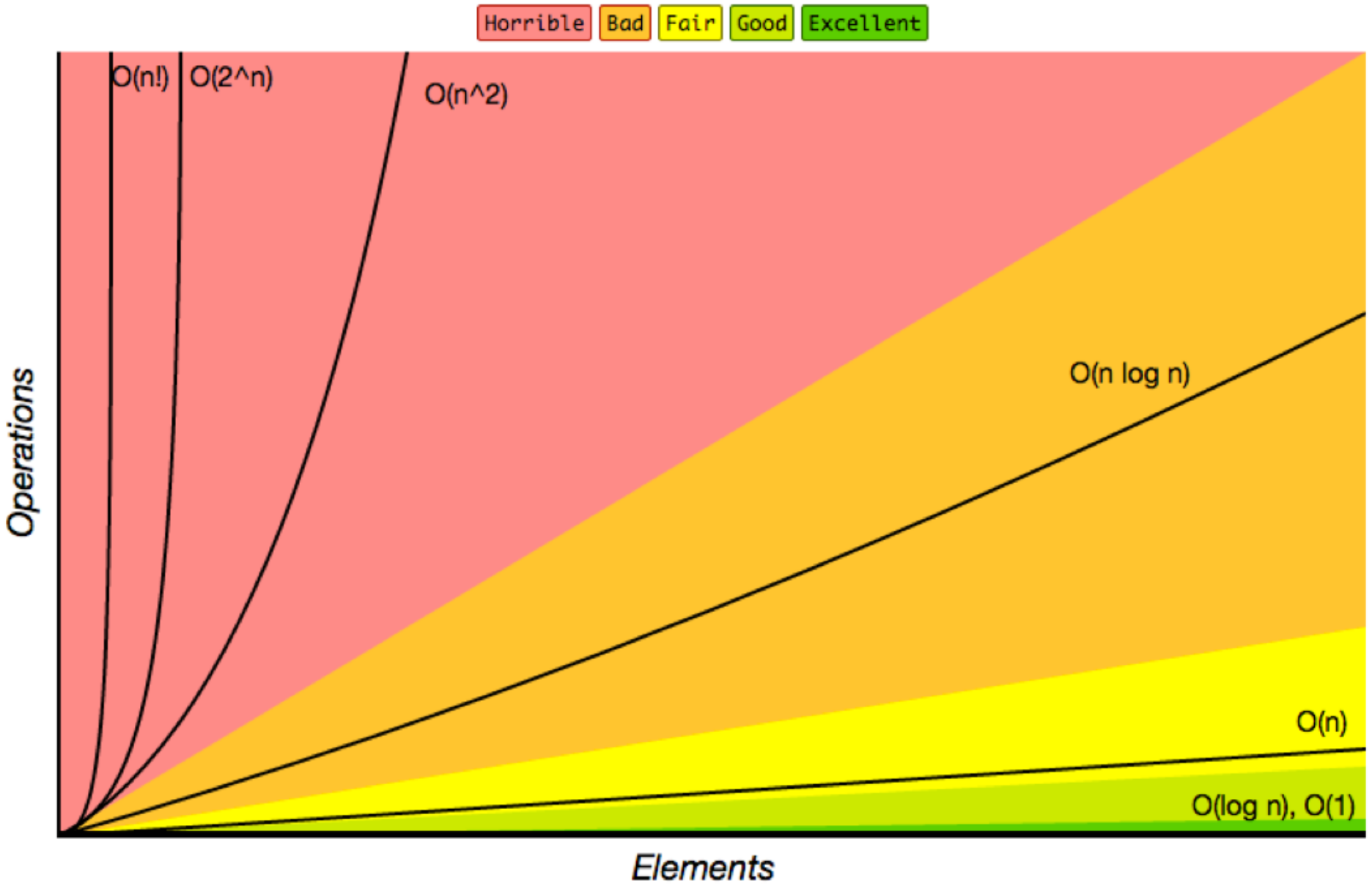
Dynamic Programming

- General approach for solving (some) complex problems
  - When applicable, the method takes far less time than naive methods.
    - Polynomial time (O(n) or $O(n^2)$) instead of exponential time ($O(2^n)$ or $O(3^n)$)
- Requirements:
  - **Overlapping subproblems**
  - **Optimal substructure**
- Applications:
  - Fibonacci
  - Longest Increasing Subsequence
  - Sequence alignment, Dynamic Time Warp, Viterbi
- Not applicable:
  - [illegible]
  - [illegible] to airport C involves a connection through some other airport D.



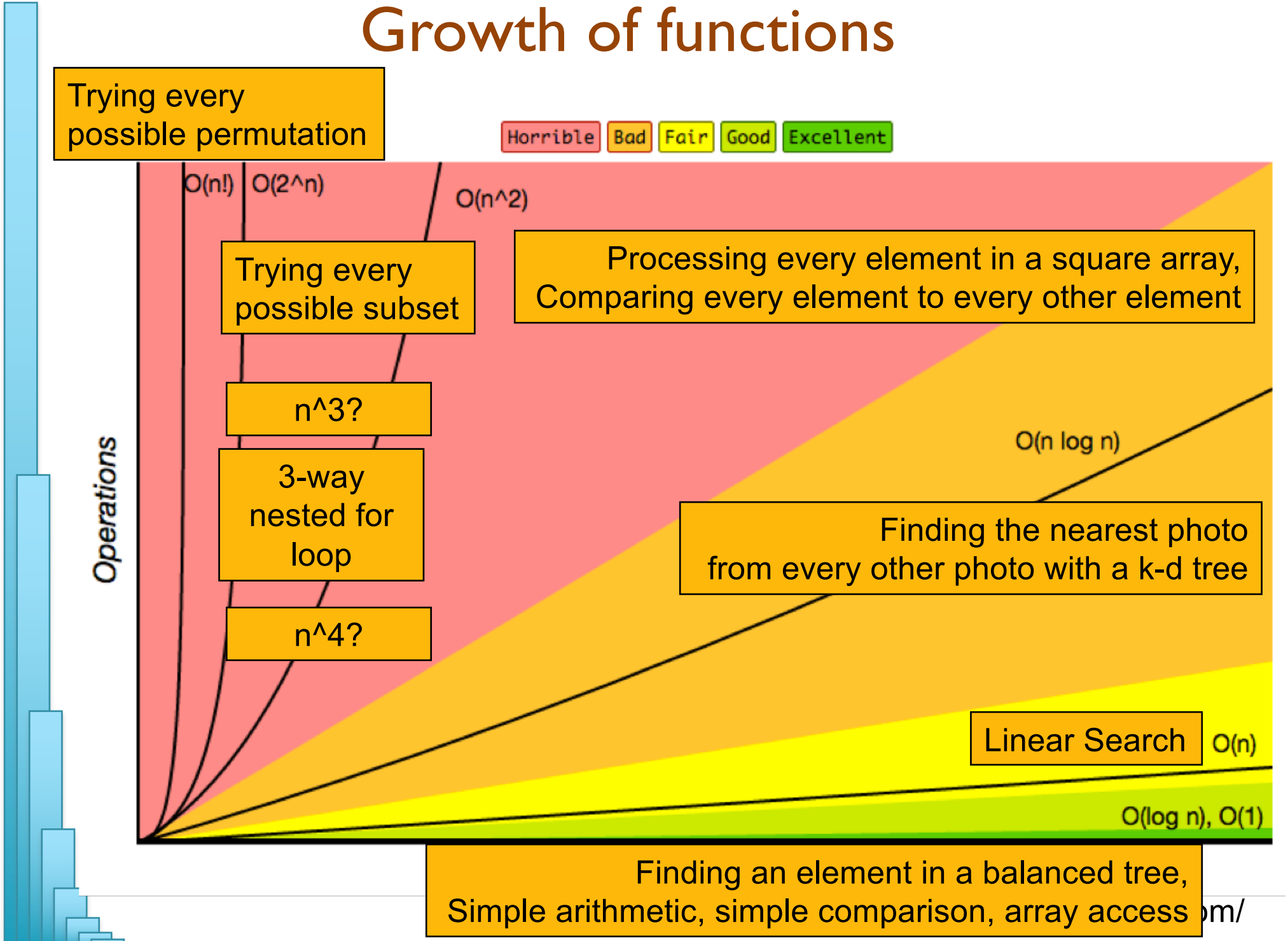
Sign up for algorithms!

Growth of functions



http://bigocheatsheet.com/

Growth of functions



Data Structure Complexities

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

http://bigocheatsheet.com/

Next Steps

1. Submit HW1
2. Work on HW2
3. Check on Piazza for tips & corrections!