CS 61BL Midterm 2 Review

Data Structures

Colleen Lewis, Kaushik Iyer, Jonathan Kotker, David Zeng, George Wang

Based on problems by Jonathan Shewchuk, Mike Clancy, Leonard Wei, Min Xu

Topics of Emphasis (not exhaustive)

- Linked Lists – 2 implementations
- Big-O Notation
- Amoeba Family Trees and Trees
- Binary Trees and Tree Iterators
- Search
- BSTs and TreeMaps
- Hash Tables
- Everything in MT1
- Upto Monday's Quiz

$$3^x \in O(x!)$$

Option A: True

Option B: False

Option C: 42

Option D: ??????????????????

Answer

$$3^x \in O(x!)$$

Option A: True

Option B: False

Option C: 42

Option D: ??????????????????

$$7x^2 + 45x + 9 \in O(x^2)$$

Option A: True

Option B: False

Option C: 42

Option D: ??????????????????

Answer

$$7x^2 + 45x + 9 \in O(x^2)$$

Option A: True

Option B: False

Option C: 42

Option D: ??????????????????

What is the Big-O run time of this piece of code?

```
public static void main (String [] a) {
        int i = 5;
        f (i);
        System.out.println("i=" + i);
 }
                  void f (int i) {
                        i = i * i;
                     }
```

Option A: O(1)

Option B: O(n) where n is a.length()

Option C: O(log(n)) where n is 5

Option D: O(f)

Answer

What is the Big-O run time of this piece of code?

```
public static void main (String [] a) {
        int i = 5;
        f (i);
        System.out.println("i=" + i);
 }
                  void static f (int i) {
                        i = i * i;
                     }
```

Option A: O(1) ←

Option B: O(n) where n is a.length()

Option C: O(log(n)) where n is 5

Option D: O(f)

What is the big-O runtime of this block of code?

```
  public static void g (int [] a) {
     for (int i = 0; i < a.length; i++) {
          f(a[i]);
     }
}
                  void static f(int s) {
                      for(int i = 0; i < s; i++) {
                         System.out.println(i);
                      }
```

Option A: O(1)

Option B: O(a.length)

Option C: O(a.length * max(a[i]))

Option D: O(a.length²)

Answer

What is the big-O runtime of this block of code?

```
  public static void g (int [] a) {
     for (int i = 0; i < a.length; i++) {
          f(a[i]);
     }
}
                  void static f(int s) {
                      for(int i = 0; i < s; i++) {
                         System.out.println(i);
                      }
```

Option A: O(1)

Option B: O(a.length)

Option C: O(a.length * max(a[i])) ←

Option D: O(a.length²)

What is the big-O runtime of this block of code?

```
  public static void main(String args) {
    for (int i = 0; i < a.length; i++) {
         for (int j = 0; j < 10; j++) {
              System.out.println(i+j);
         }
    }
  }
```

Option A: O(10 * a.length)

Option B: O(a.length²)

Option C: O(a.length)

Option D: ???

Answer

What is the big-O runtime of this block of code?

```
  public static void main(String args) {
    for (int i = 0; i < a.length; i++) {
         for (int j = 0; j < 10; j++) {
              System.out.println(i+j);
         }
    }
  }
```

Option A: O(10 * a.length) ← Correct, but unnecessarily specific

Option B: O(a.length²)

Option C: O(a.length) ←

Option D: ???

What is the big-O runtime of this block of code?

```
public static void main(String args) {
  for (int i = 0; i < a.length; i++) {
       for (int j = 0; j < i; j++) {
            System.out.println(i+j);
       }
  }
}
```

Option A: O(i * a.length)

Option B: O(a.length^2)

Option C: O(a.length)

Option D: ???

What is the big-O runtime of this block of code?

```
public static void main(String args) {
  for (int i = 0; i < a.length; i++) {
       for (int j = 0; j < i; j++) {
            System.out.println(i+j);
       }
  }
}
```

Option A: O(i * a.length)

Option B: O(a.length^2) ⬅

Option C: O(a.length)

Option D: ???

What is the Big-O runtime of f ?

```
public static void f(int[] a) {
   for (int i = 0; i < a.length; i++) {
        System.out.println(a[i]);
   }
}
```

Option A: O(a.length^2)

Option B: O(1)

Option C: O(a.length)

Option D: O(n)

What is the Big-O runtime of f ?

```
public static void f(int[] a) {
   for (int i = 0; i < a; i++) {
        System.out.println(a[i]);
   }
}
```

Option A: O(a.length^2)

Option B: O(1)

Option C: O(a.length) ⬅

Option D: O(n)

What is the Big-O runtime of f ?

```
public static void f(LinkedList a) {
   for (int i = 0; i < a.length(); i++) {
        System.out.println(a.get(i));
   }
}
```

Option A: O(a.length^2)

Option B: O(1)

Option C: O(a.length)

Option D: O(n)

What is the Big-O runtime of f ?

```
public static void f(LinkedList a) {
   for (int i = 0; i < a.length(); i++) {
        System.out.println(a.get(i));
   }
}
```

Option A: O(a.length^2) ⬅

Option B: O(1)

Option C: O(a.length)

Option D: O(n)

Data Structures Matter!

What is the Big-O runtime of f ?

```
public static void (List a) {
   for (ListNode point = a.myHead;
    point != null; point = point.myRest) {
        System.out.println(point.myFirst);
   }
}
```

Option A: $O(a.length^2)$

Option B: $O(1)$

Option C: $O(a.length)$

Option D: $O(n)$

Answer

What is the Big-O runtime of f ?

```
public static void (List a) {
   for (ListNode point = a.myHead;
    point != null; point = point.myRest) {
        System.out.println(point.myFirst);
   }
}
```

Option A: $O(a.length^2)$

Option B: $O(1)$

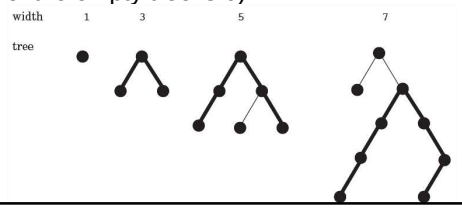
Option C: $O(a.length)$ ⟸

Option D: $O(n)$

Implementation Matters!

It's not all multiple choice ☺

- Consider the *width* of a tree defined as the maximal number of nodes on a path from one node in the tree to another (the width of the empty tree is 0).



- Write a BinaryTree method width() that returns the width of a binary tree.
- Assume that each tree node contains the height of the tree rooted at the node.
- The height of the empty tree is 0.

- Using the AmoebaFamily class you saw in lab:
- write a method that finds the closest common ancestor of two Amoebas in this Family given their names.

```
public class AmoebaFamily {
    private Amoeba myRoot;

    public class Amoeba {
        public Amoeba myParent;
        public ArrayList<Amoeba> myChildren;
        public String myName;

    }
}
```

We want to code the following method:

```
public String nameOfClosestCommonAncestor(String name1,
        String name2) {
    Amoeba f1 = myRoot.search(name1);
    Amoeba f2 = myRoot.search(name2);
    Amoeba ans = closestCommonAncestor(f1, f2);
    return ans.myName;
}
```

Implement the search function in the Amoeba class. Required runtime: O(number of nodes under this node

```
public Amoeba search (String name) {
   //looks through this node and all its children and descendants
   //for a node with myName equal to name
   //returns this node if found, null if no such node exists
   return null;
   }
```

- Implement the closestCommonAncestor in AmoebaFamily. Required runtime: O(depth of f1 + depth of f2)

```
private static Amoeba closestCommonAncestor
                    (Amoeba f1, Amoeba f2) {
    // looks for the closest common ancestor
    // of f1 and f2.
    // The closest common ancestor of two nodes
    // f1 and f2 is the closest node the furthest
    // from the root that's an ancestor of both
    // f1 and f2.
    return null;
    }
```

Coding Break!

- You have an array of int values sorted in ascending order.
- Using the insert method coded in lab for BSTs (adds new values only as leaves), we insert values one by one.
- What's the depth of the BST created?
- What would the depth be if the array was sorted in descending order?
- How long would it take to find out if the array contained a value?

- How long would it take to find out if the BST contained a value?
- Write code that would ensure that the BST was maximally balanced by modifying how you insert the values from the sorted array.

```
public BST insert(int[] sortedarray) {
    // return a new maximally balanced
    // BST created using the values in
    // sortedarray

    return null;
}
```

Binary Search Trees

What is the running time of looking for an element in a binary search tree?

Option A: O(1)

Option B: O(log n), n being the number of nodes in the tree.

Option C: O(log n), n being the height.

Option D: O(log n), n being the element.

Binary Search Trees

What is the running time of looking for an element in a binary search tree?

- Option A: O(1)
- Option B: O(log n), n being the number of nodes in the tree.
- Option C: O(log n), n being the height.
- Option D: O(log n), n being the element.

Linked Lists with abstract ListNode

```
14 public void test(ListNode x) {
15     System.out.println(x.first());
16 }
```

- Option A: Whatever x.myFirst is
- Option B: Exception
- Option C: Exception OR whatever x.myFirst is
- Option D: null

Answer Linked Lists with abstract ListNode

```
14 public void test(ListNode x) {
15     System.out.println(x.first());
16 }
```

- Option A: x.myFirst
- Option B: Exception
- Option C: Exception OR Whatever x.myFirst is
- Option D: null

Binary Search Trees

What is the running time of looking for an element in a binary search tree?

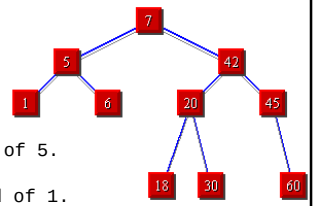
- Option A: O(1)
- Option B: O(log n), n being the number of nodes in the tree.
- Option C: O(log n), n being the height.
- Option D: O(log n), n being the element.

Answer Binary Search Trees

What is the running time of looking for an element in a binary search tree?

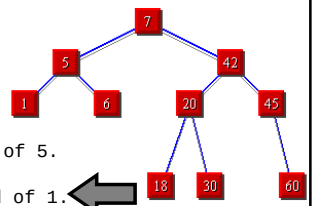
- Option A: O(1)
- Option B: O(log n), n being the number of nodes in the tree.
- Option C: O(log n), n being the height.
- Option D: O(log n), n being the element.

Where will the number **4** go?



- Option A: Left child of 5.
- Option B: Right child of 1.
- Option C: Left child of 6.
- Option D: Left child of 7.

Where will the number **4** go?



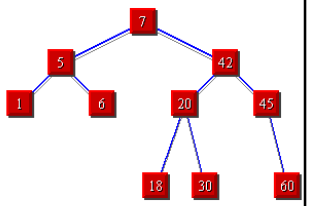
Option A: Left child of 5.

Option B: Right child of 1.

Option C: Left child of 6.

Option D: Left child of 7.

When **1** is removed, which node will take its place?



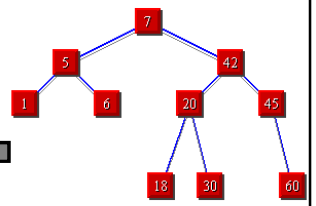
Option A: None.

Option B: 5.

Option C: 6.

Option D: 7.

When **1** is removed, which node will take its place?



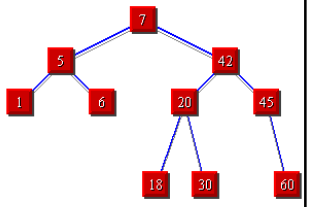
Option A: None.

Option B: 5.

Option C: 6.

Option D: 7.

When **7** is removed, which node will take its place?



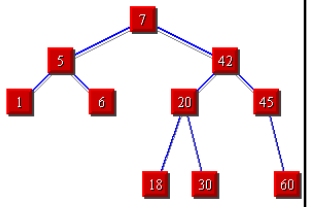
Option A: 5.

Option B: 6.

Option C: 18.

Option D: 42.

When **7** is removed, which node will take its place?



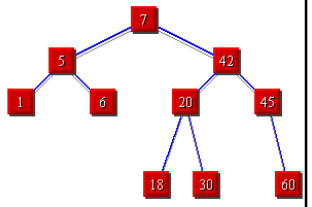
Option A: 5.

Option B: 6.

Option C: 18.

Option D: 42.

When **42** is removed, which node will take its place?



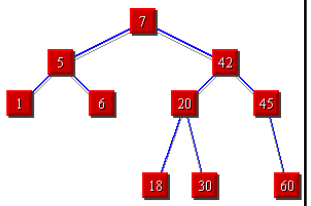
Option A: 18.

Option B: 20.

Option C: 30.

Option D: 45.

When **42** is removed, which node will take its place?



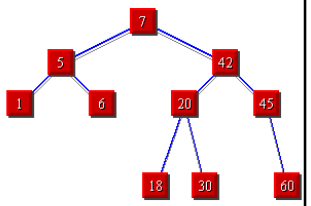
Option A: 18.

Option B: 20.

Option C: 30.

Option D: 45. ⬅

When **42** is removed, which node will take **45**'s place, if any?



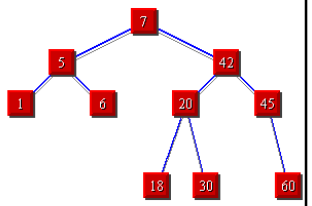
Option A: None.

Option B: 20.

Option C: 30.

Option D: 60.

When **42** is removed, which node will take **45**'s place, if any?



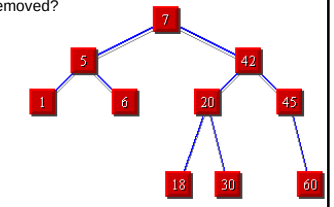
Option A: None.

Option B: 20.

Option C: 30.

Option D: 60. ⬅

Which node, when removed, will be placed back in the same position as it was before it was removed?



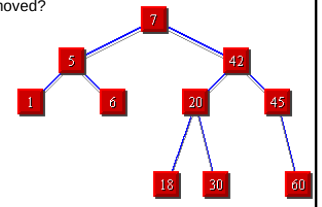
Option A: 5.

Option B: 20.

Option C: 30.

Option D: 60.

Which node, when removed, will be placed back in the same position as it was before it was removed?



Option A: 5.

Option B: 20.

Option C: 30. ⬅

Option D: 60. ⬅

Hash Codes

Suppose we were to use numbers that can only be even, all numbers appearing with equal probability. Which of the following hashing scheme among the following choices is not great?

Option A: The number itself.

Option B: The number divided by two.

Option C: Sum of the digits of the number.

Option D: Difference between the digits of the number.

Hash Codes

Suppose we were to use numbers that can only be even, all numbers appearing with equal probability. Which of the following hashing scheme among the following choices is not great?

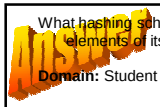
- Option A: The number itself. ⇐
- Option B: The number divided by two.
- Option C: Sum of the digits of the number.
- Option D: Difference between the digits of the number.

What hashing scheme among the following would evenly spread the elements of its domain the best?

Domain: Student Names

- Option A: CalNET ID.
- Option B: Number of characters in String.
- Option C: First letter of name.
- Option D: Sum of ASCII codes of each character.

Courtesy: Min Xu



What hashing scheme among the following would evenly spread the elements of its domain the best?

Domain: Student Names

- Option A: CalNET ID. ⇐
- Option B: Number of characters in String.
- Option C: First letter of name.
- Option D: Sum of ASCII codes of each character.

Courtesy: Min Xu

Defective Hash Functions

You're working with Louis Reasoner on a CS project and he proposed the following hash functions. Some of them are incorrect and some of them are inefficient; all of them are defective. Explain why. Assume p is a prime number and is the size of the hash table.

h : String $\rightarrow H$. $h(s)$ = memory address of String s (mod p).

h : List $\rightarrow H$. Suppose the List contains all distinct values and that a_i is an element of the List with index i. $h(a_i) = i \mod p$.

$h : \mathbb{N} \rightarrow H$. Let $n \in \mathbb{N}$, and $h(n) = n! \mod p$.

$h : \mathbb{Z} \rightarrow H$. Let $h(n) = n^{p-1} \mod p$.

Courtesy: Min Xu

	Add at front	Get at pos	Contains?	Remove			Max
				Remove from front	Remove from middle	Remove from end	
Arrays							
Linked Lists							
Trees							
Binary Trees							
Binary Search Trees							
Tree Map							