CSE 12 Week 10 Discussion

6-4-21

Focus: Final Review

Reminders

- PA8 **(open!)** due Friday, June 4th @ 11:59 PM
    - No resubmission
- PA7 Resubmission due Friday, June 4th @ 11:59 PM

Midterm 1

Midterm 1

Topics Covered

- Testing
- Interfaces
- Generics
- Lists: ArrayLists, LinkedLists
- Stacks and Queues
- BFS & DFS

Midterm 2

Midterm 2

Topics Covered

- Time Complexities
- Sorting
- Maps and HashTables

Midterm 2 Review Discussion Slides

Beyond

Beyond

Topics Covered

- BSTs
- Heaps
- Improving Lists
- Iterators
- Streams

Final Review Questions

Practice Final Review - Question 1

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  //MISSING CODE
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Complete addAtEnd(e) method in the case when tail is not null...

Clicker Q - What would be the best first step in adding a node to the end?

A. Change pointer of tail to be tail.next
B. Create a temp node to store a new node data structure
C. Set tail to head
D. Create a new node for tail.next

Practice Final Review - Question 1

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  //MISSING CODE
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Complete addAtEnd(e) method in the case when tail is not null...

Clicker Q - What would be the best first step in adding a node to the end?

A. Change pointer of tail to be tail.next
B. Create a temp node to store a new node data structure
C. Set tail to head
D. Create a new node for tail.next

Practice Final Review - Question 1

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  //MISSING CODE
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Complete addAtEnd(e) method in the case when tail is not null...

Clicker Q - What would be the best next step in adding a node to the end?

A. Change pointer of tail to be tail.next
B. Create a temp node to store a new node data structure
C. Set tail to head
D. Create a new node for tail.next

Practice Final Review - Question 1

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  //MISSING CODE
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Complete addAtEnd(e) method in the case when tail is not null...

Clicker Q - What would be the best next step in adding a node to the end?

A. **Change pointer of tail to be tail.next**
B. Create a temp node to store a new node data structure
C. Set tail to head
D. Create a new node for tail.next

Practice Final Review - Question 1

```
CLASS MyLinkedList
    DEFINE initialize MyLinkedList ()
        head = nil
        tail = nil
    ENDEF
    DEFINE addAtEnd (e)
        IF tail == nil THEN
            head = tail = new MyListNode(e)
        ELSE
            tail.next = new MyListNode(e)
            tail = tail.next
        ENDIF
    ENDEF
    DEFINE isEmpty ()
        RETURN head == nil
    ENDEF
ENDCLASS
CLASS MyListNode
    DEFINE initialize MyListNode (node_item)
        item = node_item
        next = nil
    ENDEF
ENDCLASS
```

Complete addAtEnd(e) method in the case when tail is not null...

tail.next = new MyListNode(e)
tail = tail.next

Practice Final Review - Question 2

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  tail.next = new MyListNode(e)
                  tail = tail.next
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Check if the following operation runs faster on the implementation to the left versus an implementation with reference to the head only:

Returns TRUE if List contains an element?

A. Runs faster on impl to left
B. Does not run faster on impl to left

Practice Final Review - Question 2

```
CLASS MyLinkedList
    DEFINE initialize MyLinkedList ()
        head = nil
        tail = nil
    ENDEF
    DEFINE addAtEnd (e)
        IF tail == nil THEN
            head = tail = new MyListNode(e)
        ELSE
            tail.next = new MyListNode(e)
            tail = tail.next
        ENDIF
    ENDEF
    DEFINE isEmpty ()
        RETURN head == nil
    ENDEF
ENDCLASS
CLASS MyListNode
    DEFINE initialize MyListNode (node_item)
        item = node_item
        next = nil
    ENDEF
ENDCLASS
```

A. Runs faster on impl to left

B. Does not run faster on impl to left

Still need to go through entire list to find an element at worst case - O(n)

Unpredictable for where an entry could be due to lack of ordering in LinkedList

Practice Final Review - Question 2

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  tail.next = new MyListNode(e)
                  tail = tail.next
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Check if the following operation runs faster on the implementation to the left versus an implementation with reference to the head only:

Removes the last element from LinkedList?

A. Runs faster on impl to left
B. Does not run faster on impl to left

Practice Final Review - Question 2

```
CLASS MyLinkedList
    DEFINE initialize MyLinkedList ()
        head = nil
        tail = nil
    ENDEF
    DEFINE addAtEnd (e)
        IF tail == nil THEN
            head = tail = new MyListNode(e)
        ELSE
            tail.next = new MyListNode(e)
            tail = tail.next
        ENDIF
    ENDEF
    DEFINE isEmpty ()
        RETURN head == nil
    ENDEF
ENDCLASS
CLASS MyListNode
    DEFINE initialize MyListNode (node_item)
        item = node_item
        next = nil
    ENDEF
ENDCLASS
```

A. Runs faster on impl to left

B. Does not run faster on impl to left

Even with references to both head and tail, need to go through entire list again to find and remove tail since there is no previous field

Practice Final Review - Question 2

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  tail.next = new MyListNode(e)
                  tail = tail.next
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Check if the following operation runs faster on the implementation to the left versus an implementation with reference to the head only:

Add given element to front of list?

A. Runs faster on impl to left
B. Does not run faster on impl to left

Practice Final Review - Question 2

```
CLASS MyLinkedList
    DEFINE initialize MyLinkedList ()
        head = nil
        tail = nil
    ENDEF
    DEFINE addAtEnd (e)
        IF tail == nil THEN
            head = tail = new MyListNode(e)
        ELSE
            tail.next = new MyListNode(e)
            tail = tail.next
        ENDIF
    ENDEF
    DEFINE isEmpty ()
        RETURN head == nil
    ENDEF
ENDCLASS
CLASS MyListNode
    DEFINE initialize MyListNode (node_item)
        item = node_item
        next = nil
    ENDEF
ENDCLASS
```

A. Runs faster on impl to left

B. Does not run faster on impl to left

Makes no difference having a tail node since you just need a head node to add new node to front

Practice Final Review - Question 2

```
CLASS MyLinkedList
      DEFINE initialize MyLinkedList ()
            head = nil
            tail = nil
      ENDEF
      DEFINE addAtEnd (e)
            IF tail == nil THEN
                  head = tail = new MyListNode(e)
            ELSE
                  tail.next = new MyListNode(e)
                  tail = tail.next
            ENDIF
      ENDEF
      DEFINE isEmpty ()
            RETURN head == nil
      ENDEF
ENDCLASS
CLASS MyListNode
      DEFINE initialize MyListNode (node_item)
            item = node_item
            next = nil
      ENDEF
ENDCLASS
```

Check if the following operation runs faster on the implementation to the left versus an implementation with reference to the head only:

Add given element to end of list?

A. Runs faster on impl to left
B. Does not run faster on impl to left

Practice Final Review - Question 2

```
CLASS MyLinkedList
    DEFINE initialize MyLinkedList ()
        head = nil
        tail = nil
    ENDEF
    DEFINE addAtEnd (e)
        IF tail == nil THEN
            head = tail = new MyListNode(e)
        ELSE
            tail.next = new MyListNode(e)
            tail = tail.next
        ENDIF
    ENDEF
    DEFINE isEmpty ()
        RETURN head == nil
    ENDEF
ENDCLASS
CLASS MyListNode
    DEFINE initialize MyListNode (node_item)
        item = node_item
        next = nil
    ENDEF
ENDCLASS
```

A. **Runs faster on impl to left**

B. Does not run faster on impl to left

Having a tail node gives you easy access to the end of the list. We do not have to iterate through the all entries.

Practice Final Review - Question 3

Which methods have a faster runtime when implemented with a doubly linked list instead of a singly linked list?

(recall: singly linked list only has reference to head and next; DLL has reference to both head and tail, next and previous)

A. addAtEnd(e) - adds e to the end of the list
B. removeAtEnd() - removes and returns last element
C. addAtStart(e) - adds e to the start of list
D. A and B
E. All the above methods are faster when implemented with a DDL

Practice Final Review - Question 3

Which methods have a faster runtime when implemented with a doubly linked list instead of a singly linked list?

(recall: singly linked list only has reference to head and next; DLL has reference to both head and tail, next and previous)

A. addAtEnd(e) - adds e to the end of the list
B. removeAtEnd() - removes and returns last element
C. addAtStart(e) - adds e to the start of list
D. A and B
E. All the above methods are faster when implemented with a DDL

Practice Final Review - Question 3

Using a singly linked list, which of the following pairs have different asymptotic runtimes?

A. addAtEnd(e), removeAtEnd()
B. get(ind), contains(e)
C. addAtEnd(e), addAtStart(e)
D. removeAtStart(), addAtStart(e)

Practice Final Review - Question 3

Using a singly linked list, which of the following pairs have different asymptotic runtimes?

A. `addAtEnd(e), removeAtEnd()`
B. `get(ind), contains(e)`
C. `addAtEnd(e), addAtStart(e)`
D. `removeAtStart(), addAtStart(e)`

Practice Final Review - Question 3

Using a singly linked list, which of the following pairs have different asymptotic runtimes?

A. addAtEnd(e), removeAtEnd() - both are O(n)
B. get(ind), contains(e) - both are O(n)
C. addAtEnd(e), addAtStart(e) - O(n), O(1)
D. removeAtStart(), addAtStart(e) - both are O(1)

Practice Final Review - Question 3

Recall the question: How do we differentiate a singly linked list from a doubly linked list using only the time measurements of list operations?

> **addAtEnd(e), addAtStart(e) - O(n), O(1) using singly linked list**
>
> this implies addAtEnd(e) should take longer than addAtStart() in general.

Note that addAtEnd(e) and removeAtEnd are O(1) when implemented with a DLL. Therefore if we measure the time of the above operations we expect to get similar time durations.

Practice Final Review - Question 4

What is the time complexity to retrieve an element from a LinkedList?

A. N^2
B. N log N
C. N
D. log N
E. Constant Time

Practice Final Review - Question 4

What is the time complexity to retrieve an element from a LinkedList?

A. N^2
B. N log N
C. N
D. log N
E. Constant Time

Practice Final Review - Question 5

What is the definition of a fully balanced tree?

A. The left and right subtrees’ heights of a tree differ by at most one
B. The left and right subtrees’ total number of nodes differ by at most one
C. The left and right subtrees’ must have the same height

Practice Final Review - Question 5

What is the definition of a fully balanced tree?

A. **The left and right subtrees’ heights of a tree differ by at most one**
B. The left and right subtrees’ total number of nodes differ by at most one
C. The left and right subtrees’ must have the same height

Practice Final Review - Question 5

Which of the following for the first five insertions (in the order given) will produce a fully balanced tree if inserting integers 1 to 1023?

A. 1, 2, 3, 4, 5
B. 512, 256, 768, 640, 384
C. 512, 511, 513, 510, 514
D. 512, 768, 640, 896, 256
E. 1023, 512, 768, 256, 640

Practice Final Review - Question 5

Which of the following for the first five insertions (in the order given) will produce a fully balanced tree if inserting integers 1 to 1023?

A. 1, 2, 3, 4, 5
B. 512, 256, 768, 640, 384
C. 512, 511, 513, 510, 514
D. 512, 768, 640, 896, 256
E. 1023, 512, 768, 256, 640

Practice Final Review - Question 6

What does the following method do?

```
mystery(value)
     current = head
     temp = nil
     WHILE current != nil AND current.item != value DO
     temp = current
     current = current.next
     ENDWHILE
RETURN temp
```

A. returns node before node containing value, or nil if value is in head, or last node if value is not in any nodes
B. returns the node containing value or nil if value not found
C. returns the node containing value or the last node if value not found
D. returns the node before node containing value, or nil if value was not found in any node

Practice Final Review - Question 6

What does the following method do?

```
mystery(value)
     current = head
     temp = nil
     WHILE current != nil AND current.item != value DO
     temp = current
     current = current.next
     ENDWHILE
RETURN temp
```

A. **returns node before node containing value, or nil if value is in head, or last node if value is not in any nodes**
B. returns the node containing value or nil if value not found
C. returns the node containing value or the last node if value not found
D. returns the node before node containing value, or nil if value was not found in any node

Practice Final Review - Question 7

How do we check if a node is a leaf node?

A. Check if it is null
B. Check if its left and right fields are null
C. Check if its height is greater than 1

Practice Final Review - Question 7

How do we check if a node is a leaf node?

A. Check if it is null
B. Check if its left and right fields are null
C. Check if its height is greater than 1

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

How can we break this down into subproblems to solve recursively?

> visit the children of each node and return the value of the node if it is a leaf

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

What should the base case be?

A. `if (node == nil) return 1;`
B. `if (node == nil) return 0;`
C. `if (node.left == nil AND node.right == nil) return 0;`
D. `if (node.left != nil AND node.right != nil) return 1;`

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

What should the base case be?

A. if (node == nil) return 1;
B. if (node == nil) return 0;
C. if (node.left == nil AND node.right == nil) return 0;
D. if (node.left != nil AND node.right != nil) return 1;

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    IF node.left == nil AND node.right == nil
        RETURN node.item
    IF node.left != nil
        RETURN node.item + sumLeaves(node.left)
    if node.right != nil
        RETURN node.item + sumLeaves(node.right)
```

A. Yes
B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    IF node.left == nil AND node.right == nil
        RETURN node.item
    IF node.left != nil
        RETURN node.item + sumLeaves(node.left)
    if node.right != nil
        RETURN node.item + sumLeaves(node.right)
```

A. Yes

B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    IF node == nil
        RETURN 0
    ELSE IF node.left == nil AND node.right == nil
        RETURN node.item
    ELSE
        RETURN sumLeaves(node.left) + sumLeaves(node.right)
```

A. Yes
B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    IF node == nil
        RETURN 0
    ELSE IF node.left == nil AND node.right == nil
        RETURN node.item
    ELSE
        RETURN sumLeaves(node.left) + sumLeaves(node.right)
```

A. **Yes** ✓
B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    val = 0
    WHILE node != nill
        IF node.left == nil AND node.right == nil
            val = val + node.item
        IF node.left != nil
            node = node.left
        ELSE IF node.right != nil
            node = node.right
    RETURN val
```

A. Yes
B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    val = 0
    WHILE node != nill
        IF node.left == nil AND node.right == nil
            val = val + node.item
        IF node.left != nil
            node = node.left
        ELSE IF node.right != nil
            node = node.right
    RETURN val
```

A. Yes
B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    val = 0
    IF node.left == nil AND node.right == nil
        val = val + node.item
    IF node.left != nil
        sumLeaves(node.left)
    IF node.right != nil
        sumLeaves(node.right)
    RETURN val
```

A. Yes
B. No

Practice Final Review - Question 7

sumLeaves(node) - a method that returns the sum of values of all leaf nodes

Does the following implementation work?

```
sumLeaves(node):
    val = 0
    IF node.left == nil AND node.right == nil
        val = val + node.item
    IF node.left != nil
        sumLeaves(node.left)
    IF node.right != nil
        sumLeaves(node.right)
    RETURN val
```

A. Yes
B. No

Practice Final Review - Question 8

We have a program that stores a list of Strings and the user is permitted to access the string at a given position in the list and can make as many accesses as needed. N is the number of strings in this list. Which data structure should we use?

A. ArrayList
B. Sorted DoublyLinkedList
C. Unsorted DoublyLinkedList

Practice Final Review - Question 8

We have a program that stores a list of Strings and the user is permitted to access the string at a given position in the list and can make as many accesses as needed. N is the number of strings in this list. Which data structure should we use?

A. **ArrayList**
B. Sorted DoublyLinkedList
C. Unsorted DoublyLinkedList

Practice Final Review - Question 8

Why would we use an ArrayList to store a list of Strings over a DoublyLinkedList?

A. It guarantees constant-time access
B. It guarantees access time proportional to logN using binary search

Practice Final Review - Question 8

Why would we use an ArrayList to store a list of Strings over a DoublyLinkedList?

A. **It guarantees constant-time access**
B. It guarantees access time proportional to logN using binary search

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Can the root of T1 hold the median of T1?

A. Yes
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Can the root of T1 hold the median of T1?

A. Yes

B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Are there more values in the right subtree of T1 than the left subtree?

A. Yes
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Are there more values in the right subtree of T1 than the left subtree?

A. **Yes**
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Could adding the last new value take time proportional to logN?

A. Yes
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Could adding the last new value take time proportional to logN?

A. **Yes**
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Could adding the last new value take time proportional to N?

A. Yes
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Could adding the last new value take time proportional to N?

A. **Yes**
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Could adding the last new value take time proportional to NlogN?

A. Yes
B. No

Practice Final Review - Question 9

We have a BinarySearchTree T0 that contains N numerical values and has its root as the median value in the tree. We add N more values to T0 and form T1 as a result. Each added value is greater than the max value in T0 in an unknown order. Could adding the last new value take time proportional to NlogN?

A. Yes
B. No

Practice Final Review - Question 10

We are implementing an undo functionality to save the user’s actions and undo them in reverse order. We have a SinglyLinkedList with a reference to its head. For best performance, how would we store the user’s actions?

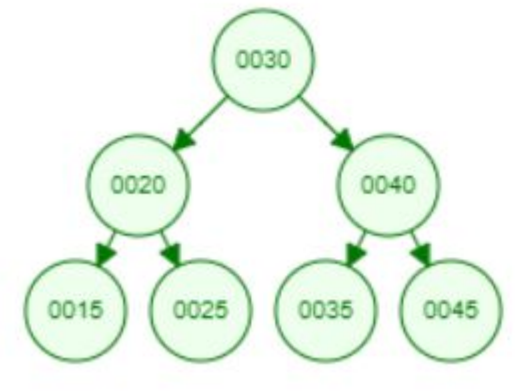
A. Add to end of list and remove from start
B. Add to start of list and remove from end
C. Add to end of list and remove from end
D. Add to start of list and remove from start
E. C and D

Practice Final Review - Question 10

We are implementing an undo functionality to save the user’s actions and undo them in reverse order. We have a SinglyLinkedList with a reference to its head. For best performance, how would we store the user’s actions?

A. Add to end of list and remove from start
B. Add to start of list and remove from end
C. Add to end of list and remove from end
D. Add to start of list and remove from start
E. C and D

Practice Final Review - Question 11

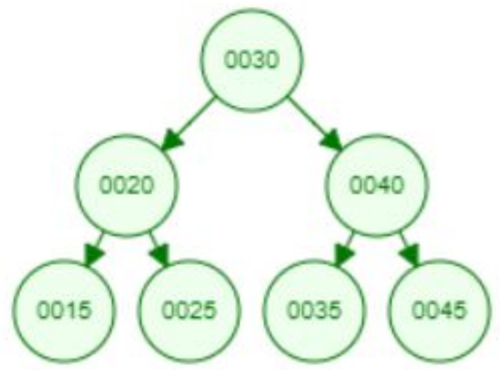


search (e)
Input: An element e
Description: Determines if e is in the tree
Return: True if e is in the tree; False if not

Suppose you just wrote the search method for your BinarySearchTree implementation. Which set of 4-integer values should we search for to gain as much confidence in our algorithm?

A. 10, 15, 20, 30
B. 15, 25, 35, 45
C. 15, 20, 30, 40
D. 20, 30, 40, 50
E. 25, 30, 40, 50

Practice Final Review - Question 11



search (e)
Input: An element e
Description: Determines if e is in the tree
Return: True if e is in the tree; False if not

Suppose you just wrote the search method for your BinarySearchTree implementation. Which set of 4-integer values should we search for to gain as much confidence in our algorithm?

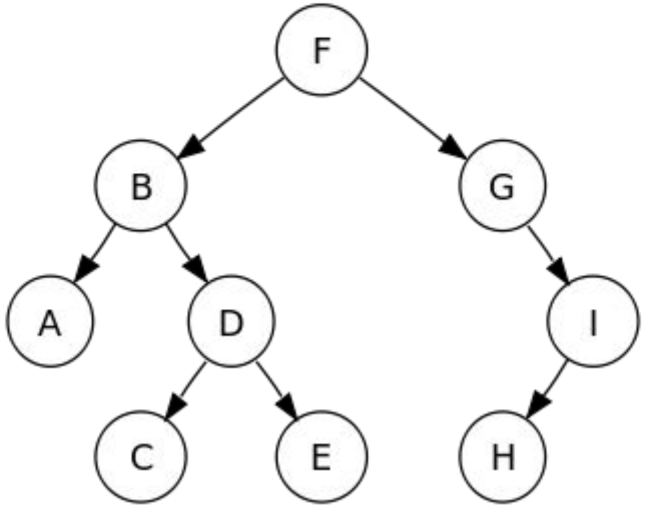
A. 10, 15, 20, 30
B. 15, 25, 35, 45
C. 15, 20, 30, 40
D. 20, 30, 40, 50
E. 25, 30, 40, 50

Practice Final Review - Question 12

width: maximal number of nodes on a path from one node in the tree to another

What is the width of this tree?

A. 4
B. 5
C. 6
D. 7

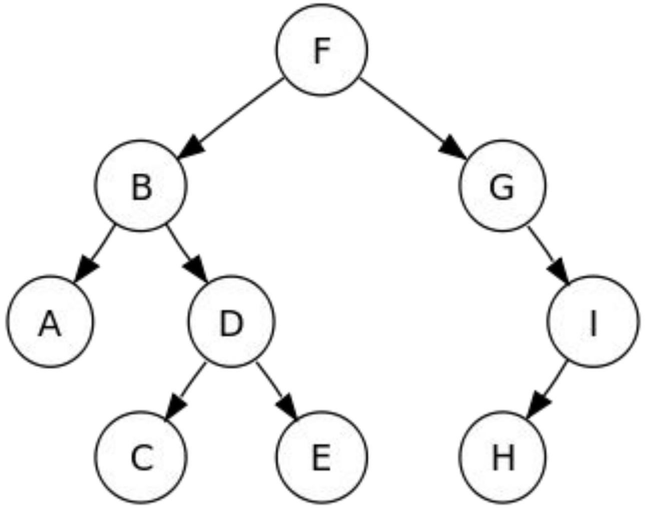


Practice Final Review - Question 12

width: maximal number of nodes on a path from one node in the tree to another

What is the width of this tree?

A. 4
B. 5
C. 6
D. 7



Practice Final Review - Question 12

How can we calculate the width of a Binary Tree?

A. Calculate the max height of the tree
B. Calculate the sum of the max number of nodes in the left and right subtree
C. Calculate the sum of the max height of the left and right subtree
D. Calculate the sum of the max number of children in the left and right subtree

Practice Final Review - Question 12

How can we calculate the width of a Binary Tree?

A. Calculate the max height of the tree
B. Calculate the sum of the max number of nodes in the left and right subtree
C. Calculate the sum of the max height of the left and right subtree
D. Calculate the sum of the max number of children in the left and right subtree

Practice Final Review - Question 13

We have a Set interface, which does not add element e if e is already in the Set. We also have a List interface, which adds all elements e to the List. We need to determine the number of unique email addresses we’ve received an email from. If we choose to use the Set interface rather than the List interface, did we choose the right interface?

A. Yes
B. No

Practice Final Review - Question 13

We have a Set interface, which does not add element e if e is already in the Set. We also have a List interface, which adds all elements e to the List. We need to determine the number of unique email addresses we’ve received an email from. If we choose to use the Set interface rather than the List interface, did we choose the right interface?

A. **Yes - the Set interface cannot contain duplicates**
B. No