CS310 Workbook: A Work in Progress!

by Lots of People

CS211 Review and Basic Java

You have a Java class named Program in Program.java which contains a main method. Main takes a single argument (your GNumber). Write the two lines of code below which will compile and run this code:

Compile: _____

Run: _____

Given an iterator, where is the current position of the iterator `iter` at the end of the code sequence on the right?

a. positioned at A
b. positioned at B
c. positioned at C
d. positioned before A
e. positioned between A and B
f. positioned between B and C

```
LL l = new LL([A, B, C, D])
iter = l.iterator()
iter.next()
iter.next()
```

Suppose we define a static method **`removeHalf()`** that removes the first half of a List (or slightly less than one-half of the list if odd length) as shown on the right. What can be used to fill the blank?

a. only **`size`**
b. only **`lst.size()`**
c. either **`size`** or **`lst.size()`**
d. neither **`size`** nor **`lst.size()`**

```
public static void removeHalf(List<T> lst) {
       int size = lst.size( );
       for(int i=0; i<_________/2; i++)
              lst.remove( 0 );
}
```

Self-Study Questions [No Answers!]

1. Explain how to define and use a generic class in Java and theorize why it might be useful for data structures (collections of objects).

2. Describe how you make a class "iterable" in Java. What interfaces do you need? What methods are required by those interfaces? Are there any additional optional methods?

3. Write code to implement a basic iterator for an array of integers.

4. Explain recursion to someone who has never heard of it before. Make sure your answer includes the following terms: base case, recursive case, readability, efficiency, and iteration.

5. Explain the four ways to "nest" a class inside another class. What are the differences and advantages of each type of nesting?

Answers:

`javac *.java OR javac Program.java`

`java Program G`

f (between B and C), a (only size since the list is changing)

CS310 Workbook: A Work in Progress!

by Lots of People

Asymptotic Analysis (Big-O)

[Math Review] Match the following terms on the left with the terms on the right:

a. Logarithm	1. lg(__)
b. Logarithm base 2	2. ln(__)
c. Natural Logarithm	3. log(__)
d. Logarithm base 10	4. $\log_{10}$(__)

[Math Review] Match the following equations on the left with the values on the right:

a. lg(1)	1. 0
b. lg(2)	2. 1
c. $\lg(2^2)$	3. n
d. $\lg(2^n)$	4. 2
e. $\lg(n^2)$	5. 2n
f. $2^{\lg(2n)}$	6. 2 lg n

What are the two types of complexity we discussed in class for assessing algorithms?

_____ complexity _____ complexity

What is the Big-O of the following? [Careful! Tricky...]

_____
```
10n + O(lg(n))
```

_____
```
log(n2) + 100
```

_____
```
for(int i = 0; i < n*n; i++)
    for(int j = 0; j < i; j++)
        System.out.println(1);
```

_____ Concatenating n strings with the '+' operator?

Place the following in order from "best" to "worst" asymptotic complexity:

O(n) , $O(2^n)$, O(1) , $O(n^3)$, O(log n) , $O(n^2)$, O(n log n), O(n!)

_____ < _____ < _____ < _____ < _____ < _____ < _____ < _____

Week 1: Self-Study Questions [No Answers!]

1. Explain time/space complexity using a real life example (like the time examples we did in class).
2. If I stated that the big-O of a method was O(1) and that it also contained a loop, describe that loop.
3. What's the difference between "best case", "worst case", and "average case" Big-O?
4. Explain the formal definition of Big-O:
   T(n) is O(F(n)) if there are positive constants c and n_0 such that, when n >= n_0, T(n) <= c F(n).

Answers:
a3|b1|c2|d4, a1|b2|c4|d3|e6|f5,
time & space, O(n), O(lg n), $O(n^4)$, $O(n^2)$

CS310 Workbook: A Work in Progress!

by Lots of People

Dynamic Arrays

You have a dynamic array with 2 elements in it and a capacity of 3. The capacity doubles when additional space is needed and shrinks by 50% (rounding down) when the size is at or below 1/3 capacity. What will be the capacity of the list after the following operations have been performed (answer should be a single integer number):

Answer:

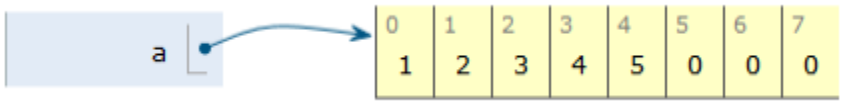
Add, Add, Remove, Remove, Add

You have a dynamic array with 2 elements in it and a capacity of 3. The capacity doubles when additional space is needed and shrinks by 50% (rounding down) when the size is at or below 50% capacity. What will be the capacity of the list after the following operations have been performed (answer should be a single integer number):

Answer:

Add, Add, Remove, Add

Given the dynamic list named "a" below, which contains the elements 1-5 and has a capacity of 8:



Indicate which of the following operations would be valid and which would produce an error. Assume that **each operation starts with a fresh copy of the list above**. Circle the correct answer where:

`Valid | Error` Insert an element at index 7

`Valid | Error` Insert an element at index 8

`Valid | Error` Insert an element at index 0, then another at 0, then another at 0

`Valid | Error` Insert 10 elements at index 0

`Valid | Error` Insert an element at index 5, then another at 6, then another at 7

Self-Study Questions [No Answers!]

1. Explain (with a drawing) what the memory looks like for a list stored as a dynamic array list. Make sure to represent the memory needed to store additional helper variables.

2. Write the pseudocode for `insert()` for a dynamic array list with generic type `T`.

3. Explain why we say that the worst case big-O of appending to a dynamic array list is O(n) while its amortized cost is only O(1).

Answers:
3, 6, E, E, V, V, V

CS310 Workbook: A Work in Progress!

by Lots of People

Linked Lists

Given the Node class definition on the right, which of the following are ints if the variable **`head`** is a reference to a **`Node`** at the start of a linked list **`1->2->3`**?

a. **`head`**
b. **`head.next`**
c. **`head.next.next`**
d. **`head.value`**
e. **`head.next.value`**

```
class Node { int value; Node next; }
```

For the same linked list (**`1->2->3`**), what is the value of **`head.next.value`**?

Answer:

Given the Node class definition and code snippet on the right, which of the following are null?

a. **`head`**
b. **`head.next`**
c. **`head.next.next`**
d. **`head.value`**
e. **`head.next.value`**

```
class Node<T> { T value; Node<T> next; }

Node<Integer> head = new Node<>();
head.next = new Node<>();
```

Given the **`Node`** constructor definition on the right, which statement inserts an item **`x`** after node **`c`**?

a. **`c = new Node<>(x, c);`**
b. **`c = new Node<>(x, c.next);`**
c. **`c.next = new Node<>(x, c);`**
d. **`c.next = new Node<>(x, c.next);`**
e. **`none of the above`**

```
public Node(T v, Node<T> n) {
       this.value = v;
       this.next = n;
}
```

Suppose we define a static method **`totalEvenLoc()`** that returns the total of even-index items of an integer List as shown below. What is the Big-O running time of the method if **`lst`** is a **`LinkedList`** of size N?

Answer:

```
public static int totalEvenLoc(List<Integer> lst) {
       int sum = 0;
       for(int i=0; i <lst.size(); i+=2 )
              sum += lst.get(i);
       return sum;
}
```

Self-Study Questions [No Answers!]

1. In one sentence, explain the difference between a dynamic array list and a linked list.

2. Explain the difference between a "node" class and a "linked list" class. Include in your description the terms: **`node`**, **`next`**, **`value`**, **`head`**, and **`tail`**.

3. Compare and contrast singly and doubly linked lists, including the pros and cons of using them.

4. Explain (with a drawing) what the memory looks like for a list stored as a doubly linked list. Make sure to represent the memory needed to store additional helper variables (such as size, head, etc.).

Answers:
d & e (anything .value), 2,
c,d,e (anything .value and head.next.next),
d, $O(n^2)$ because of .get() in a loop

CS310 Workbook: A Work in Progress!

by Lots of People

Stacks and Queues

What will be the contents of the stack (shown from stack bottom on the left to stack top on the right), after the given sequence of operations shown on the right?	`Stack s = new Stack();`
	`s.push(4);`
	`s.push(8);`
	`if (s.peek()>5)`
	`  s.pop();`
	`s.push(1);`
	`s.push(6);`
	`s.pop();`

What will be the contents of the queue (shown from queue front on the left to queue back on the right), after the given sequence of operations shown on the right?	`Queue q = new Queue();`
	`q.enqueue(4);`
	`q.enqueue(8);`
	`if (q.peek()>5)`
	`  q.dequeue();`
	`q.enqueue(1);`
	`q.enqueue(6);`
	`q.dequeue();`

For the circular array-based queue implementation with optimal Big-O, what will be the contents of the array at the end of the given sequence of operations? Assume an initial size of 3 for the array implementation and the size of array doubles each time it needs to expand and never shrinks. Initially the empty array is represented as [-,-,-].	`Queue q = new Queue();`
	`q.enqueue(1);`
	`q.enqueue(2);`
	`q.dequeue();`
	`q.enqueue(3);`
	`q.enqueue(4);`
	`q.dequeue();`

What is the optimal worst-case big-O for the following priority queue operations?

_____ enqueue() when stored as a sorted list

_____ dequeue() when stored as an unsorted list

_____ peek() when stored as a sorted list

_____ peek() when stored as a unsorted list

Self-Study Questions [No Answers!]

1. Explain how to implement a [queue | stack] with a [singly | doubly] linked list with the optimal Big-O? Which end(s) should you insert/remove from for the optimal Big-O?
2. Explain how to store a stack in an array with the optimal big-O.
3. Explain how a circular queue works for storing a queue in an array.
4. Write the code to perform `peek()` for a queue stored as an array.
5. Write the code to perform `push()` for a stack stored as a singly linked list.

Answers:
[4, 1], [8, 1, 6], [4,-,3]
O(n), O(n), O(1), O(n)

CS310 Workbook: A Work in Progress!

by Lots of People

Hashing Part 1: Basics and Separate Chaining

What is the range of values computed by the function `f(x) = x mod 10`?

Answer:

Given the following hash function: `h(x) = floor(x.key)`, compute the hash code for each key-value pair and the location in a hash table where the element would go if the hash table was completely empty and had a fixed size of 7.

Element shown as (key,value) pairs	Hash Code (your answer)	Hash Table Position (your answer)
(1.55, 10)		
(7.12, 10.4)		
(10.2, 44)		
(0.0, 1.4)		

Which of the following is true for a hash table with separate chaining? Circle all that apply:

a. There is an upper bound to the number of elements that we can insert into the hash table because of space constraints in the array.
b. There is no upper bound to the number of elements that we can insert but inserting too many elements can affect the run time of insert
c. There is no upper bound to the number of elements that we can insert but inserting too many elements can affect the run time of find
d. There is no upper bound to the number of elements that we can insert and no performance impact for any operations since we can always append or prepend to the list

Self-Study Questions [No Answers!]

1. What is the goal of hashing? (i.e. Why would we want to hash something into a hash table?)
2. Explain the difference between a hash code, a hash function, and a hash table. How does a hash code differ from the index of a hash table?
3. Create a hash function for playing cards which generates relatively unique values. (Standard playing cards have values Ace-King; suits Spaces, Diamonds, Clubs, and Hearts; and there are also two Jokers in the deck. Jokers have no suit.)
4. Explain Java's hash-contract and why it is necessary.
5. Explain how Java Strings, Integers, Double, and Character classes produces hash codes.
6. Hash 4 floating point numbers into a hash table of size 5 using separate chaining and the hash function `h(x) = floor(x)`. After hashing all numbers, remove 2 of them.

Answers:
0-9 (inclusive),
1 | 1, 7 | 0, 10 | 3, 0 | 0 ,
b & c (insert slow to check for duplicates)

CS310 Workbook: A Work in Progress!

by Lots of People

Hashing Part 2: Open Addressing

Given the input {4371, 1323, 6173, 4199, 4344, 9679, 1989}, a fixed table size of 10 with linear probing, and a hash function `h(x) = x`. Which option(s) below are **NOT true** of the table after we add the inputs in order (no deletion). Assume that we start with an empty hash table (t) which stores Integers (not ints). Circle all that apply:

a. `t[0] == 9679`
b. `t[4] == 4344`
c. `t[6] == null`
d. `t[9] == 1989`
e. All are true

Given a hash table with a fixed size of 10 using quadratic probing and a hash function `h(x) = x`. If we start from an empty hash table and perform only insertions. Given the final status of the table shown below on the right, which of the following **can NOT be true** in terms of insertion order? Circle all that apply:

a. 55 was inserted before 76
b. 76 was inserted before 55
c. 25 was inserted before 76
d. 76 was inserted before 25
e. 25 was inserted before 15
f. 15 was inserted before 25

```
0: null
1: null
2: null
3: null
4: 15
5: 55
6: 76
7: null
8: null
9: 25
```

Complete the blanks in the sentences below with the correct operation. Operation options are: “adding”, “removing”, “searching”, “rehashing”, and these words may be used more than once!

We create tombstones when _____ to ensure _____ works correctly.

When _____ we “skip” tombstones because they indicate we should “keep going”. When

_____ we replace tombstones to keep the data “closer” to its target location in the table.

When _____ we do not copy tombstones.

Self-Study Questions [No Answers!]

1. Explain the idea of "load" on a table and how it relates to the big-O of the hash table's add, remove, and contains methods.
2. Hash 4 floating point numbers into a hash table of size 5 using open addressing quadratic probing and the hash function h(x) = floor(x). After hashing all numbers, remove 2 of them, making sure to visually indicate any special states.
3. Rehash your tables from Question 2 to a size 8.
4. Compare and contrast the three types of probing we covered for open addressing. Make sure to note the benefits and disadvantages of each.

Answers:
b & d ([9679, 4371, null, 1323, 6173, 4344, null, 1989, null, 4199]
c & f (not true: 25 was inserted before 76, 15 was inserted before 25)
removing/searching/searching/adding/rehashing

CS310 Workbook: A Work in Progress!

by Lots of People

Tree Basics and Storage

Given the tree in the box on the right, answer the following questions:

How many leaves are in the tree?

Answer:

What is the height of the tree?

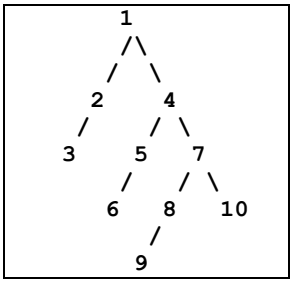
Answer:

How many siblings does node 8 have?

Answer:

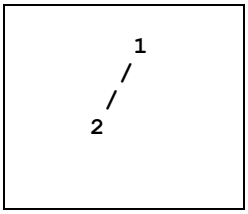
How many parents does node 8 have?

Answer:

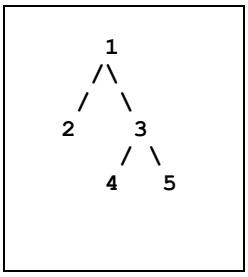


Assuming the tree on the right is a **binary** tree, which of the following properties does it have? Select all that apply.

a. It is full
b. It is perfect
c. It is nearly complete
d. It is balanced
e. It is degenerate
f. None of the above



Given the tree shown on the right, and assuming the tree is a binary tree, complete the array representation of that tree below. Leave ***empty*** any spots which are null and ***cross out*** any spots that go past the end of the tree.



0	1	2	3	4	5	6	7	8	9

Given the same tree, and assuming the tree is a k-ary tree with k=3, give a *possible* array representation in the same way below:

0	1	2	3	4	5	6	7	8	9	10	11	12

Self-Study Questions [No Answers!]

1. If you have a *perfect* binary tree of height 2, how many nodes are in the tree?
2. Explain how to store a binary tree using an array. Make sure to explain how to find the child/parent of a given node based only on the index of that element. Can you generalize the idea to k-ary trees?

Answers:

4/4/1/1, c & d & e (nearly-comp, balanced, degenerate),

[1,2,3,_,_,4,5,X,X,X], [1,2,3,_,_,_,_,4,5,X,X,X]

CS310 Workbook: A Work in Progress!

by Lots of People

Tree Traversals

[Math Review] Answer the following questions as precisely as possible:

a. What is the height of the tallest tree you can make with n nodes? _____

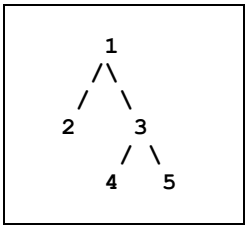
b. What is the height of the shortest binary tree you can make with n nodes? _____

c. How long (in Big-O terms) would it take to reach the lowest leaf of a tree with n nodes *in the worst case*? _____

d. How long (in Big-O terms) would it take to reach the lowest leaf of a *balanced* tree with n nodes *in the worst case*? _____

For the given tree, which of the following traversals processes in order "12345"? Select all that apply.

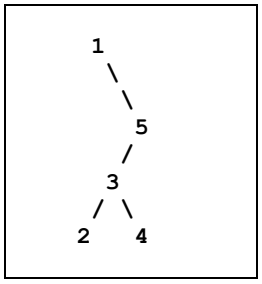
a. pre-order
b. in-order
c. post-order
d. level-order
e. none of the above



Given the tree on the right, perform the following walks:

In-Order: ____ ____ ____ ____ ____

Post-Order: ____ ____ ____ ____ ____



For the given tree, Let `c(t)` be the number of leaves in a **non-empty binary tree** rooted at `t`. Assume that `isLeaf(t)` returns 1 if `t` is a leaf and 0 otherwise. Which of the following leads to a recursive implementation of `c(t)`?

a. `c(t)= c(t.left)+ c(t.right)`
b. `c(t)= c(t.left)+ c(t.right)+ 1`
c. `c(t)= c(t.left)+ c(t.right)+ isLeaf(t)`
d. `c(t)= c(t.left)+ c(t.right)+ isLeaf(t)+ 1`
e. None of the above

Self-Study Questions [No Answers!]

1. Ask a classmate to draw you a binary tree with 6 nodes… Perform a pre-order, post-order, in-order, and level-order walk of their tree.

2. Write the code for a *recursive* method to calculate the size or height (your choice) of a **binary** tree. Assume a generic Node<T> class with a data field and separate left and right child fields (writing this class out may be helpful to you).

Answers:
n-1 (heights start at 0), floor(lg(n)),
a & d (both), 1|2|3|4|5, 2|4|3|5|1, c

CS310 Workbook: A Work in Progress!

by Lots of People

Heaps and Heap Sort

Start with an **empty heap**, insert 10,12,6,14,1,5 in order into a **min-heap**. Which numbers **require swapping** (percolating up) when we insert them? Circle all that apply:

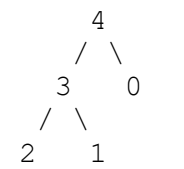
a. 10
b. 12
c. 6
d. 14
e. 1
f. 5

Start with an **empty heap**, insert 9,15,6,21,-5,1 in order into a **min-heap**. Which numbers have **ever been stored at the top** (root)? Select all that apply.

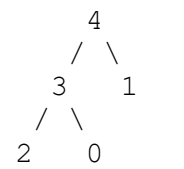
a. 9
b. 15
c. 6
d. 21
e. -5
f. 1

Start with an **empty heap**, insert 3,4,2,5,0,1 in order into a **max-heap**. If we then perform **remove()** once after insertion, which option gives the right heap after deletion? Circle one:

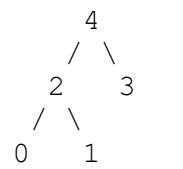
a.



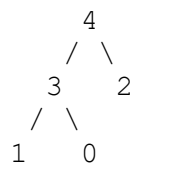
b.



c.



d.



Circle true or false for the following questions:

True | False The **maximum** item of a **min-heap** must be at one of the leaves.

True | False There are ⌈N/2⌉ **leaves** in a heap with N nodes.

True | False Heaps are stored in "level order" when stored as arrays.

Self-Study Questions [No Answers!]

1. Write an O(n) method that accepts an array and checks if it is a **min-heap** OR a **max-heap**.
2. How can you determine the **max** value in a **max-heap**? How can you determine the **min** value in a **max-heap**? What is the Big-O of each operation?
3. Explain how heapsort is O(n lg n) and can be done "in place".
4. Given an array of numbers, show the steps of the optimal heapify algorithm.

Answers:
c & e & f, a & c & e, d, all True

CS310 Workbook: A Work in Progress!

by Lots of People

Graph Basics and Traversals

What type of graph might be stored, given the adjacency matrix on the right? Circle all that apply:

a. an undirected graph
b. a directed graph
c. an unweighted graph
d. a weighted graph

```
[[0,1,0,0,1,0],
 [1,0,1,0,1,0],
 [0,1,0,1,0,0],
 [0,0,1,0,1,1],
 [1,1,0,1,0,0],
 [0,0,0,1,0,0]]
```

Determine if the adjacency matrix and the adjacency list below represent the same graph:

```
[[0,1,0,0,1,0],
 [1,0,1,0,1,0],
 [0,1,0,1,0,0],
 [0,0,1,0,1,1],
 [1,1,0,1,0,0],
 [0,0,0,1,0,0]]
```

```
0: → 1 → 4
1: → 0 → 2 → 4
2: → 1 → 3
3: → 2 → 4 → 5
4: → 0 → 1 → 3
5: → 3
```

Answer:

Given the adjacency list in the previous problem, perform a depth-first traversal starting at **node 3**. Choose neighbors as the computer would (following the storage).

Depth-fist: ____ ____ ____ ____ ____ ____

Given the adjacency list in the previous problem, perform a breadth-first traversal starting at **node 5**. Choose neighbors as the computer would (following the storage).

Breadth-fist: ____ ____ ____ ____ ____ ____

Self-Study Questions [No Answers!]

1. Explain the following terms:

   a. graph
   b. node
   c. edge
   d. adjacent to
   e. directed edge
   f. weight
   g. |V|
   h. |E|
   i. path
   j. simple path
   k. path length
   l. cycle
   m. degree
   n. DAG

2. Explain the difference between the following types of graphs:

   a. directed/undirected
   b. weighted/unweighted
   c. cyclic/acyclic
   d. dense/sparse
   e. connected/disconnected

3. Explain the difference between breadth-first and depth-first *graph* traversals and *tree* traversals.

4. Draw an undirected graph with six nodes and choose a source node id. Perform a breadth-first traversal showing the steps of the accompanying queue data structure. Given a choice of neighbors, chose nodes in numerical order.

5. Write code for a recursive depth-first search. Assume you are given an adjacency matrix m (int[][]), the id of a node to start from, and an id of a node to search for. This should return true/false depending on if one can get to the search target from the starting node.

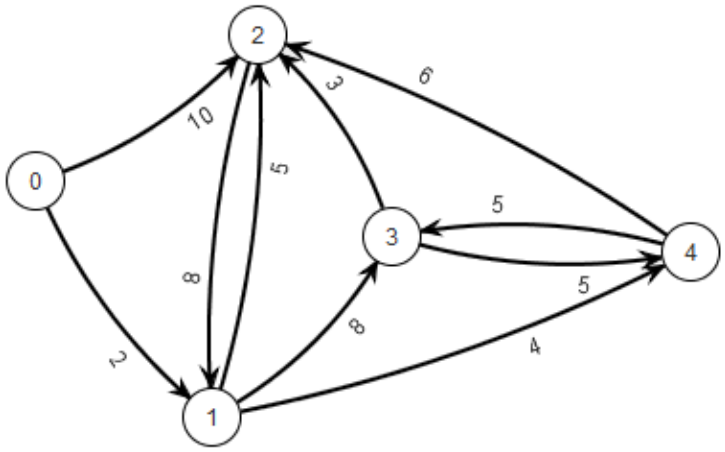
Answers:
all are possible, same,
3|2|1|0|4|5. 5|3|2|4|1|0

CS310 Workbook: A Work in Progress!

by Lots of People

Graph Algorithms Cont.

For the given graph on the right, and breaking ties by choosing the lowest node id, perform Dijkstra's shortest path algorithm starting at **node 0**, then answer the following questions.



Give the order nodes are finished in: ___ ___ ___ ___ ___	Give the parent IDs of each node: ___ ___ ___ ___ ___ 0 1 2 3 4
Put a checkmark over each node that is updated ***more*** than once before it is "done": ___ ___ ___ ___ ___ 0 1 2 3 4	Give the final cost of each node: ___ ___ ___ ___ ___ 0 1 2 3 4

Answer the following questions about topological sorting:

a. Can you perform a topological sort of the graph in the previous problem? _____

b. Given a simple directed graph: 0 → 1, if you start a DFT-based topological sort, does it matter which node you start from? _____

c. Given this directed graph: 0 ← 1 → 2, how many valid topological sorts exist? _____

d. Complete the following sentence: "When performing a topological sort with a DFT, we put nodes on a _____ [data structure] when the node is _____ [finished | started].

Self-Study Questions [No Answers!]

1. What is Dijkstra's algorithm used for?
2. Draw a directed graph with six nodes and choose a source node id. Perform Dijkstra's shortest path algorithm on a graph showing the steps. Use a table to track the current status, distance, and parent pointer for each node.
3. Given a DAG, perform a topological sort on the given graph showing the depth-first-traversal order and the final sorted output.

Answers:
0|1|4|2|3, check only on #2, -1|0|1|1|1,
0|2|7|1|0|6, no (not a DAG), no, 2, stack|finished

CS310 Workbook: A Work in Progress!

by Lots of People

Binary Search Trees and AVLs

Starting from a null root, construct a binary search tree T1 by applying insert() method to the sequence of numbers: 10, 5, 1, 2, 4, 60, 22; construct a binary search tree T2 by applying insert() method to the reverse order: 22, 60, 4, 2, 1, 5, 10. Which option is true about the height of T1/T2? Circle one:

a. T1's height > T2's height
b. T2's height > T1's height
c. T1 and T2 have the same height

Starting from a null root, construct a binary search tree T1 by applying insert() method to the sequence of numbers: 10, 5, 1, 2, 4, 60, 22. Which of the following is true? Circle all that apply:

a. tree root is the first number we insert: 10
b. tree root is the median: 5
c. 1 and 60 are both leaf nodes
d. 4 and 22 are both leaf nodes

Starting from a null root, construct a binary search tree T2 by applying insert() method to the sequence of numbers: 22, 60, 4, 2, 1, 5, 10. Assuming a remove implementation as we discussed in class (successor replacement), if we call T2.remove(4), what will be the child nodes of 22 after removal? Circle one:

a. 4 and 60
b. 5 and 60
c. 2 and 60
d. null and 60
e. none of the above

Determine an order of inserting the keys 1, 2, and 3 into an empty AVL tree to meet the given requirements. Write your answer in the provided box as three consecutive numbers, like: 1 2 3. Give only one order:

An order which would require no rotations:

An order which would require a double rotation:

How many rotations are performed if you insert the following numbers into an empty AVL? Count single rotations as 1 and double rotations as 2.

700, 100, 300, 600, 800, 200, 500, 400

Answer:

Self-Study Questions [No Answers!]

1. Explain the "rules" of a binary search tree? What properties have to be maintained when adding a node to and removing a node from the tree?
2. Why is the Big-O of inserting into a BST O(n) and not O(lg n)?
3. Write the code for searching a BST. Assume a generic `Node<T>` class with a data field and left/right references. Assume the search method is given the root of the tree and a value to search for.
4. Explain the "cases" for inserting into an AVL tree.

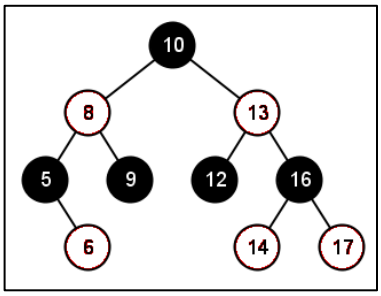
Answers:
a (T1.height > T2.height), a & d (root 10, 4&22 leaves),
b (5 and 60), 2 1 3 or 2 3 1, 3 1 2 or 1 3 2,
3 (2 for 300, 1 for 400)

CS310 Workbook: A Work in Progress!

by Lots of People

Red-Black Trees

For the next three problems, use the Red-Black tree shown on the right:



For the given red-black tree, inserting which number will **not need any rotation** or any recolor? Assume that white nodes are actually red (since this paper is printed in black-and-white), and we are using bottom-up insertion (the method we did in class). Circle all that apply:	a. 1 b. 7 c. 11 d. 15

For the given red-black tree, inserting which number will require both rotation and recolor? Use the same assumptions from Question 1. Circle all that apply:	a. 1 b. 7 c. 11 d. 15

For the given red-black tree, after adding node 15, how many nodes will get a different color during the fixing process? Use the same assumptions from Question 1.	Answer:

If we insert the numbers 15 and 3 into an **empty** red-black tree (in that order). What case will we have if we next try to insert the number 1? Circle one:

a. Case 1: This node is the root
b. Case 2: The parent of this node is black
c. Case 3: The parent of this node is red and the uncle of this node is red
d. Case 4: The parent of this node is red and the uncle of this node is black

Self-Study Questions [No Answers!]

1. Explain the "cases" for inserting into a Red-Black tree.
2. Determine a set of keys and an order to insert them which would produce each of the cases for inserting into a Red-Black tree. Label each case with a meaningful name (not just "case 1", etc.).
3. Compare AVL trees and Red-Black trees pros and cons. Explain when/why you would use an AVL instead of a Red-Black Tree and explain when/why you would use an a Red-Black Tree instead of an AVL.
4. Write the code for `printSubset(int min, int max)` in a red-black tree where this method prints the subset of the tree with values greater than or equal to the min and less than or equal to the max. For example, if `printSubset(9,15)` was called on the tree used in the participation questions (on the other side of this sheet), this method should print "`9 10 12 13 14`" Assume a generic `Node<T>` class with a data field, a boolean called `isBlack`, and left/right references.

Answers:
a & c (1 and 11), b (7), 6 nodes (10, 8, 13, 16, 14, and 17 are all changed at some point), d (uncle is a null link)