Test Review Materials

General Resources

https://www.swamiiyer.net/cs210/lecture_material.html - - Dont worry about all the material - only review what you covered

Non-exhaustive summary

- BigO notation time complexities (What are they, What do they mean, Examples, which are best)
- Data Structures (Usage, Properties, Time Complexities of different operations + implementations)
  - Stacks
  - Heaps (Max/Min)
  - Queues / Priority Queues
  - Linked Lists
  - Binary Tree
  - etc
- Object Oriented / Java Key Terms
  - Super
  - Polymorphism
  - Inheritance
  - Etc
- Java Concepts
  - Defining a simple Class
  - Comparators
  - Iterators
  - Constructors / Overloading / Overriding
  - Private Public Static Final
  - Packages
  - General Java (Reading + Writing code)
  - etc
- Sorting (Runtime,
  - Bubble
  - Insertion
  - Quick
  - Merge
- Other Algorithms
  - Binary Search

Essentially, review anything you covered in class!

Note:

Follow examples are not necessarily reflective of exam questions. However, they will be beneficial for you understanding of different structures/concepts so that you can easily answer any question about them.

BigO Notation

List them in order of worst to best and give an example of each!

Helpful Resource: https://www.bigocheatsheet.com

BigO Notation

List them in order of worst to best and give an example of each!

Helpful Resource: https://www.bigocheatsheet.com

n^n, 2^n, n^(n > 2), n^2, nlogn, n, logn, 1

What does this method do?

Exercise 1. Consider the following function:

```
public static int mystery(Node<Integer> first) {
    int x = 0;
    for (Node y = first, int i = 0; y != null; y = y.next, i++) {
        x += (i % 2 == 0) ? y.item : 0;
    }
    return x;
}
```

a. What does `mystery()` compute and return in general?

b. What will `mystery()` return if the argument `a` represents a linked list containing integers $1, 2, 3, \ldots, 10$?

https://www.swamiiyer.net/cs210/basic_data_structures_exercises.pdf

What does this method do?

Exercise 1. Consider the following function:

```
public static int mystery(Node<Integer> first) {
    int x = 0;
    for (Node y = first, int i = 0; y != null; y = y.next, i++) {
        x += (i % 2 == 0) ? y.item : 0;
    }
    return x;
}
```

a. What does `mystery()` compute and return in general?

b. What will `mystery()` return if the argument `a` represents a linked list containing integers $1, 2, 3, \ldots, 10$?

a. Computes and returns the sum of every other integer in `node`, starting at the first.

b. 25

https://www.swamiiyer.net/cs210/basic_data_structures_exercises.pdf

Stack

Exercise 3. Suppose that a minus sign in the input indicates pop the stack and write the returned value to standard output, and any other string indicates push the string onto the stack. Further suppose that following input is processed:

```
it was - the best - of times - - - it was - the - - worst - of times -
```

a. What is written to standard output?

b. What are the contents (from top to bottom) left on the stack?

Exercise 5. Consider the following code fragment:

```
Stack<Integer> s = new Stack<Integer>();
while (n > 0) {
   s.push(n % 2);
   n = n / 2;
}
while (!s.isEmpty()) {
    StdOut.print(s.pop());
}
StdOut.println();
```

a. What does the code output when `n` is 50?

b. What does the code output in general for a non-negative integer `n`?

https://www.swamiiyer.net/cs210/basic_data_structures_exercises.pdf

Stack

Exercise 3. Suppose that a minus sign in the input indicates pop the stack and write the returned value to standard output, and any other string indicates push the string onto the stack. Further suppose that following input is processed:

```
it was - the best - of times - - - it was - the - - worst - of times -
```

a. What is written to standard output?

b. What are the contents (from top to bottom) left on the stack?

Solution 3.

a. `was best times of the was the it worst times`

b. `of it`

Exercise 5. Consider the following code fragment:

```
Stack<Integer> s = new Stack<Integer>();
while (n > 0) {
   s.push(n % 2);
   n = n / 2;
}
while (!s.isEmpty()) {
    StdOut.print(s.pop());
}
StdOut.println();
```

a. What does the code output when `n` is 50?

a. `110010`

b. What does the code output in general for a non-negative integer `n`?

b. Prints the binary representation of `n`.

https://www.swamiiyer.net/cs210/basic_data_structures_exercises.pdf

Queues

Exercise 8. What does the following code fragment do to the queue `q`?

```
Stack<String> s = new Stack<String>();
while(!q.isEmpty()) {
   s.push(q.dequeue());
}
while(!s.isEmpty()) {
   q.enqueue(s.pop());
}
```

https://www.swamiiyer.net/cs210/basic_data_structures_exercises.pdf

Queues

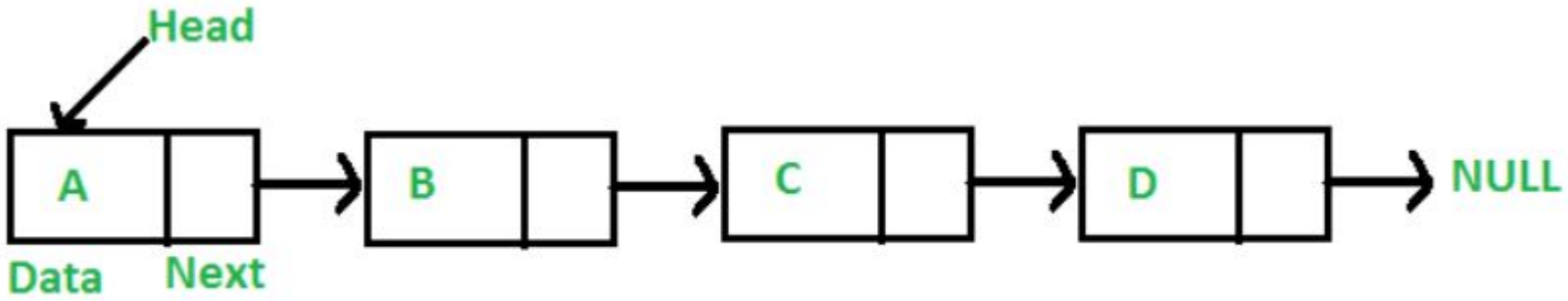
Exercise 8. What does the following code fragment do to the queue `q`?

```
Stack<String> s = new Stack<String>();
while(!q.isEmpty()) {
   s.push(q.dequeue());
}
while(!s.isEmpty()) {
   q.enqueue(s.pop());
}
```

Solution 8. Reverses the items on the queue.

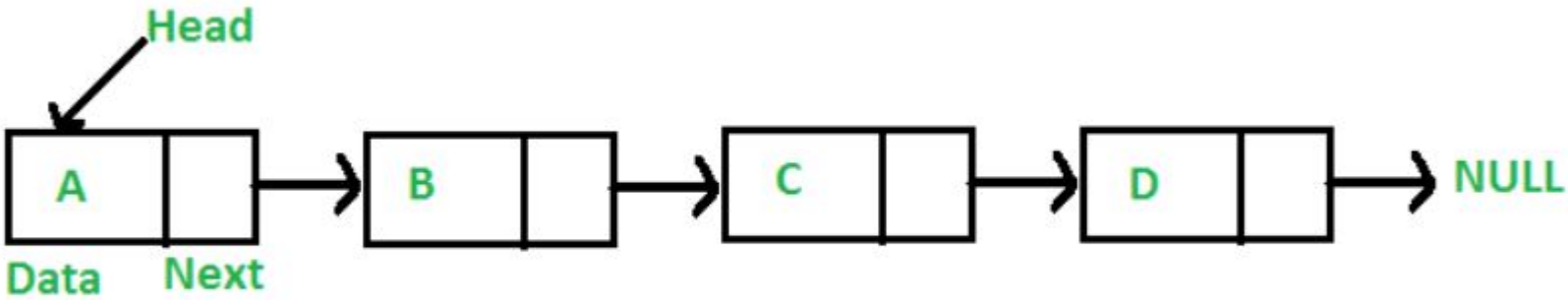
https://www.swamiiyer.net/cs210/basic_data_structures_exercises.pdf

Linked List



In general terms, how would you search for C in this linked list? What would the runtime be of that operation.

Linked List



In general terms, how would you search for C in this linked list? What would the runtime be of that operation.

I'd look at the data of the first node, and compare to C. If its not C, I would replace my current node with current.next. I would repeat this process until C is found.

This would have a runtime of O(n)

Heaps (Both Min, Max, and General)

Exercise 1. Insert the following keys in that order into a max-heap:

E A S Y Q U T I O N

a. What is the state of the array `pq` representing the resulting tree?

b. What is the height of the tree (the root is at height zero)?

C. Runtime for insertion and
getting the Max?

https://www.swamiiyer.net/cs210/priority_queues_exercises.pdf

Heaps (Both Min, Max, and General)

Exercise 1. Insert the following keys in that order into a max-heap:

E A S Y Q U T I O N

a. What is the state of the array `pq` representing the resulting tree?

b. What is the height of the tree (the root is at height zero)?

C. Runtime for insertion and getting the Max?

Solution 1.

a. - Y S U O Q E T A I N

b. 3

C. O(logn) and O(1)

https://www.swamiiyer.net/cs210/priority_queues_exercises.pdf

Priority Queue

Exercise 2. Suppose that a letter in the input means *insert the letter* into an initially empty min-PQ and an asterisk (*) means *remove the minimum* from the priority queue. What is left in the priority queue after the following input is processed?

P R I O * R * * I * T * Y * * * Q U E * * * U E *

https://www.swamiiyer.net/cs210/priority_queues_exercises.pdf

Priority Queue

Exercise 2. Suppose that a letter in the input means *insert the letter* into an initially empty min-PQ and an asterisk (*) means *remove the minimum* from the priority queue. What is left in the priority queue after the following input is processed?

P R I O * R * * I * T * Y * * * Q U E * * * U E *

Solution 2. U

https://www.swamiiyer.net/cs210/priority_queues_exercises.pdf

Binary Tree

1 Exercises

Exercise 1. Consider inserting the following keys (assume values to be non `null` and arbitrary) into a binary search tree (ordered) symbol table `st`, an object of type `BST`.

S Y M B O L T A E X P

a. What is the binary search tree (BST) that results? List the keys along with their indices (starting at 1) in level order.

b. What is the height of the BST (assume root to be at height 0)?

c. What is the order in which the keys are visited if we traverse the BST in pre-order?

d. What is the order in which the keys are visited if we traverse the BST in in-order?

e. What is the order in which the keys are visited if we traverse the BST in post-order?

f. What is the order in which the keys are visited if we traverse the BST in level-order?

Binary Tree

1 Exercises

Exercise 1. Consider inserting the following keys (assume values to be non `null` and arbitrary) into a binary search tree (ordered) symbol table `st`, an object of type `BST`.

S Y M B O L T A E X P

a. What is the binary search tree (BST) that results? List the keys along with their indices (starting at 1) in level order.

b. What is the height of the BST (assume root to be at height 0)?

c. What is the order in which the keys are visited if we traverse the BST in pre-order?

d. What is the order in which the keys are visited if we traverse the BST in in-order?

e. What is the order in which the keys are visited if we traverse the BST in post-order?

f. What is the order in which the keys are visited if we traverse the BST in level-order?

Depth First Traversals:

(a) Inorder (Left, Root, Right)

(b) Preorder (Root, Left, Right)

(c) Postorder (Left, Right, Root)

Breadth-First or Level Order Traversal

Binary Tree Solutions

a. 1: S, 2: M, 3: Y, 4: B, 5: O, 6: T, 8: A, 9: L, 11: P, 13: X, 18: E

b. 4

c. S M B A L E O P Y T X

d. A B E L M O P S T X Y

e. A E L B P O M X T Y S

f. S M Y B O T A L P X E

https://www.swamiiyer.net/cs210/binary_search_trees_exercises.pdf

Object Oriented Programming

- Polymorphism - think inheritance
- Encapsulation - making instance variables private so that they can only be modified through public methods — gives more control to the application than the user
- Abstract - can only be used through inheritance, cannot be used on own as an object
- Packages - used to group related classes
- Inheritance - to get attributes and methods that belong to another class

https://www.w3schools.com/java/java_polymorphism.asp
https://www.w3schools.com/java/java_encapsulation.asp
https://www.w3schools.com/java/java_abstract.asp
https://www.w3schools.com/java/java_inheritance.asp

Keywords

- Super - to refer to the class you are directly inheriting from
- Public - access modifier - allows an attribute to be accessed from any other class
- Private - access modifier - allows an attribute to only be accessed from within its own class
- Protected - access modifier - allows an attribute to only be accessed from within the same package and subclasses
- Final - access modifier - prevents an variable from being changed, and a method/class from being overridden or inherited
- Static - access modifier - methods/attributes can be accessed without creating an object of a class

https://www.w3schools.com/java/ref_keyword_super.asp
https://www.w3schools.com/java/ref_keyword_static.asp
https://www.w3schools.com/java/java_ref_keywords.asp

Comparators

What are comparators used for?

“A comparator interface is used to order the objects of user-defined classes”

The Comparator interface can be used to keep comparisons standard.

https://www.geeksforgeeks.org/comparator-interface-java/

Iterators

What does an iterator allow?

Allows for a class to be looked over

Ex. A stack has an iterator, so you are able to looke over it in a for each loop

https://www.w3schools.com/java/java_iterator.asp

Define a simple class for a Node in a Linked List (no comparator necessary)

Define a simple class for a Node in a Linked List (no comparator necessary)

```
Class Node {
        Int data;
        Node next;
}
```

Constructors / Overloading / Override

What does a constructor do?

What is overloading a method and what does it allow for?

What is overriding a method and what does it do?

Constructors / Overloading / Override

What does a constructor do?

When a new ClassName(); is called, a new object of type ClassName is generated and initialized.

What is overloading a method and what does it allow for?

Overloading a method is when you have multiple methods called methodX, but each has a varying number of parameters. This allows for the method to accept different inputs so the user does not have to stick to a single model.

What is overriding a method and what does it do?

Overriding a method is to create your own implementation of an inherited class. Override allows a developer to ignore the method from the class' super, and instead make the method better fit their need.

Binary Search

In your own words, describe what the binary search algorithm is, how it works, and its time complexity.

Binary Search

In your own words, describe what the binary search algorithm is, how it works, and its time complexity.

Binary Search is an algorithm to efficiently search an already sorted structure for a particular item. The algorithm first finds the middle element of the structure and compares it to the item they are searching for. If the middle element is compared to be less than the desired element, the scope of the search is limited to all the elements to the left of the middle and the process is repeated. The same applies for if the middle element is greater than the desired item, but instead only elements to the right are considered, and the search continues in that partition. If at any point the middle element is the desired item, its index is returned.

The time complexity of this algorithm is O(logn). This is because with each comparison, the scope of the search is cut in half.

Bubble Sort

Describe how bubble sort works in your own words:

Runtime?

Bubble Sort

Describe how bubble sort works in your own words:

Starting from the beginning of the array, look at first element and compare it to the next element. If they are out of order, swap them. Move to the next element and repeat. Once the of the array is reached, go back to the beginning and repeat the process. If no elements are swapped in that next run, the array is sorted and the process can end.

Runtime?

O(n^2)

https://visualgo.net/en/sorting

Insertion Sort

Describe how insertion sort works in your own words:

Runtime?

Insertion Sort

Describe how insertion sort works in your own words:

Runtime?

Insertion Sort

Describe how insertion sort works in your own words:

Iterate over the array from index 1 to index n. Compare the current index to the previous index. If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element

Runtime?

O(n^2)

https://visualgo.net/en/sorting

Merge Sort

Describe how merge sort works in your own words:

Runtime?

Merge Sort

Describe how merge sort works in your own words:

Break down an array into halves. When a partition has reached a point where it has 2 or 1 elements, sort the two (or 1). And then go back up a level, combine the partitions in that level, and sort the elements of that partition..

Runtime?

O(nLogn)

https://visualgo.net/en/sorting

Quick Sort

Describe how quick sort works in your own words:

Runtime?

Quick Sort

Describe how quick sort works in your own words:

Iterate over the array from index 1 to index n. Compare the current index to the previous index. If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element

Runtime?

O(n^2)

https://visualgo.net/en/sorting

Sorting = Stable and inPlace?

In Place -> no additional memory is needed for manipulating inputs (extra array)

Stable -> does not change the order of elements with the same value.

Sorting Algorithms
Bubble Sort
Selection Sort
Insertion Sort
Quick Sort
Merge Sort

https://afteracademy.com/blog/comparison-of-sorting-algorithms

Sorting = Stable and inPlace?

In Place -> no additional memory is needed for manipulating inputs (extra array)

Stable -> does not change the order of elements with the same value.

Sorting Algorithms	In - Place
Bubble Sort	Yes
Selection Sort	Yes
Insertion Sort	Yes
Quick Sort	Yes
Merge Sort	No (because it requires an extra array to merge the sorted subarrays)

https://afteracademy.com/blog/comparison-of-sorting-algorithms

Sorting = Stable and inPlace?

In Place -> no additional memory is needed for manipulating inputs (extra array)

Stable -> does not change the order of elements with the same value.

Sorting Algorithms	In - Place	Stable
Bubble Sort	Yes	Yes
Selection Sort	Yes	No
Insertion Sort	Yes	Yes
Quick Sort	Yes	No
Merge Sort	No (because it requires an extra array to merge the sorted subarrays)	Yes

https://afteracademy.com/blog/comparison-of-sorting-algorithms