EECS 233 SI Session 10
Leader: Bertram Su
October 10, 2019

Objectives:
Upon completion of this SI session, participants will be able to:

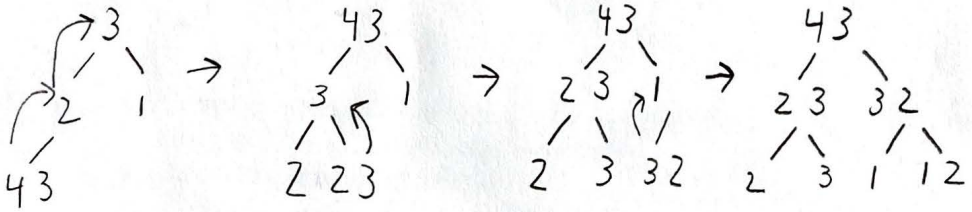
1. Determine what adding and removing would do to a heap
2. Implement heapify in our array implementation of heaps

Foundations:

1. What variables does our heap class need?

int itemCount
array

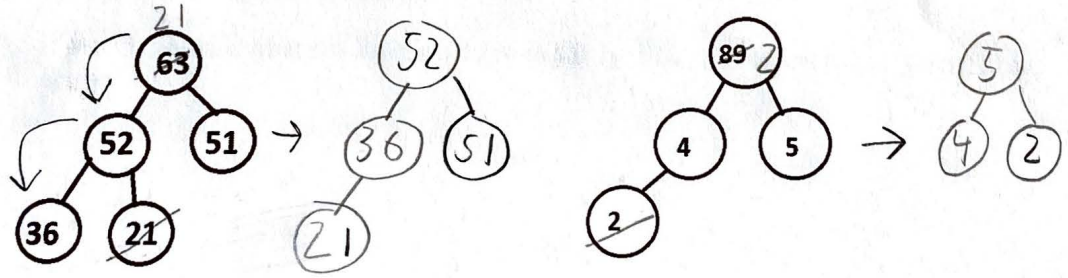
2. Draw the max heaps after adding the following numbers: 3, 2, 1, 43, 23, 32, 12



3. What are the rules for creating a complete binary tree?

Top to bottom
Left to right Bertram Su

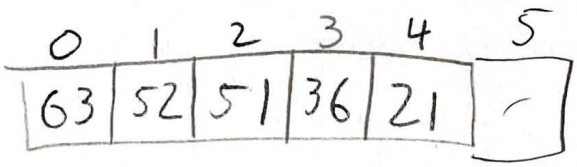
4. Draw the following max heaps after removing the top from both



5. What are the left child, right child, and parent indexes of a node at index i?

left = i×2+1
right = i×2+2
parent = $\frac{i-1}{2}$

6. Draw the first tree from question 4 in its array form before any operations



Exercises:

7. Complete the following method for swapping integers from an array. Assume you have access to an array called data

```
public void swap(int index1, int index2){
    int temp = data[index1]
    data[index1] = data[index2]
    data[index] = temp

}
```

8. Complete the heapify up method. Assume our data in the array is ints, we have access to swap, and can use everything a heap needs

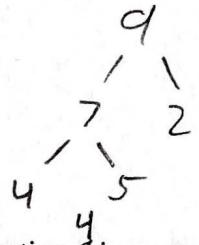
```
public void heapifyUp(int index){ aka sift up
    while( index > 0 && data[index] > data[(index-1)/2] ) {
        swap(index, (index-1)/2);
        index = (index-1)/2;

}
```

9. In words, describe what sift down would do in a heap. You have access to everything a heap needs.

check greater child (if it exists)
check if current node > greatest child
    do nothing
else
    swap current w/ greatest child
    keep going until no longer less than child
    or we are a leaf

Summary

10. Draw the max heap after the following operations:
Add 5, 4, 2, 9, 7, 12, Remove top



11. Draw the second heap from question 6 in an array before any of the operations

0	1	2	3
89	4	5	2

12. What are the big O times of inserting and removing? Why?

log n , sifting

13. What situations would we use a heap?

paying user , boss , rankings , sorting
n log n

14. Using the methods that we created, complete the add method for a heap

```
public void add(int element) {
        if (itemCount == data.length)
                //We are full or increase size, depending on what we want to do;
        data[itemCount] = element;
        heapify up (item count)
        item Count++;

}
```

Upcoming Events and Suggestions for Further Study:

Events:

- Next SI session is Sunday from 1:00 to 2:30 at Sears 336
    - It will be a review session for topics covered before the midterm

Further Study:

- http://www.algolist.net/Data_structures/Binary_heap/Array-based_int_repr
    - An elaboration on the array implementation of heaps. I took an example from here

EECS 233 SI Session 11
Leader: Bertram Su
October 13, 2019

n^2 vs $\log n$

n^3 vs $n \log n$

Objectives:
Upon completion of this SI session, participants will have reviewed the major topics from the weeks before the midterm.

Runtime Analysis big O expressions:
Write the simplest big O expression

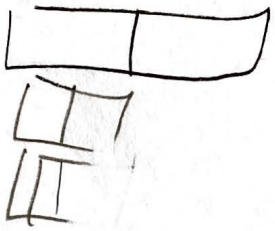
1. $1234^{8342} + 2n^2 + 97n$ — $O(n^2)$
2. $\frac{1}{2} n^{1/2} + \log_2 n$ — $n^{\frac{1}{2}}$
3. $300n + 64n + 32$ — n
4. $2^n + n^{120}$ — 2^n
5. $9999 + \log n$ — $\log n$
6. $64^{32} + n\log n + 2^n + n + \log n$ — 2^n

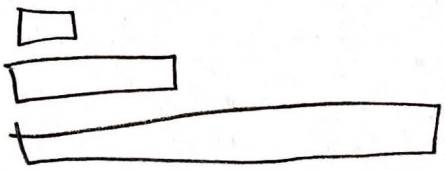
Runtime Analysis big O code:
Give the simplest big O expression

1.
```
void algebra(int n) {
    for (int i = 0; i < n; ++i) {                    n
        for (int j = 1; j < n; j=j*2) {              log2n
            System.out.println("j = " + j);
        }
        for (int k = 0; k < n * 3; ++k) {            3n
            System.out.println("k = " + k);
        }
    }
}
```
n^2

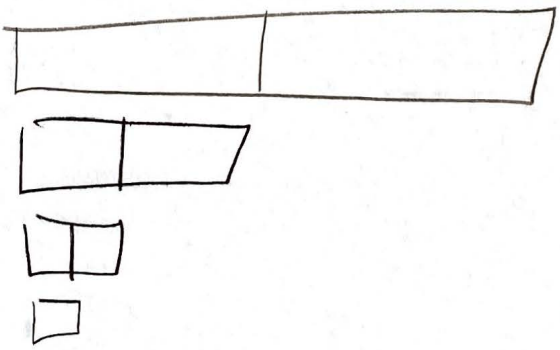
$n(\log_2 + 3n)$

$n\log_2 + 3n^2$



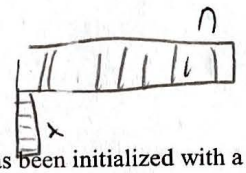


2.
```
public void hmmmmm(int n){
    for(int j = 0 ; j< n; j++){                      n
        int r = n;
        for(int k = 0 ; r>0; k++){                   log n
            r = r/2;
        }
    }
}
```
$n \log n$



3. ! void tricky(int n){
for(int i = 0 ; i < n; i++){ n
for(int j =0 ; j<x ;){ x
x--;
}
System.out.println("This question was on a test");
}
}

n x



//Assume x has been initialized with a value

n + x

Linked Lists

Operation	Linked Lists	Arrays
Access nth item	O(n)	O(1)
Search	O(n)	O(n)
Insertion (You have a field storing where to insert)	O(1)	O(n)
Deletion (You have a field storing where to delete)	O(1)	O(n)

1. Assume we have a Linked List comprised of Student nodes that hold a student id. We want to be able to find a Student by id. Complete the method

public Student search(int id) {

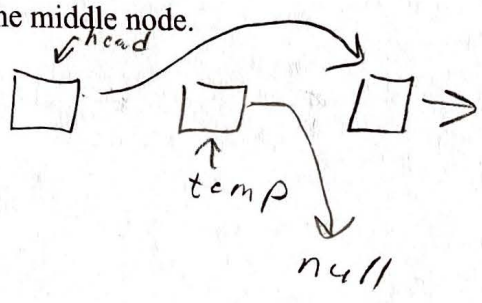
```
Student ptr = head;
while (ptr != null) {
    if (ptr.id == id)
        return ptr;
    ptr = ptr.next;
}
return null;
```

}

2. Assume we have three student nodes. Write the code to delete the middle node.

public void deleteMiddle(){

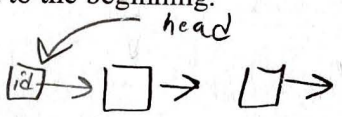
```
temp = head.next
head.next = temp.next
temp.next = null
```



3. Assume we have one student node. Write the code to add to the beginning.

public void addBeginning(int id){

```
head = new Student (id, head);
size++;
```

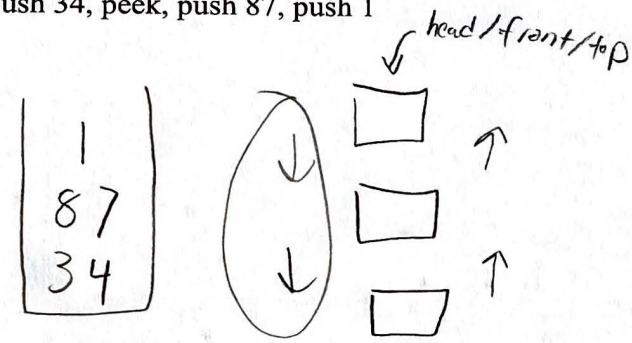


}

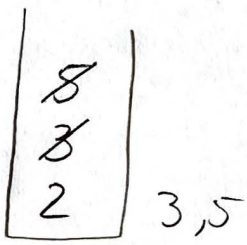
Stacks and Queues

1. Draw the resulting stacks of the following operations.

a) push 34, peek, push 87, push 1

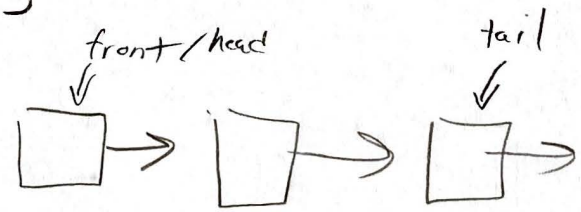


b) push 2, push 3,pop, push 5, pop



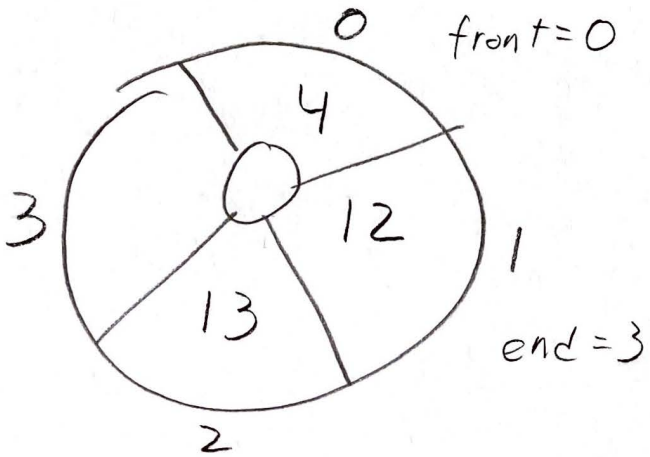
2. Draw the Queue after the following operations: add 56, add 4, remove, add 12, add 13

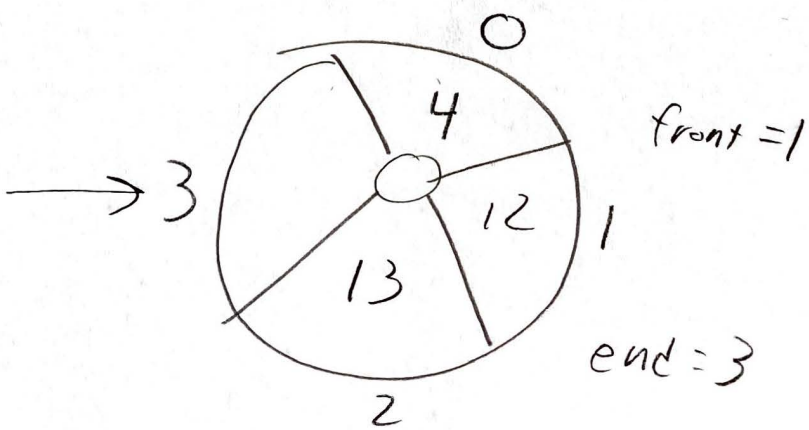
~~56~~, 4, 12, 13



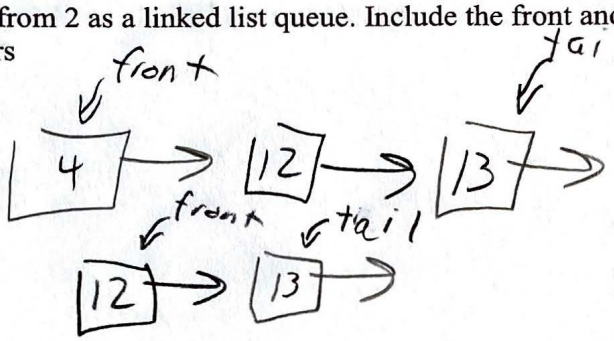
56

3. Draw the final queue from 2 as a fresh circular queue of size 4 before and after a Dequeue. Give the index values of the front and end values before and after the dequeue



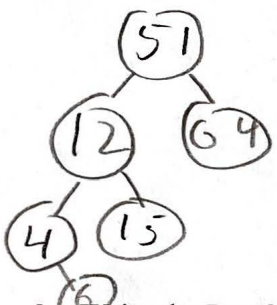


4. Draw the final queue from 2 as a linked list queue. Include the front and rear pointers. Deque and update the pointers



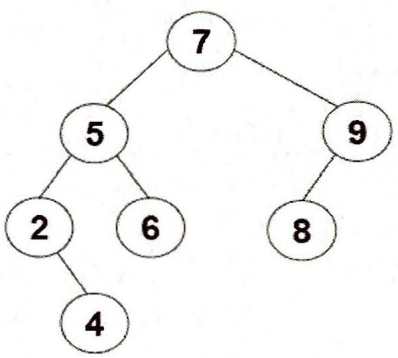
Binary Search Trees

1. Draw the binary search tree after the following operations: Add 51, 64, 12, 4, 6, 15



2. Write the Pre-Order, In-Order, and Post-Order traversal for the tree

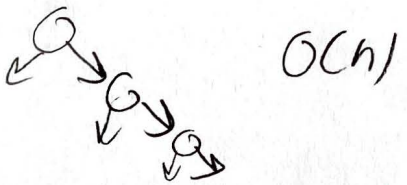
Pre: print, Left, right,
in: Left, print, right
Post: Left, right, print



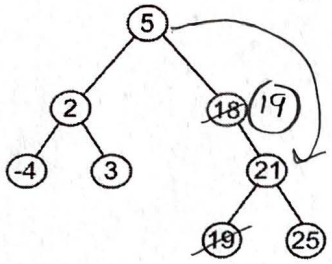
Pre: 7, 5, 2, 4, 6, 9, 8

in: 2, 4, 5, 6, 7, 8, 9

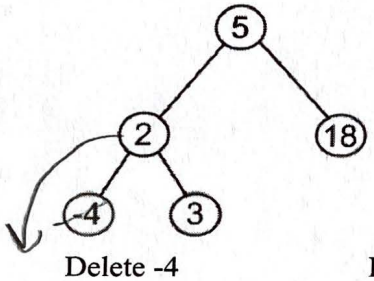
post: 4, 2, 6, 5, 8, 9, 7



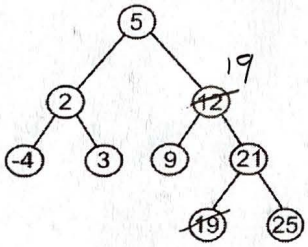
3. Redraw the following BST trees. There are technically two correct answers for each



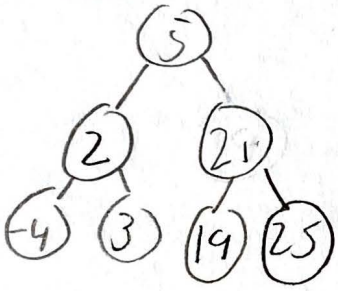
Delete 18

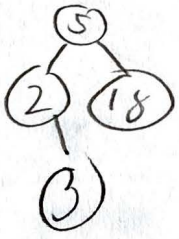


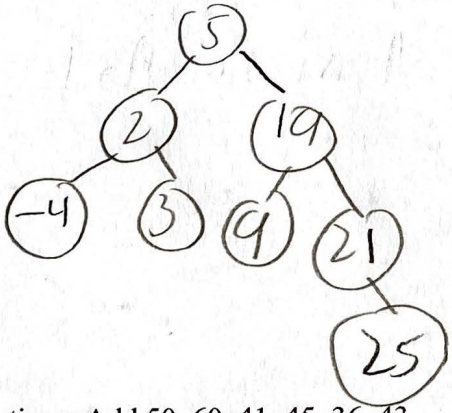
Delete -4



Delete 12

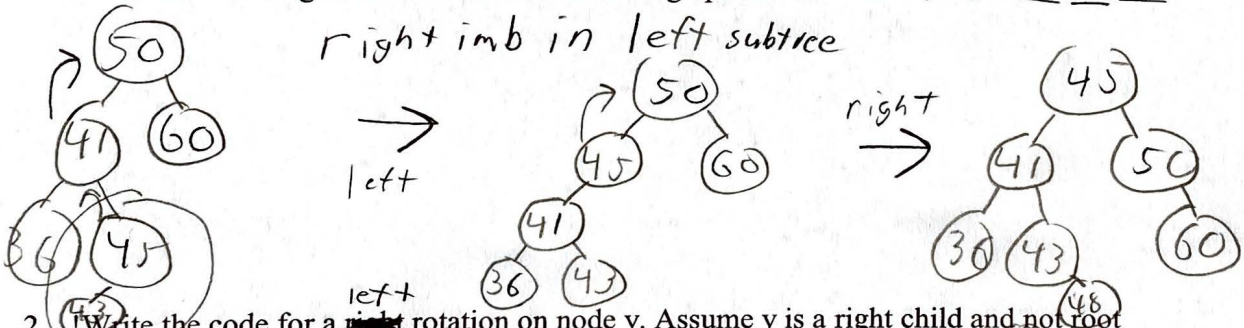






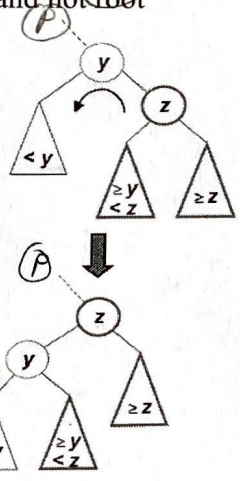
AVL Trees

1. Draw the resulting AVL tree after the following operations: Add 50, 60, 41, 45, 36, 43



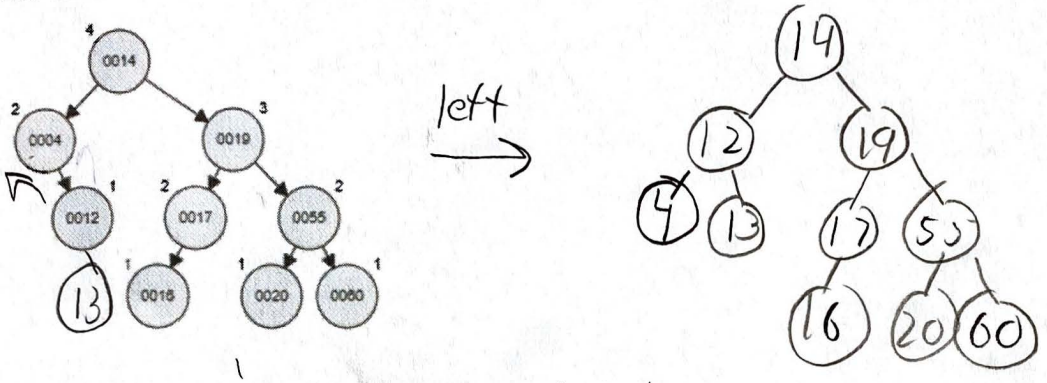
2. Write the code for a ~~right~~ left rotation on node y. Assume y is a right child and not root

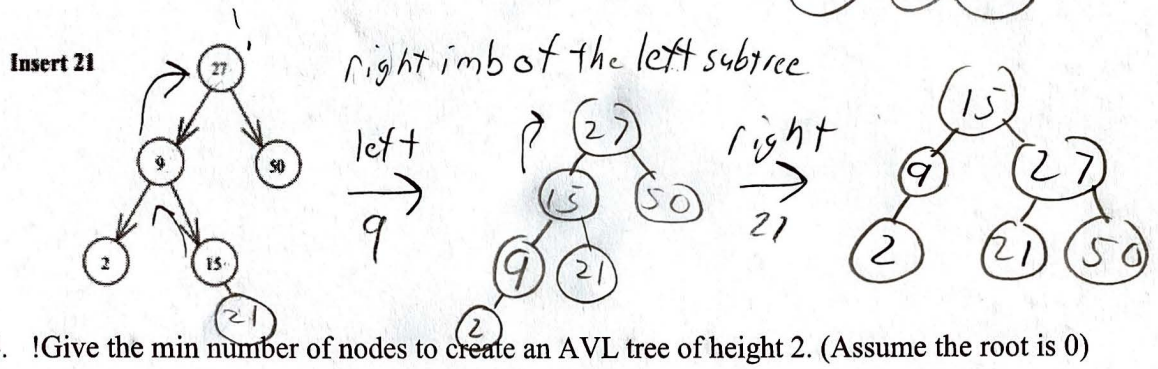
```
private void LeftRotation(Node y){
    Node z = y.right
    Node p = y.parent
    p.right = z

    y.right = z.left
    y.parent = z

    z.left = y
    z.parent = p

    y.right.parent = y
```



3

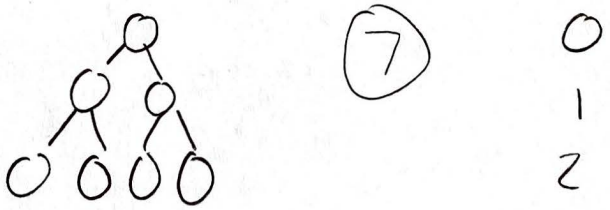
3. Redraw the following AVL trees if they need to be rebalanced after the following operations
Add 13





4. !Give the min number of nodes to create an AVL tree of height 2. (Assume the root is 0)

5. Give the max number of nodes to create an AVL tree of height 2. (Assume the root is 0)



Upcoming Events and Suggestions for Further Study

Events:

- Next SI session on Thursday is cancelled because we have a test the class before

Further Study:

- bigocheatsheet.com
  - A great graph that visualizes the big o complexity chart. It also has the big O time of data structures and algorithms that we will cover in the future.
- Review the quizzes
- Review linked list implementation