Ethan Ho
erh101@case.edu

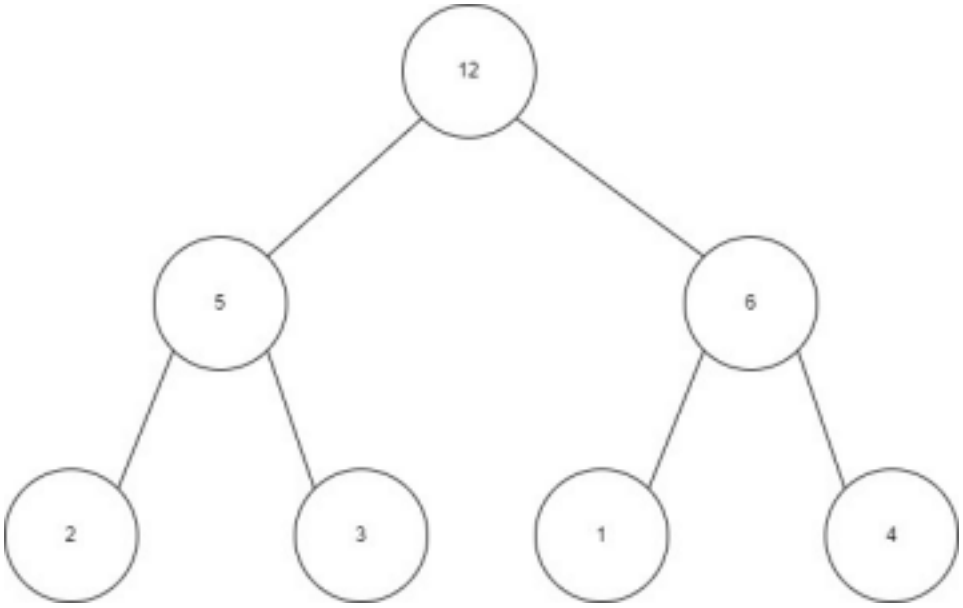
CSDS 233 - THE FINAL SESSION

Objective: Upon completion of this session, students will be goated data structures students with the ability to destroy the final with ease no hiccups. (Everything about heaps, hash tables, sorts, and graphs)

Note: All of the problems on this sheet are from after B-trees in the content order.

Problems:

1. Assume the following heap:



a. Draw what the heap would look like after removing the maximum vertex once.

b. **Use the heap from the original problem (meaning don't use your modifications from part a for this part)** Draw what the heap would look like after inserting a node of 7.

c. **Use the heap from the original problem**. Draw what the heap would look like after updating node 12 to a value of 0.

d. **Use the heap from the original problem**. Draw what the heap would look like after removing node 5.

e. **Use the heap from the original problem**. Write this heap as an array.

2. Let's sort the following array using HeapSort!
   [12, 4, 8, 9, 10]
   a. Show array after the BuildHeap() method.

   b. Now show each heap during the drainage method.

3. Assume the following hash table.

We have a class Student with an int variable of “grade” that ranges from 0-100, which represents a student’s grade in terms of percentage. The Student object also has a String variable of “name”, that represents the student’s name.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

a. Create hashing functions int h1(Student) and int h2(Student) out of the data to be used for a double hashing open addressing. There are many possible hashing functions you can make.

b. Use your hashing function to insert data into the table. Draw the hash table after inserting the values.

{name = ‘Ethan Ho’, grade = 24}
{name = ‘Marilyn Brown’, grade = 96}
{name = ‘Emmanuel Makoye’, grade = 100}
{name = ‘Kate Deson”, grade = 44}
{name = ‘Desmond Benjamin”, grade = 96}

4. Show a case demonstrating why the REMOVED flag is necessary in hash tables. In other words, assume we have a hash table without a REMOVED flag. What problem may occur?

5. What is one potential problem with the following code for the probe method for double hashing? Modify the code so the problem is solved.

```
private int probe(String key) {
      int i = h1(key); // first hash function
      int j = h2(key); // second hash function
      // keep probing while the current position is occupied
       (non-empty and non
      removed)
      while (table[i] != null && table[i].removed == false)
            i = (i + j) % tableSize;
      return i;
}
```

6. How do we handle situations where our hash table gets filled with REMOVED flags? Write out a method that might help us out.

7. For each of the following arrays, identify what sorting algorithm would be best to use to sort it as efficiently as possible. Justify your answer.

    a. [2, 10, 3, 19, 4, 6, 8, 1]

    b. [1, 2, 3, 4, 5, 6, 7]

8. Write the time complexity of each of the following sorts.

    a. Selection sort

    b. Insertion sort

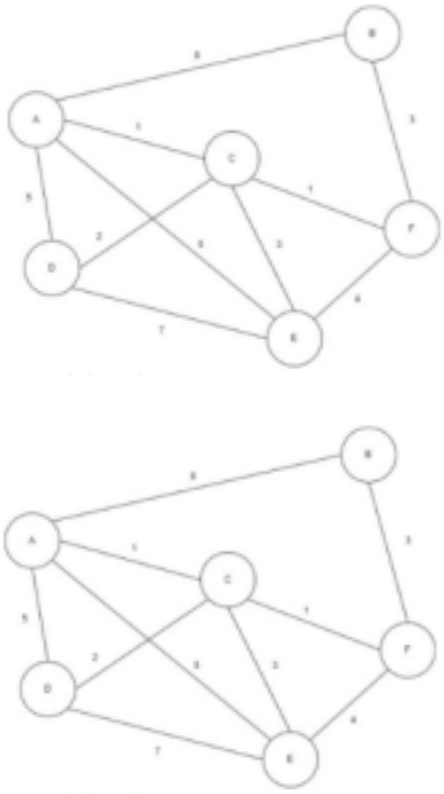
    c. Quick sort

    d. Merge sort

    e. Bubble sort

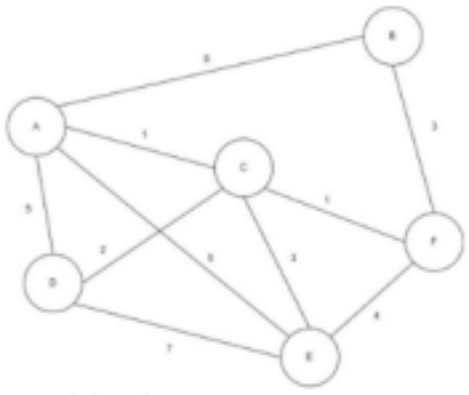
9. Write out method selectionSort(int[] arr], which sorts the array of integers in ascending order.

10. Outline a depth first traversal and a breadth first traversal from node A from the following graph.



(one picture for BF other for DF)

11. For the following graph, find the shortest paths from F to all other nodes using Dijkstra's. Draw out a trace chart if necessary.



Conclusion: Thank you guys for a great semester! Good luck on the final. There is no next session.

Today, we went over…

- Hash tables
- Heaps
- Graphs
- Sorts

Please feel free to email me at erh101@case.edu if you have any questions! Thank you guys :)

Hi now that you're at the end please do this survey:

https://cwru.az1.qualtrics.com/jfe/form/SV_8ffso6UAXBL6ayO