Build Heap

Michael Bushman

1. Let index = length/2-1. This is the parent of the last node in the tree, i.e. list[index + 1] . . . list[length-1] are leaves
2. Convert the subtree with root of list[index] into a heap.
    a. Given list[a] is root of tree, list[b] is left child (root *2 +1), list[c] is right child (root*2+2), if exists
    b. Compare list[b] with list[c] to determine larger child, list[largerIndex]
    c. Compare list[a] with list[largerIndex]. If list[a] < list[largerIndex], then swap, else already a heap
    d. If swap, repeat step 2 for the subtree of list[largerIndex]
3. Convert the subtree with the root of list[index-1] into a heap, repeat until list[0]

Heap Sort

1. Swap the root with the end of the list.
2. Heapify the list up to but not including the root
3. Repeat until there is only one node in the list

Heap Sort in Java | Tutorial

- Complete the following tutorial as best as you can (spend 4 hours or so)
  - https://stackabuse.com/heap-sort-in-java/
  - Create a new Ellipse project, get the provided code (heapify(), headSort() from the tutorial to work. From there, you can code insert() and delete() if you like.
- The purpose of the lab is for us to learn how Heap works and how to code a little of it
  - Include your java code in your submission
  - Simply submit any code that you have completed while working on the tutorial.
  - Discuss your overall learning experience

Simulate the heapsort algorithm manually by sorting the following 9 element array:

- [5, 22, 9, …, 28]
- Show all steps (Use an application of choice to complete the assignment)

1. Make into a heap
2. Sort

Max-Heap

[0]	5	5	5	5	5	92	92	92			
[1]	22	22	22	92	92	5	76	76			
[2]	9	9	81	81	81	81	81	81			
[3]	76	92	92	22	76	76	5	54			
[4]	63	63	63	63	63	63	63	63			
[5]	81	81	9	9	9	9	9	9			
[6]	48	48	48	48	48	48	48	48			
[7]	92	76	76	76	22	22	22	22			
[8]	54	54	54	54	54	54	54	5			
[9]	28	28	28	28	28	28	28	28			

Blank worksheet for your use

[0]	92	28	81	81	5	76	76	22	63	63	28	54
[1]	76	76	76	76	76	5	63	63	22	54	54	28
[2]	81	81	28	48	48	48	48	48	48	48	48	48
[3]	54	54	54	54	54	54	54	54	54	22	22	22
[4]	63	63	63	63	63	63	5	5	5	5	5	5
[5]	9	9	9	9	9	9	9	9	9	9	9	9
[6]	48	48	48	28	28	28	28	28	28	28	63	63
[7]	22	22	22	22	22	22	22	76	76	76	76	76
[8]	5	5	5	5	81	81	81	81	81	81	81	81
[9]	28	92	92	92	92	92	92	92	92	92	92	92

Blank worksheet for your use

[0]	54	9	48	5	28	28	5	22	9	5	5	
[1]	28	28	28	28	5	22	22	5	5	9	9	
[2]	48	48	9	9	9	9	9	9	22	22	22	
[3]	22	22	22	22	22	5	28	28	28	28	28	
[4]	5	5	5	48	48	48	48	48	48	48	48	
[5]	9	54	54	54	54	54	54	54	54	54	54	
[6]	63	63	63	63	63	63	63	63	63	63	63	
[7]	76	76	76	76	76	76	76	76	76	76	76	
[8]	81	81	81	81	81	81	81	81	81	81	81	
[9]	92	92	92	92	92	92	92	92	92	92	92	

Learning Experience

I learned a lot about heap sorting with this activity. I learned that comparison-based sorting technique based on binary heap data structure. It is similar to the selection sort where we first find the minimum element and place the minimum element at the beginning. I learned that heapify is the process of creating a heap data structure from a binary tree represented using an array. It is used to create Min-Heap or Max-heap. You start from the last index of the non-leaf node whose index is given by n/2 – 1. I found your link along with the tutorial from GeeksforGeeks to be very helpful in completing this lab. I also learned about min heap and max heap as well. In min heap, the key present at the root node must be less than or equal among the keys present at all of its children. This is useful for accessing the minimum element in the heap. In max heap, the key present at the root node must be greater than or equal among the keys present at all of its children. This is useful for accessing the maximum element in the heap. Overall, I found the sorting activity along with the actual coding assignment to be very helpful and informative.