Michael Kolb

ECE 36800

Test 1 BRAC

1. Reviewer #1
   1. Gave me full points for all the requirements I will readdress this in 2
2. Agree/Disagree
   1. I think that Reviewer 2 and the instructor did touch on things that I forgot to add into my lecture. I will address those further in the other sections, I think that in part 1,4,5,6,7 and 8. I think that many of the issues I had in these sections also sort of related to eachother.
3. Reviewer #2
4. I did not receive full points for part 4,6,7, and 8. To address these issues,
5. Mentioned why a heap is better for implementing a Priority Queue than a linked list, regular array, or binary search tree 0.5 pts. doesn't really mention why heap is better than the others but does roughly go over why heap is ideal for implementing a priority queue.
6. Provided an intuitive explanation as to why Insert(key) is constant time on average. 0.5 pts. doesn't really give a proper explanation as to why it is of constant time complexity. only mentions it.
7. Demonstrated how to have the Heap be an IMPLICIT tree using an array. 0.5 pts. it doesn't seem to go into how it is implemented as an array or what the parent child relationship in our array is. Would give a 0.25 if possible.
8. Described and contrasted using Upward\_Heapify() and Downward\_Heapify() for organizing a heap from an unordered array. 0.5 pts. does attempt to explain how upward and downward heapify work but not really on an unordered array. also does not contrast them.
9. Agree/Dissagree – For most of the items the reviewer found I agree with him/her about their reasonings and what they have found
10. 4, I think a much better way to address why Heap is better than others would be to include an example and/or the time complexity at the very least.
11. 6, I needed to do a better job of explaining why insert(key) is constant, I tried to show an example but I think it got muddled in my recording, I wanted to show how it worked and some times it could be in place and not need moved, however showing the time complexity make up of the function would be a good idea too.
12. 7, I did not give a good example of what can make the Heap implicit using an array, I should have explained and shown that using a Priority Queue which says every entry has a priority to where it is released and those with the highest priority get released first. Also I would want to explain how the children are related to the parents and are of a higher/lower priority to their children.
13. 8, I needed to give a better example and demonstration of how to use up and down heapify for building a heap. I did up heap but I should have done it again with down heapify to show it can be just as fast or in some cases faster.

* Reviewer #3

1. Gave me full point for all the requirements, I will readdress this in 2.

* Agree/Disagree
  1. Much like reviewer 1, it was nice of this person to give me full points but I do disagree and I would restate what I said in part 2 for reviewer 1 here.
* Instructor
  1. Got a 4/8 on this review will address what the instructor did not see and how to improve.
     + Part 1Described the “Priority Queue” abstract data type and its operations
     + Part 4Partial points. I heard the video mention BST and arrays, but I don't remember hearing a comparison of all these data structures for how they can implement a priority queue. (Correct in BRAC.)
     + Part 5, Partial points. It sounds like there was a small mistake at 12:40. For insert, best is O(1), worst is O(log n ), but \*average\* is O(1). (Correct in BRAC.)
     + Part 6, No points. Missing. (Correct in BRAC.)
     + Part7, No points. Missing. (Correct in BRAC.)
     + Part8, Partial points. Error at 7:15. Building a heap in the worst case is only O(n) if you use Downward\_Heapify. Did not cover that method. (Correct in BRAC.)
* Agree/Disagree
  1. In task 1, I described priority Queue, however I do believe that I failed to explain that in a priority queue there is an enqueue and dequeue function as well as a get max\_priority. I would show these as well as how they worked which is very important for understanding more on 7 which I will touch more on in that.
  2. In Task 4, I did not give a good enough reason, I used the proof by authority, just believe me not going to explain it. I Should have used more of a proof by time complexity comparison between the other options or demonstrated with an example.
  3. In Task 5, I explained some of the time complexities, but I believe that the time complexity I got wrong was I stated that worst case insert was O(n\*logn) which is wrong it should only be O(logn). I also could have done a better job of explaining or demonstrating why those time complexities were.
  4. In task 6, I think my explanation of the insert working got muddled, pretty sure it got basically cut out. I would want to do a much better example, I like the white board and magnets easier to manipulate without issues. In this would need to explain that when you insert a key, it goes to new node and many times this will not need to be changed around much more if at all.
  5. In Task 7,I failed to show that Heap can be an implicit tree, this I believe is two part in the first task I needed to give a better example of why priority queue uses heap so well, and I also think that explaining how a heap actually works and its relationship between the children and parents affects the structure of the tree, mainly how it determines where numbers are located given their parents or children.
  6. In Task 8, I tried to explain both downward and upward heapify, I ran out of time and I think the best way to explain downward heapify would be to show the building of a new heap using downward heapify. Also I think that showing pseudocode for both downward and upward heapify would be very helpful to the understanding and showing how the parts work.

1. Additionally I think that the method of recording was a bad choice I was excited to use an application I have for iPad but after looking at some other options I think that using MS Paint and or a White Board with magnets would be a much better option and something I can have set up ahead of time. Another thing would be to use more pseudocode or just to use code would be a good idea something to explain. I also think more demonstrations in general would never hurt, they might take more time but if there is a faster way to demonstrate it would work out beautifully.
2. As stated in the video I used tutorialpoint and geeksforgeeks with very little assistance from just general googling the topics.