# Michael Chillemi

# 06/16/2023

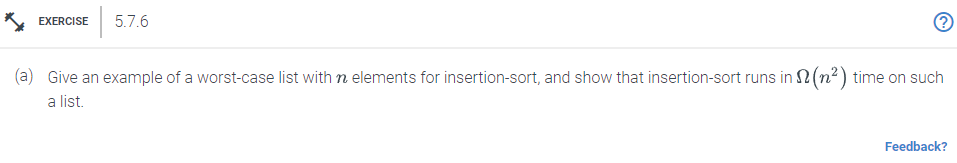
# CS 590 - Algorithms

# M4.B1: Module 4 Priority Queues and Heaps Reinforcement Exercises

# 

# 

Problem 5.7.6



Answer:

In this example we will perform insertion-sort on the list below and show why insertion-sort runs in time. Through each step we will be adding to our total comparison counter.

list[6]= {1,24,17,8,5,29}

In insertion sort the first element is skipped. So we begin at element list[1] which is 24 and we compare it to 1. Since 24 is greater than 1 the elements will stay put and the comparison counter will be incremented by one.

{1,24,17,8,5,29}

{1,17,24,8,5,29}

Next we will compare 17 and 24 since 17 < 24 those elements will be swapped. Then it will check and 17 is greater than 1 so it will stay in place. The comparison counter will increment twice.

{1,17,24,8,5,29}

{1,17,8,24,5,29}

{1,8,17,24,5,29}

Now we will compare 24 and 8 since 24 is greater than 8, those elements will be swapped. After we compare 8 and 17 since 8 is lower than 17 those elements will be swapped. Before moving to the next element we do one more comparison on 8 and 1 and since 8 is greater than 1 the elements will stay in place. The comparison counter will increase by three.

{1,8,17,24,5,29}

{1,8,17,5,24,29}

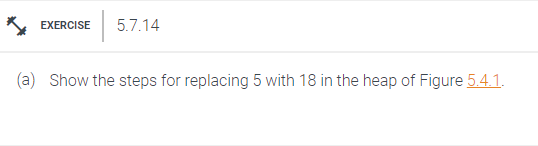
{1,8,5,17,24,29}

{1,5,8,17,24,29}

Moving onto the next element we compare 5 and 24. 5 is less than 24 so those elements will be swapped. Next compare 5 and 17 since it is less than 17 the elements will be swapped. Compare 5 and 8 since 5 is less than 8 then those elements will be swapped. Lastly compare 5 and 1 since 5 is greater than 1 then the elements will stay put. The comparison counter will be incremented by four. Giving the total amount of comparisons to be 10. As the size of the list increases so will the number of totaL comparisons.

In summary, as each element is processed, it will first be put to the front of the sequence, and then gradually go backward in the series. There will be n moves made to each element as a result. This implies time overall for n elements because there are n elements in total.

Problem 5.7.14



Answer:

The Heap in Figure 5.4.1



First replace 18 for 5.



Next do a min heapify on node 18. Since 18 is greater than 9 the right child the elements will be swapped.



After it would see that 18 is greater than 12 so those nodes would be swapped.



Since the min heapify will be satisfied after this step. Above is the resulting heap.