Complexity- and test analysis LAB2 DAT037

Ali Mohamed & Ali Mohamud

Complexity Analysis

It will take constant time to insert a new bid containing name, buy/sell and price. But since we are using a sorted priority queue also known as binary heap, we must ensure that a child never is less then its parent. The solution to this is using Bubble sort algorithm. The algorithm compares a child with its parent to see if the child is smaller in value then its parent, if this is the case then the algorithm will swap these to values and continue until it reaches the root or iff the next element is smaller in value then the previous. This implementation has the time complexity $O(\log n)$

To delete the smallest element in sorted list/array takes $O(n)$, but in our case this operation takes $O(1)$. This takes constant times because we swap the content between index 0 and the last index in the ArrayList, thus making the removal constant. Since this process modifies the heap structure we must implement bubble algorithm mentioned earlier, that has the time complexity $O(\log n)$.

To modify a certain element takes constant time. But the same can't be said for the time it takes to find this specific element, since we are using a function, findValue(), that loop through the hole array to compare a arbitrary element with the content of the index. The time complexity for the function is $O(n)$. But since this operation modifies the heap structure we must implement Bubble algorithm that bubbles up if the new value is larger then the previous value and vice versa. The time complexity for this operation is $O(n) + O(\log n) = O(n)$.

When the program has tried to make a transaction between every buyer and seller the constructor calls a method named orderBook(), whose purpose is to print the remaining byers and sellers. To accomplish this, we used a loop that calls deleteMin() for every element in the priority queue. Since deleteMin() is bubble algorithm the time complexity is $O(\log n)$. The time complexity for the loop $\sum_{i=0}^{\text{heap.size()-1}} x_i$ is $O(n)$. The total time complexity is $O(\log n)O(n) = O(n \log n)$

$$\text{findValue(E e)} = \sum_{i=0}^{n} x_i = O(n) \tag{1}$$

$$\begin{aligned} \text{(Bubble algorithm) } n &= 2^k - 1 \\ log(n) &= k \log 2 - 1 \\ log(n) &= k \log 2 \\ \frac{\log n}{\log 2} &= k \\ O(\log n) &= k \end{aligned} \tag{2}$$

Test analysis

When we have written any method we have tested it to see if it behaves as expected. For example the bubble sort which is two methods bubbleUp() and bubbleDown()- we have tested it to see if the tree becomes sorted by using the insert and deleteMin method. Other methods were tested in the same way. Our program can take multiple bids from the same user but handles them independent of each other. If a user who have multiple bid want to change the bid the program changes only the last one.

The input for the test were:
Bengt S 71
Cecilia S 70
Bengt NS 71 72
David K 72
Erik S 73
Frida S 65
Gustaf K 69
Hanna S 72
Erik S 71
Åke S 70
Erik NS 71 68
Ali K 72
Håkan S 71
Bert K 69
Bo S 72
David K 30
David K 30
David NK 30 31
****

The expected output:
David köper från Cecilia för 72 kr
Gustaf köper från Frida för 69 kr
Ali köper från Erik för 72 kr

Orderbok
Säljare: Åke 70, Håkan 71, Hanna 72, Bo 72, Bengt 72, Erik 73,
Köpare: Bert 69, David 31, David 30,
****

The actual output: David köper från Cecilia för 72 kr
Gustaf köper från Frida för 69 kr
Ali köper från Erik för 72 kr

Orderbok
Säljare: Åke 70, Håkan 71, Hanna 72, Bo 72, Bengt 72, Erik 73,
Köpare: Bert 69, David 31, David 30,

As you can see it orders the heaps perfectly and makes the transactions accordording to which element is in the root of heaptree.