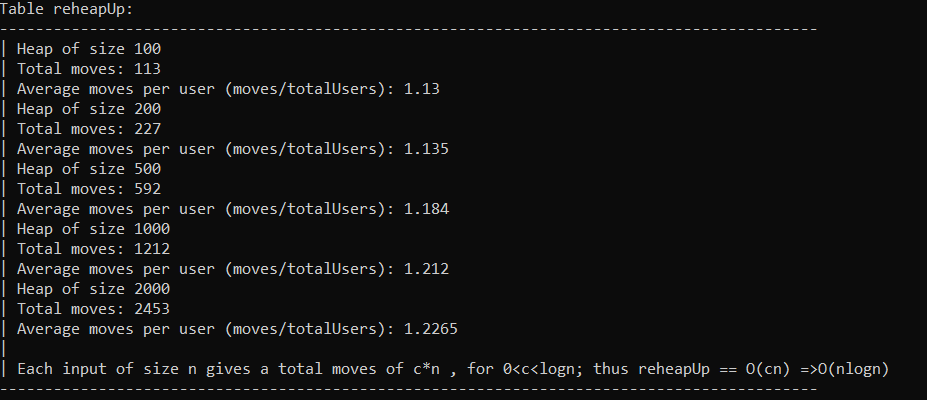
明文

0416329

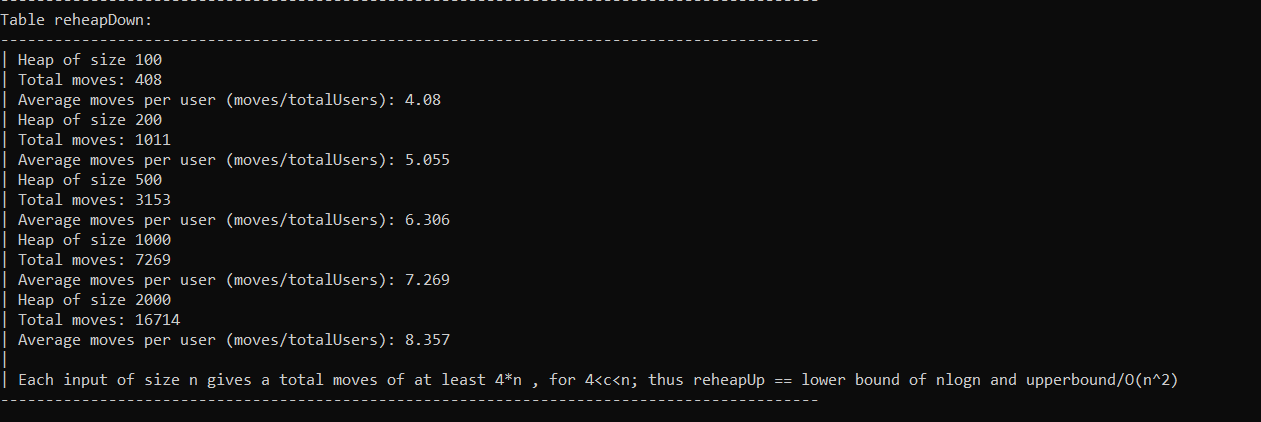
To derive my Upper bound (Big O) calculations for either reheapUp or reheapDown functions I took my data moves to be equal to swaps. First for the reheapUp function on any of the 5 sizes of heaps the amount of moves where at least log n. This would normal happen only on the heap of size 100 where the amount of data moves could be <100, thus as I would say this function has a lower bound of log n.

For the upper I would calculate the average moves per heap size.

Here are some example results:  


The amount of moves is n times a constant c, where I will say not the most exact range but I left it in the range (0,log n) thus there are a total of n \* log n moves at most thus a O(nlogn).

For the reheap down I used the same process, same results here:



You can tell reheap down for an input size n it takes at least 4\*n data moves thus I would at least give it a lower bound of nlogn but as you can everytime we doubled the input size so did the number of moves doubled. Thus if given a sufficiently large input the function would no more than n^2 data moves. For example on the heap of size 2000 the amount of moves is way past nlogn but even near n^2. The amount of data moves will keep increasing but nowhere near to n^2 thus we can give it a not so tight upper bound of O(n^2).

From the graph we can tell that building a heap will always be relatively efficient as input grows but when deleting a node , as the input size grows the more complicated is to reorganize our heap.