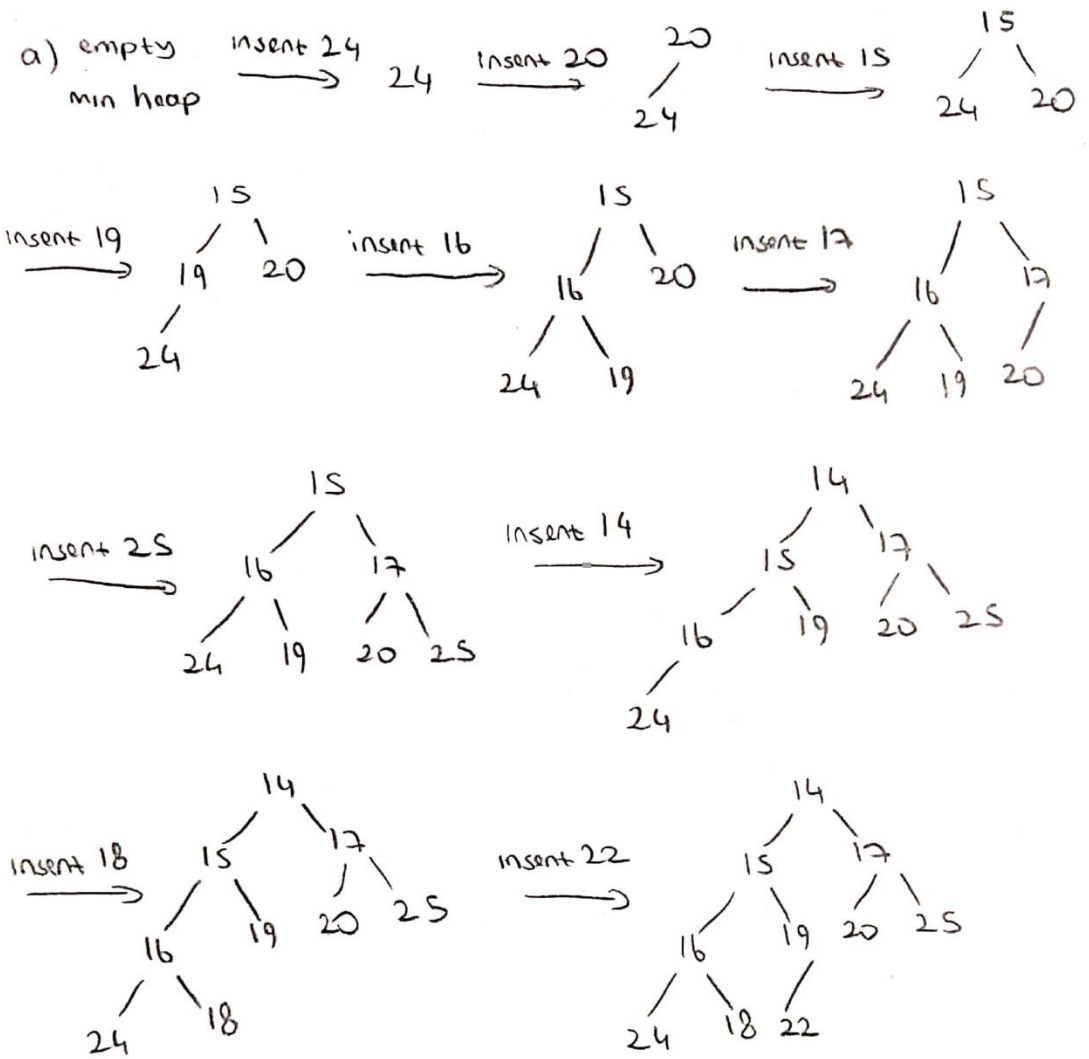
Cemhan Kaan Özaltan

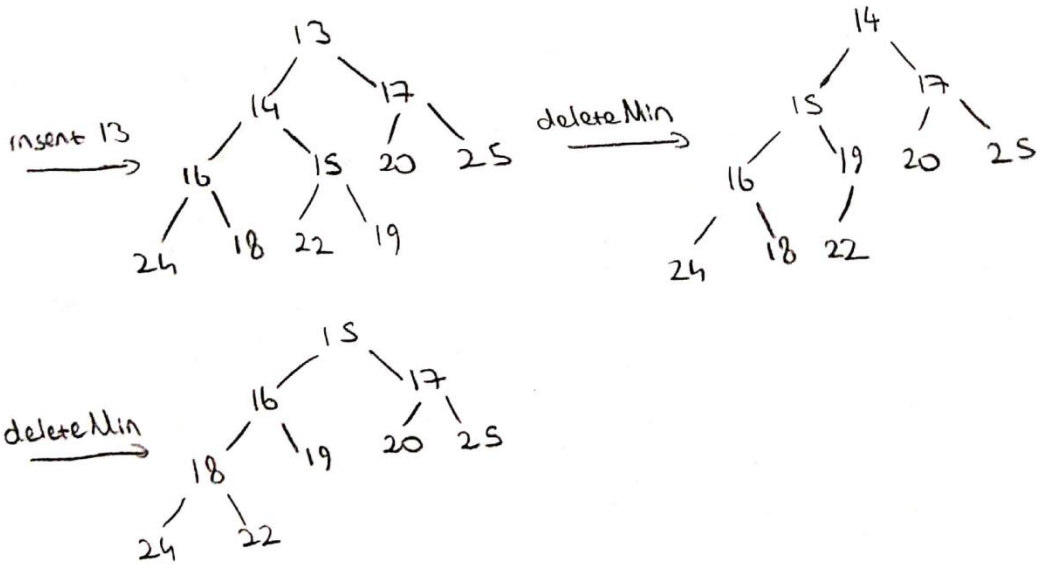
21902695

Section 3

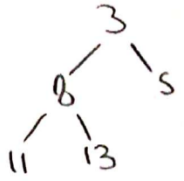
Homework 3 Report

Question 1





b) Let us consider the following min heap:



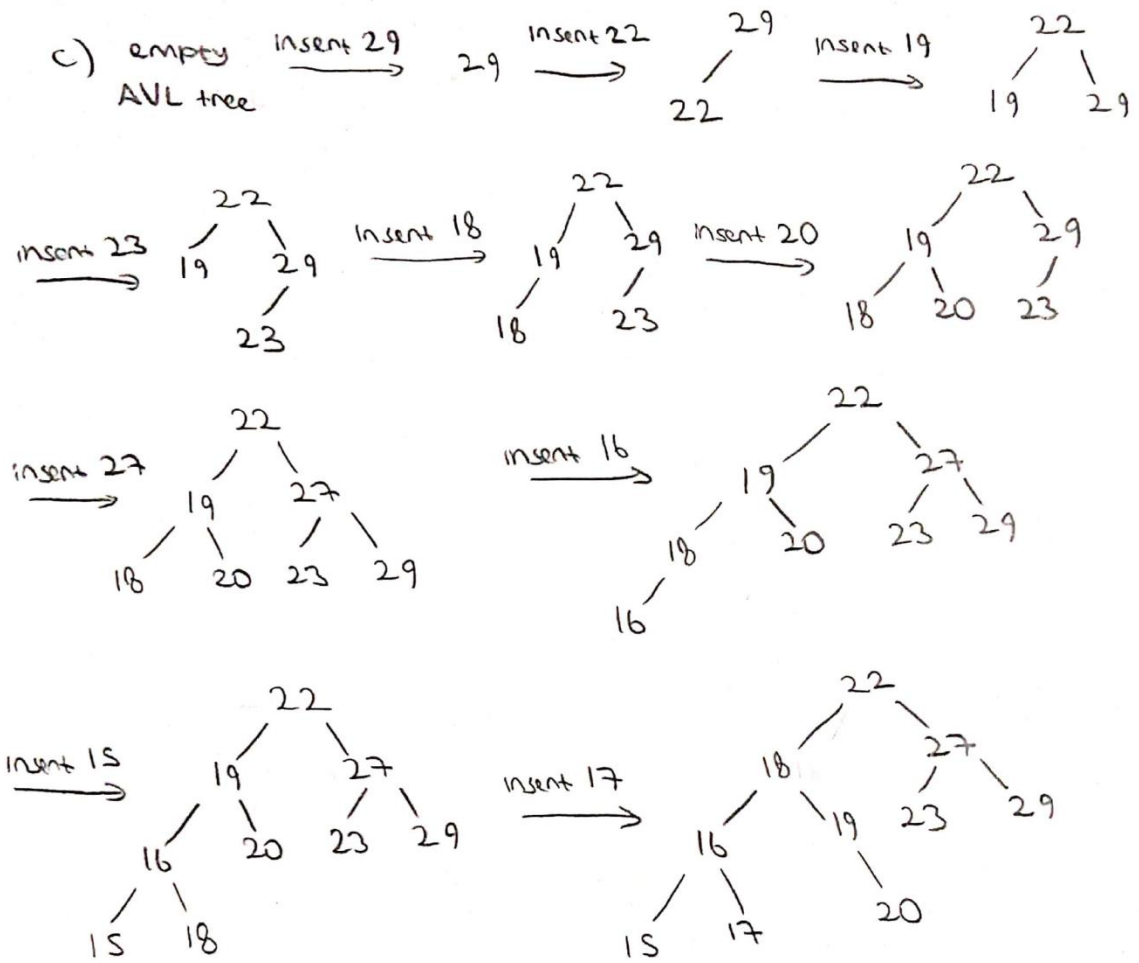
Its preorder traversal is 3, 8, 11, 13, 5. This traversal is not sorted since it would have to be 3, 5, 8, 11, 13 to be sorted. This is because preorder traversal only yields sorted output when all children are on the right side for binary search trees, which is not supported by the min heap structure since it has to be a complete binary tree.

Its inorder traversal is 11, 8, 13, 3, 5. This traversal is not sorted for the same reason. This is because the parent must be smaller than its children in min heaps. Because of this, no node's left child can be smaller than it, which is necessary for sorted inorder traversal, and therefore inorder traversal which normally gives sorted output for binary search trees does not do the same for min heaps.

Its postorder traversal is 11, 13, 8, 5, 3. This traversal is not sorted for the same reason. This is because postorder traversal only yields sorted output when all children are on the left side for binary search trees, which is not supported by the min heap structure since it has to be a complete binary tree and nodes cannot have smaller children than themselves which happens for this case in binary search trees.

There are special cases with special min heaps for the above traversals where sorted output may be given but the average behavior is as described. Also, we considered being sorted in increasing order. For decreasing order, the corresponding versions of the above cases are valid.

c) empty AVL tree



Question 3

In the case of very large printer and print request numbers, the currently used algorithm would not be feasible. This is because, the simulate function starts from one printer and keeps iterating until the desired average waiting time is reached, making it O(n). Moreover, the simulateCase function uses loops to iterate all printers in each minute, which are inside a loop that iterates time until all requests are processed. Since the time loop's iteration count and heap operations are not directly proportional to the total printer or request count, simulateCase is roughly O(n), although it may be slower in the worst case. This makes the entire algorithm $O(n^2)$ and therefore not usable with large inputs. A better strategy in this case would be to start the simulation printer count (K) at total printer count's (N) medium value which is N/2 since we know the number of total printers. If the simulation gives a lower average waiting time than desired, it will be repeated with the medium value of the lower half as K. If it gives a higher average waiting time than desired, it will be repeated with the medium value of the upper half as K. This would be repeated until the number of printers for the desired average waiting time is found. Here, we are doing something similar to binary search and the simulate function's time is growing logarithmically with time complexity O(logn) in contrast to the linear growth of the initial case. Therefore, the entire algorithm becomes O(nlogn) and usable with large inputs.