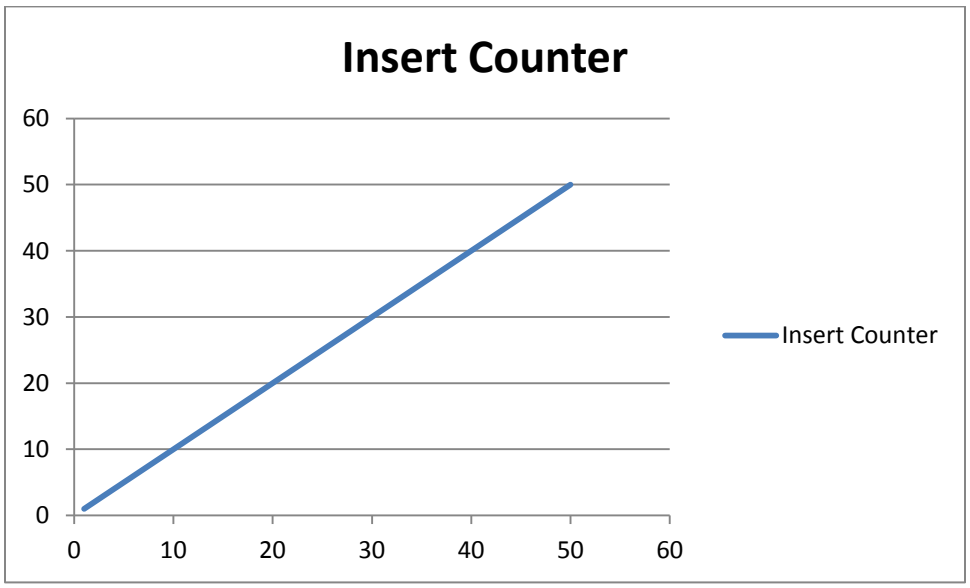
CS 583 – Fall 2013, Shaeq Khan G00774501

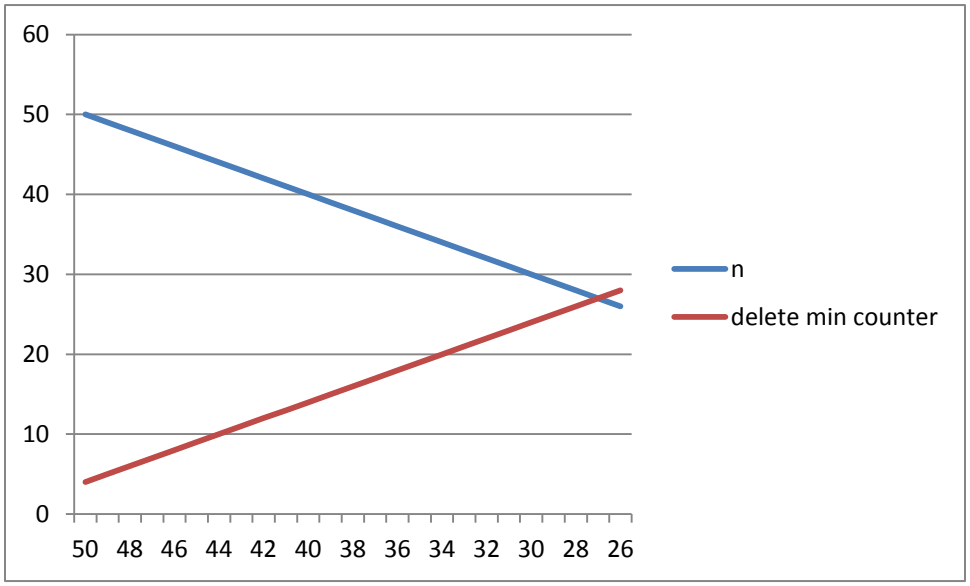
Programming Assignment – Violation Heaps

Analysis

insert(x,h): the insert operation has a linear graph because each insert operation on the heap charges the counter by one. As the number of nodes in the heap increases the counter increments by one which essentially means that we are doing this operation in constant time.

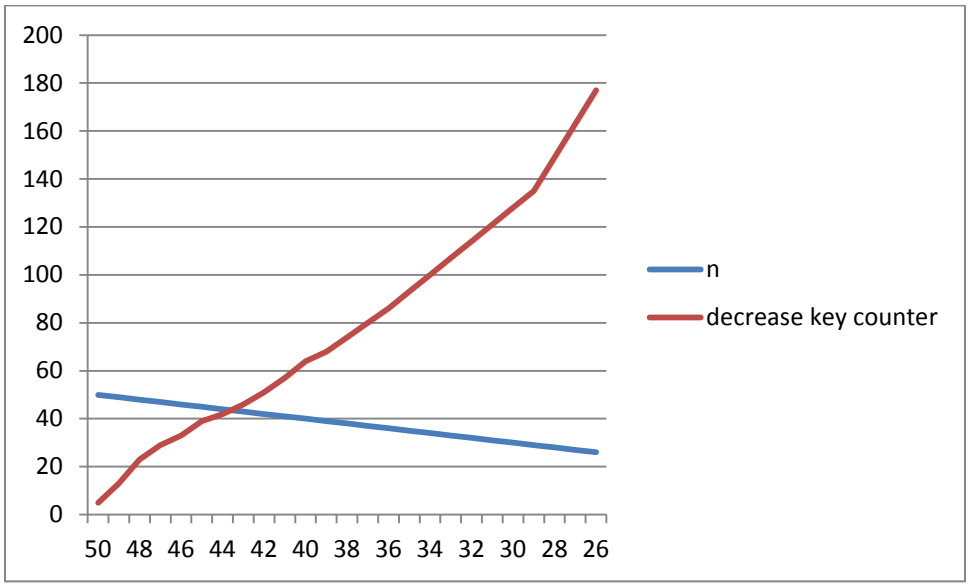


delete-min(h): the delete-min operation on the heap charges the counter by one. As the number of nodes decrease after each operation, the counter increment forms a linear graph. Again, this implies that delete-min operation is achieved in constant time for small size of a heap.



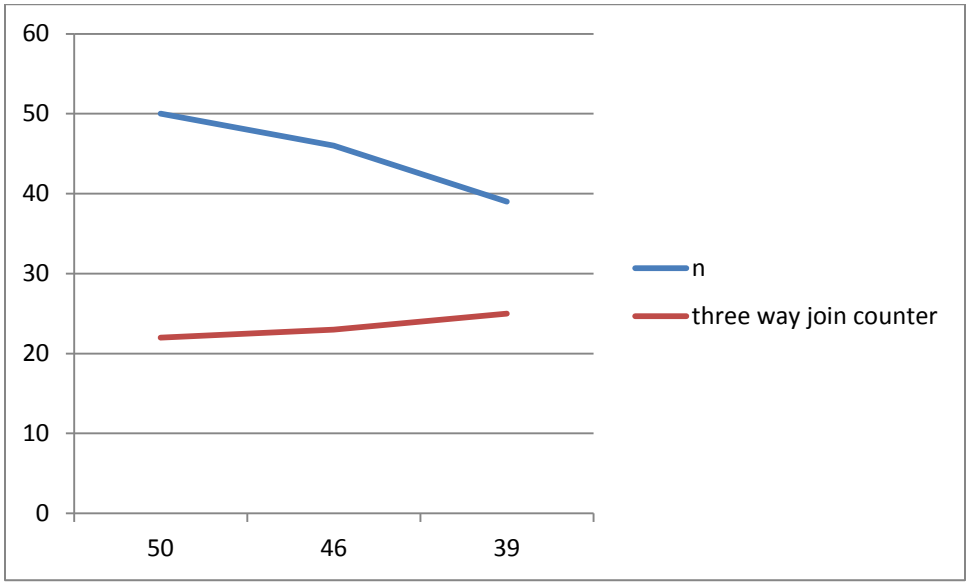
Delete Min Counter

decrease-key(delta,x,h): this operation closely increases in a linear fashion, with the decrease in the number of keys in the heap after each delete-min, the counter takes nearly the same amount of intervals to perform a decrease key the nodes in the heap. Some variations can be noticed due to different number of times we loop inside the function depending on how we find the node we were looking for.



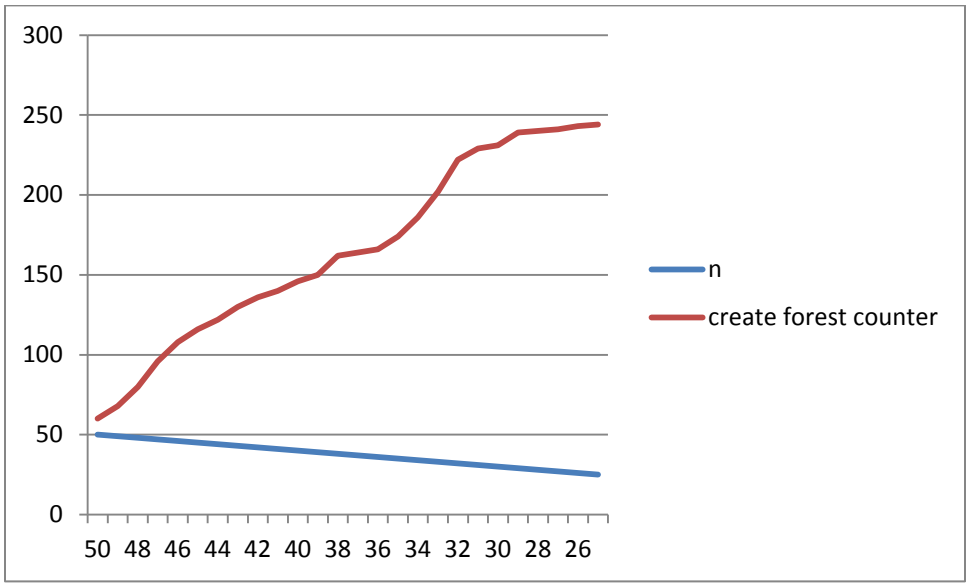
Decrease Key Counter

3-way-join(z,z1,z2): the counter for this method starts from a relatively high value showing the amount of three way joins performed initially after the delete-min to satisfy the violation property of the heap. After this, the counter pops up only a couple more times to get rid of three or more nodes with the same rank in the root and wraps up quickly shown by the slight increase in counter value.



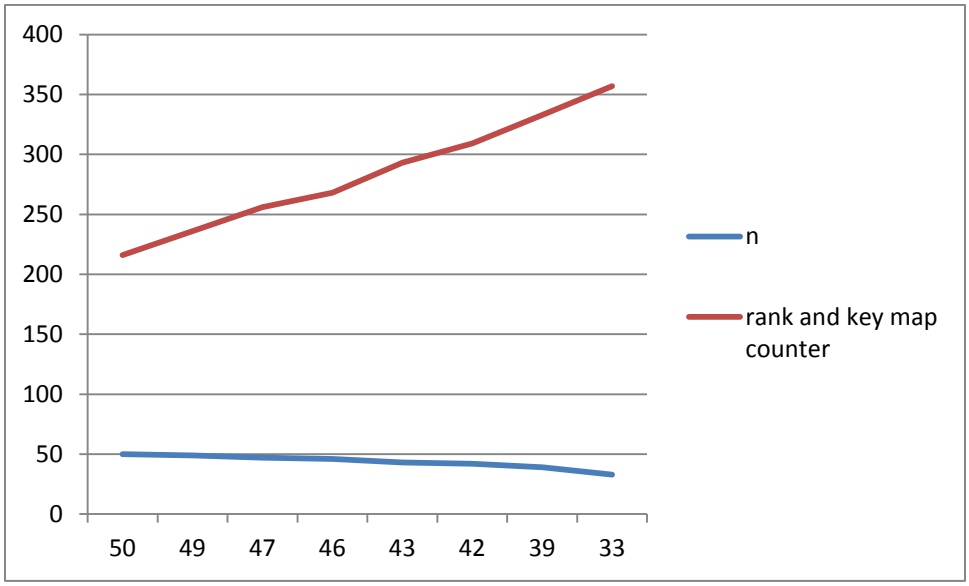
Three Way Join Counter

createForest() – this was a method I used that took a list of ViolationHeapNodes as an argument and return a root structure (creating a forest by joining these trees together) while assigning the minNode for the heap at the same time. This method was used by the delete-min and decrease-key operations. The counter was dependent on the number of roots in the heap that was to be created in O(lg n) time. We can see a logarithmic structure to the curve as the number of nodes decrease in the heap.



Create Forest Counter

rankAndKeyMap() – I used this function to create a HashMap of the ranks and the ViolationHeapNodes in the root of the heap. This helped to determine which nodes in the root are in violation and need to undergo a three-way-join until no node is under a violation. The initial counter cost to create the HashMap was big since there were a lot of nodes with the same rank i.e. zero for the first delete-min. But then on, the counter increments showed up only for a few more violations that occurred with some noticeable increments which was basically the cost to iterate over the root list in no more than O(lg n) time.



Rank and Key Map Counter