Oliver Benning
7798804

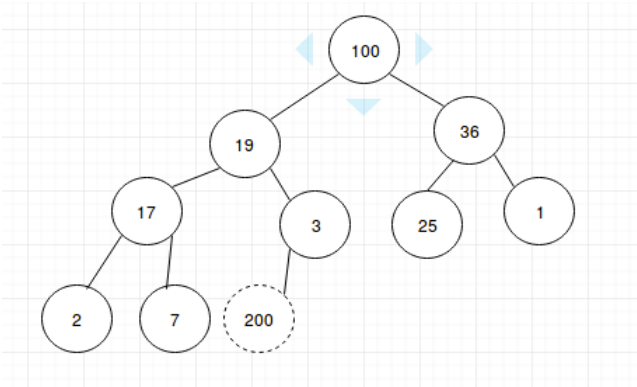
CSI 2110 Assignment 3

Question 1:

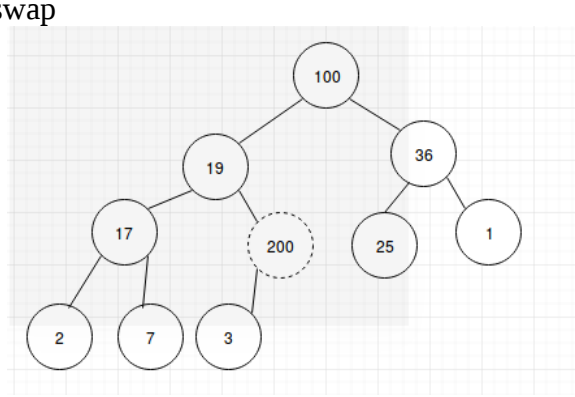
a) 100 – 19 – 17 – 2 – 7 – 3 – 36 – 25 – 1

b) 2 – 7 – 17 – 3 – 19 – 25 – 1 – 36 – 100

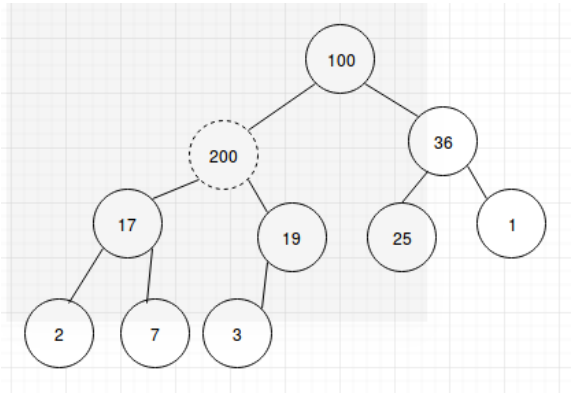
c) Insert at tail



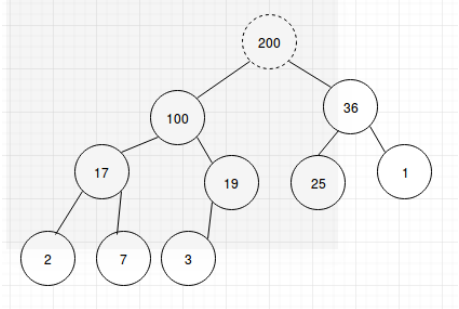
Upheap, first swap



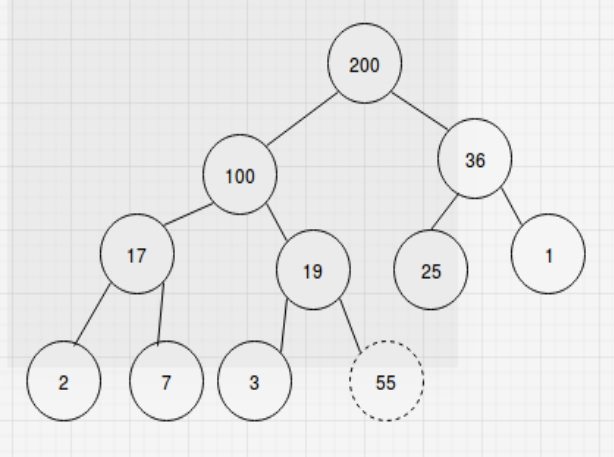
Second Swap



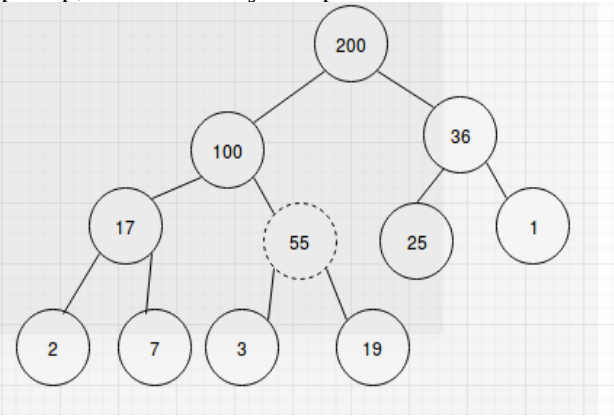
Final Swap, 200 is now inserted



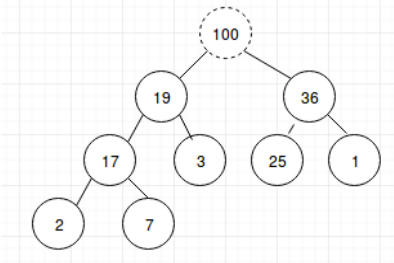
Insert 55 at tail



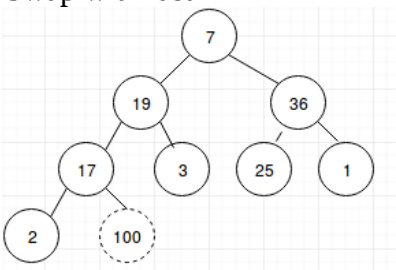
Commence upheap, first and only swap. 55 is now successfully inserted.



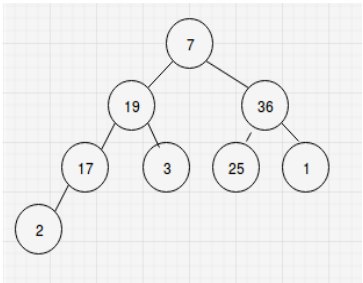
d) Select Max



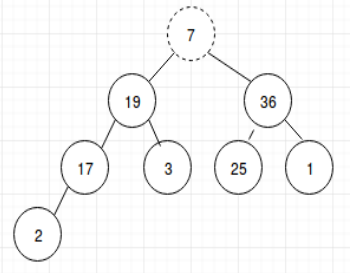
Swap with last

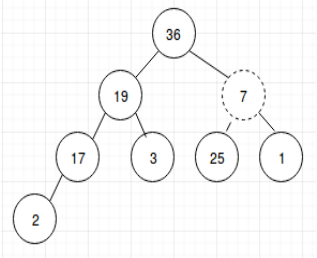


Remove

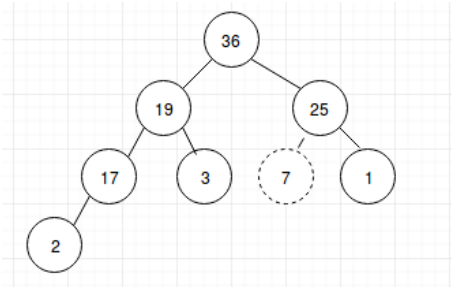


Downheap root





Finished downheap, and therefore removal



Question 2:

It would take 3 comparisons. The first minimum key is simply the root, no comparisons needed. To find the next smallest key, you'd need to compare the smallest of the root's two nodes, one comparison. To find the third element you'd need to find the smallest of the second smallest node's children, and then compare that element with the child of the root not previously selected, you'd then find the last element with two comparisons. In total you'd need 3 comparisons to find the 3 smallest keys.

Question 3:

a) Because a heap is a complete binary tree, the height in relation to the # of nodes is $h = \lfloor \log_2 n \rfloor$
Therefore in this case $h = \lfloor \log_2 5000 \rfloor = \lfloor 12.28.. \rfloor = 12$ so the height is 12.

b) I know that the number of nodes *excluding* those at the last level is $2^h - 1$ which is in this case $2^{12} - 1 = 4095$ therefore the number of nodes at the deepest level is $5000 - 4095 = 905$ nodes.

c) The number of nodes at the second deepest level will be a complete power of 2, equal to 2^{h-1} .
In this case we calculate it to be $2^{12-1} = 2^{11} = 2048$ nodes.

d) In the absolute worst case, we would insert an element at the tail which is smaller than the element at the root. It would thus be smaller than each of it's parent nodes and would need to be swapped once through each level of the tree in it's upheap. Therefore in the worst case we would need precisely 12 swaps.

e) In the best case, we insert an element at the tail which is already larger than it's parent element. In this case no swap would be necessary therefore at best 0 swaps would be required.

f) Worst case would be checking every single node in the tree, which is 5000 comparisons.

Question 4:

a)

```
void downheap (int[] heap, int N, int k) {
        int left = 2*k;
        int right = 2*k + 1;
        int max = k;

        // Check left first if of bounds, then whether or not it is bigger
        if ( left <= N && heap[left] > a[k] ) {
                max = left;
        }
        // Same check for right element
        if ( right <= N && heap[right] > a[max] ) {
                max = right;
        }
        // Swap the biggest node if not already the root, then call recursively on swapped
```

```
        // the newly swapped position
        if (max != k) {
                int temp;
                temp = heap[max];
                heap[max] = a[k];
                heap[k] = temp;
                downheap(heap, n, max);
        }
}
```

b)
_ 4 11 7 12 32 8 12 21 2
_ 4 11 7 21 32 8 12 12 2
_ 4 11 12 21 32 8 7 12 2
_ 4 32 12 21 11 8 7 12 2
_ 32 21 12 12 11 8 7 4 2
Finish build heap
_ 21 12 12 4 11 8 7 2 32
_ 12 11 12 4 2 8 7 21 32
_ 12 11 8 4 2 7 12 21 32
_ 11 7 8 4 2 12 12 21 32
_ 8 7 2 4 11 12 12 21 32
_ 7 4 2 8 11 12 12 21 32
_ 4 2 7 8 11 12 12 21 32
_ 2 4 7 8 11 12 12 21 32
FINAL (sorted)
_ 2 4 7 8 11 12 12 21 32

Question 5:
a) To find a sequence, we need to check if its first element exists and if it does we visit that node, this is one step. Then we check if the next element exists and if so visit that node, 1 step. We repeat that last step recursively until the last element, which makes N repetitions. Therefore to find a sequence we'd need to make N repetitions of 1 comparison, which is N comparisons. The search is therefore O(N).

b) For the longest sequence of length zero all we have is a null root element, which is $1=4^0$ elements. For a longest sequence of 1, we add 4 elements, $4=4^1$. For a sequence of two we add 4 elements for each of the previous elements or $16=4^2$.

The resulting sum, $4^0+4^1+4^2+...+4^h$ where h is the height can be written as $n=\sum_{i=0}^{h} 4^i$ which can be equivalently represented as $n=\frac{4}{3}(4^h-1)+1$

c) It would be N + 1. Each tree layer would have one node, including the first root node. This would give N nodes plus the one root node, N + 1.

d) "A" , "CAA", "CCAAA" , "CCAAT" , "CCAA", "CCC" , "CCG" , "CC" , "T".