Name: Pheakdey Luk
ID: 986591

Assignment 4

R-2.8 Illustrate the performance of the selection-sort algorithm on the following input sequence (22, 15, 26, 44, 10, 3, 9, 13, 29, 25).

Answer:

- Step 1: **<u>22</u>**, 15, 26, 44, 10, **3**, 9, 13, 29, 25
- Step 2: 3, **<u>15</u>**, 26, 44, 10, 22, **9**, 13, 29, 25
- Step 3: 3, 9, **<u>26</u>**, 44, **10**, 22, 15, 13, 29, 25
- Step 4: 3, 9, 10, **<u>44</u>**, 26, 22, 15, **13**, 29, 25
- Step 5: 3, 9, 10, 13, **<u>26</u>**, 22, **15**, 44, 29, 25
- Step 6: 3, 9, 10, 13, 15, **<u>22</u>**, 26, 44, 29, 25
- Step 7: 3, 9, 10, 13, 15, 22, 26, 44, 29, 25
- Step 8: 3, 9, 10, 13, 15, 22, <u>26</u>, 44, 29, 25
- Step 9: 3, 9, 10, 13, 15, 22, 25, <u>44</u>, 29, 26
- Step 10: 3, 9, 10, 13, 15, 22, 25,26 <u>29</u>, 44
- Step 11: 3, 9, 10, 13, 15, 22, 25,26 29, <u>44</u>
- Step 12: 3, 9, 10, 13, 15, 22, 25,26 29, 44

Selection sort find minimum element and swap it with the first element of unsorted section.
Runtime of finding minimum element= O(n).So, for n element the run time is = $O(n^2)$.

R-2.9 Illustrate the performance of the insertion-sort algorithm on the input sequence of the previous problem.

- Step 1: 22, 15, 26, 44, 10, **3**, 9, 13, 29, 25
- Step 2: **15**, 22, 26, 44, 10, **3**, 9, 13, 29, 25
- Step 3: **10**,15, 22, 26, 44, **3**, 9, 13, 29, 25
- Step 4: **3**, 10,15, 22, 26, 44, 9, 13, 29, 25
- Step 5: 3, **9**, 10,15, 22, 26, 44, 13, 29, 25
- Step 6: 3, 9, 10, **13**, 15, 22, 26, 44, 29, 25
- Step 7: 3, 9, 10, 13, 15, 22, 26, **29**, 44, 25
- Step 8: 3, 9, 10, 13, 15, 22, **25**, 26, 29, 44
- Step 9: 3, 9, 10, 13, 15, 22, 25, 26, 29, 44

In insertion sort, we take an unsorted element and swap it to left until it find a proper position, which takes O(n) time. So, for n element, it takes $O(n^2)$ time.

R-2.10 Give an example of a worst-case sequence with n elements for insertion-sort runs in $\Omega(n^2)$ time on such a sequence.

Answer:
44, 29, 26, 25, 22, 15, 13, 10, 9, 3

R-2.13 Suppose a binary tree T is implemented using a vector S, as described in Section 2.3.4. If n items are stored in S in sorted order, starting with index 1, is the tree T a heap? Justify your answer.

Answer:

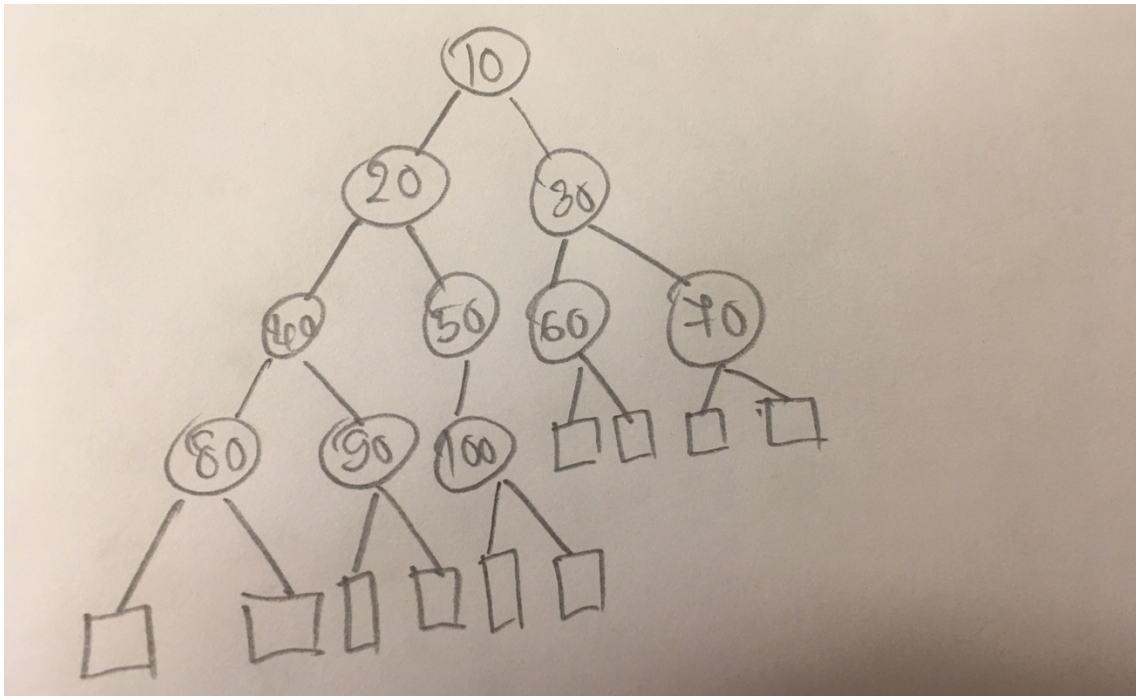
Yes, it is.

Justification:

Vector, S with 10 items in ascending order.

0	1	2	3	4	5	6	7	8	9	10
	10	20	30	40	50	60	70	80	90	100

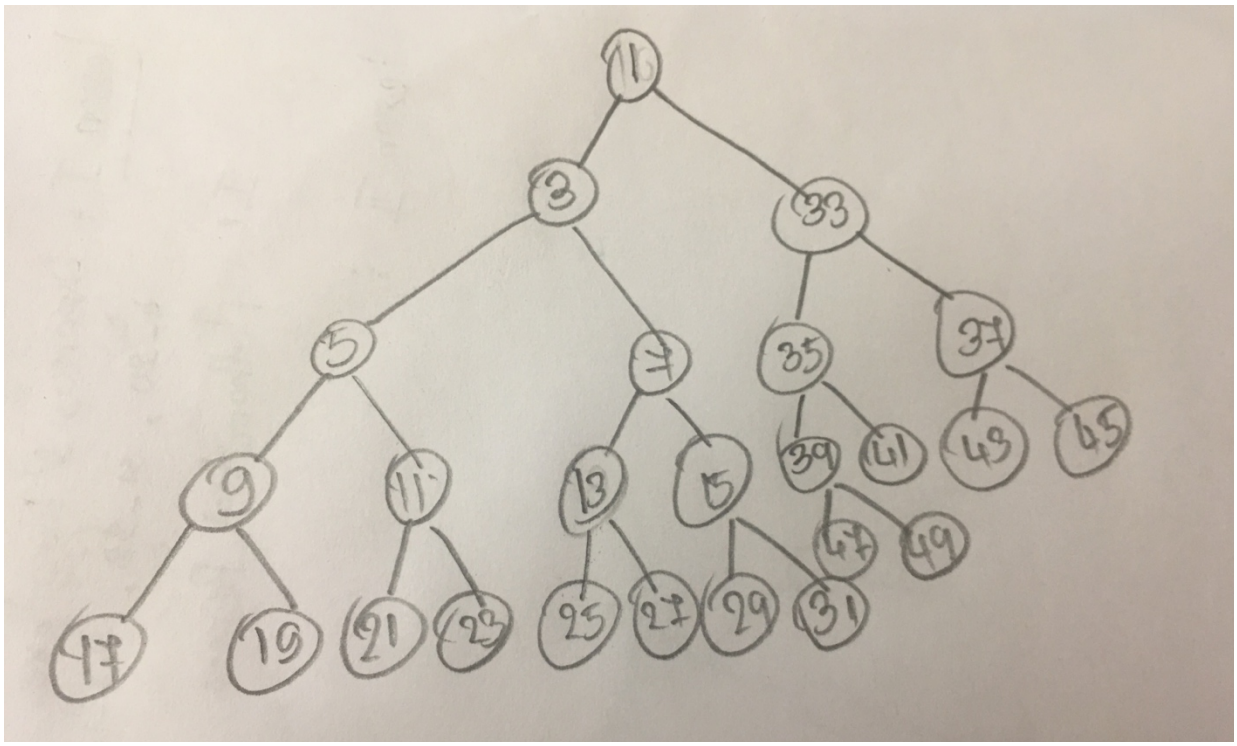
If we draw a tree, it will be like this.



As we know that, for heap

a. For all nodes except root key(child) >= key(parent).This tree maintain this property.
b. Root has minimum value. Here, minimum value 10 is in the root.
c. Subtrees are also heap and values are in ascending order from root to leaf

R-2-18 Draw an example of a heap whose keys are all the odd numbers from 1 to 49 (with no repeats), such that the insertion of an item with key 32 would cause up-heap bubbling to proceed all the way up to a child of the root (replacing that child's key with 32).



C-2.32 Let T be a heap storing n keys. Give an efficient algorithm for reporting all the keys in T that are smaller than or equal to a given query key x (which is not necessarily in T). For example, given the heap on Figure 2.41 and query key x=7, the algorithm should report 4, 5, 6, 7. Note that the keys do not need to be reported in sorted order. Ideally, your algorithm should run in O(k) time, where k is the number of keys reported.

Answer:

```
Algorithm getSmallerEqualKey(T,key,i)


Input: Vecotr T representing a Heap,index i of an element in the heap and a
query key x
Output: Vecotr V contining all the keys which are less than or equal to given
key
    if i < T.size() ^ T.elementAtRank(i) <= key
        V.insertLast(T.elementAtRank(i))
        getSmallerEqualKey(T,key,2 * i)
        getSmallerEqualKey(T,key,2 * i + 1)

        return
```

Design an algorithm, isPermutation(A,B) that takes two sequences A and B and determines whether or not they are permutations of each other, i.e., same elements but possibly occurring in a different order. Hint: A and B may contain duplicates.

What is the worst case time complexity of your algorithm? Justify your answer.