assignment-4

Friday, 1 November 2024 11:14 PM

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).

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

- Selection sort finds the minimum element & swap it with the first element of the unsorted section.
- The time Complexity for each Operation is $O(n)$, so the total time Complexity 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:	15	22	26	44	10	3	9	13	29	25
step 4:	15	22	26	44	10	3	9	13	29	25
step 5:	10	15	22	26	44	3	9	13	29	25
step 6:	3	10	15	22	26	44	9	13	29	25
step 7:	3	9	10	15	22	26	44	13	29	25
Step 8:	3	9	10	13	15	22	26	44	29	25
step 9:	3	9	10	13	15	22	26	29	44	25
Step 10:	3	9	10	13	15	22	25	26	29	44

- In insertion sort we take element by element and insert it in the correct order.
- And this take $O(n)$ per Operation, it means it takes $O(n^2)$ for the entire Course.

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.

- The worst-Case sequence For Insertion Sort occurs When the sequence is sorted in reverse order.
  [10, 8, 7, 5, 3, 2]

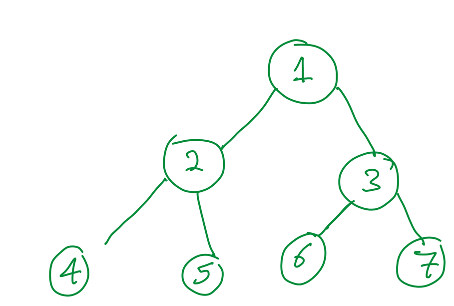
In this Case, every new element needs to be Compared with all previously sorted elements resulting in $\Omega(n^2)$ time Complexity

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.

Yes. the tree is Heap.

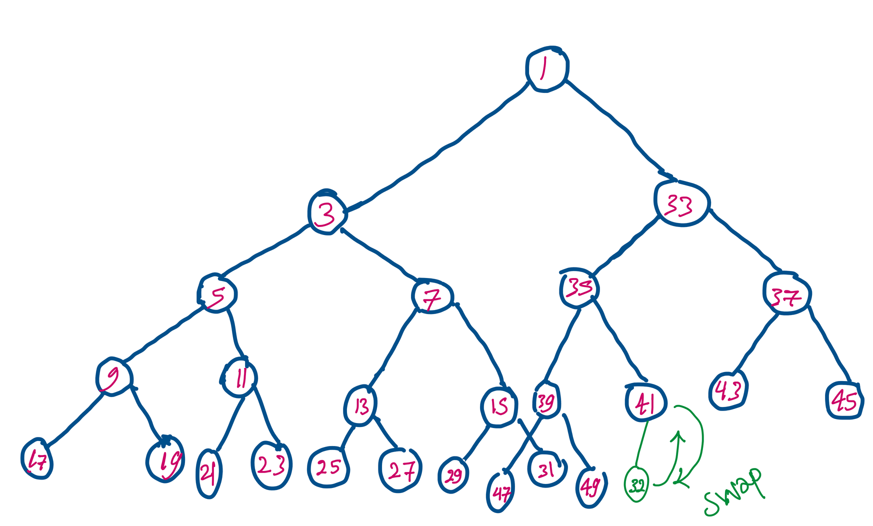
Suppose vector S which Consists of n elements with sorted order from index 1

	1	2	3	4	5	6	7			
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]

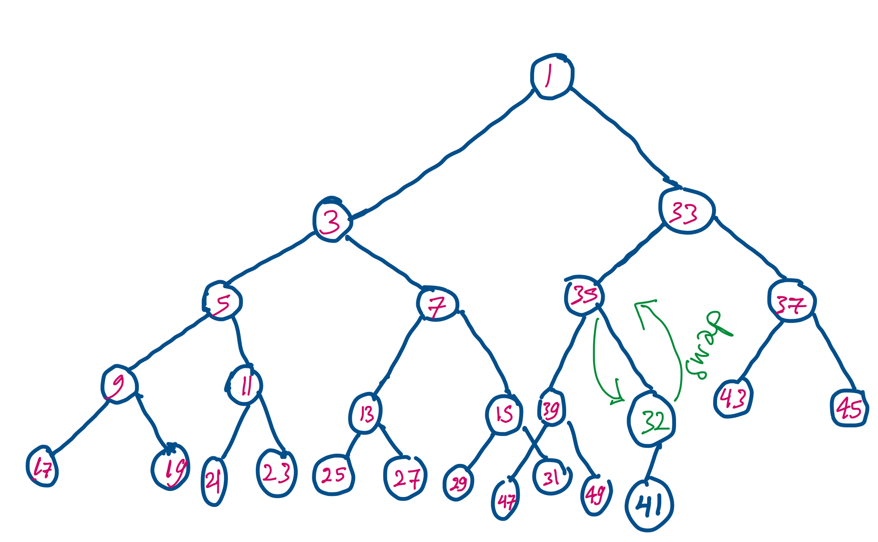


We know for heap order, for every Internal node v
$K(v) >= Key(parent(v))$
This means child will be always greater than parent.

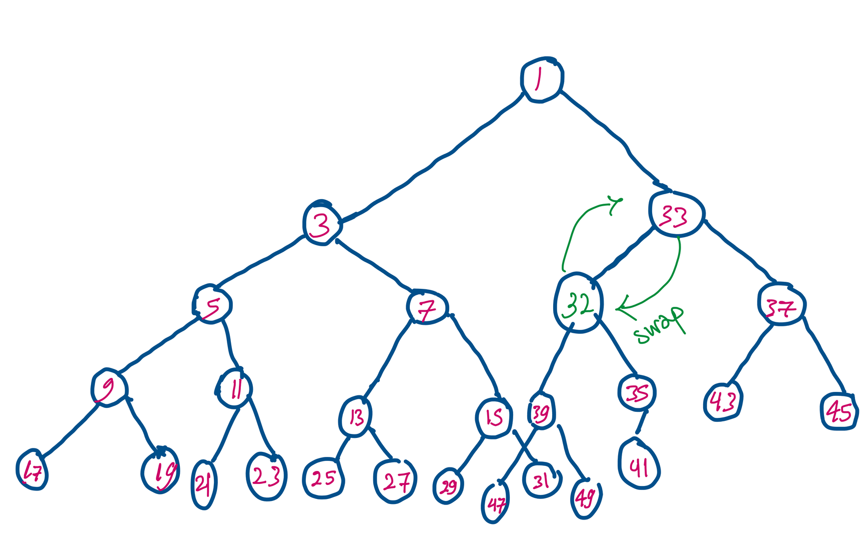
R-2-18 Draw an example of a heap whose keys are all the odd numbers from 1 to 49 (with no repeats), such that the insertion of an item with key 32 would cause up-heap bubbling to proceed all the way up to a child of the root (replacing that child's key with 32).



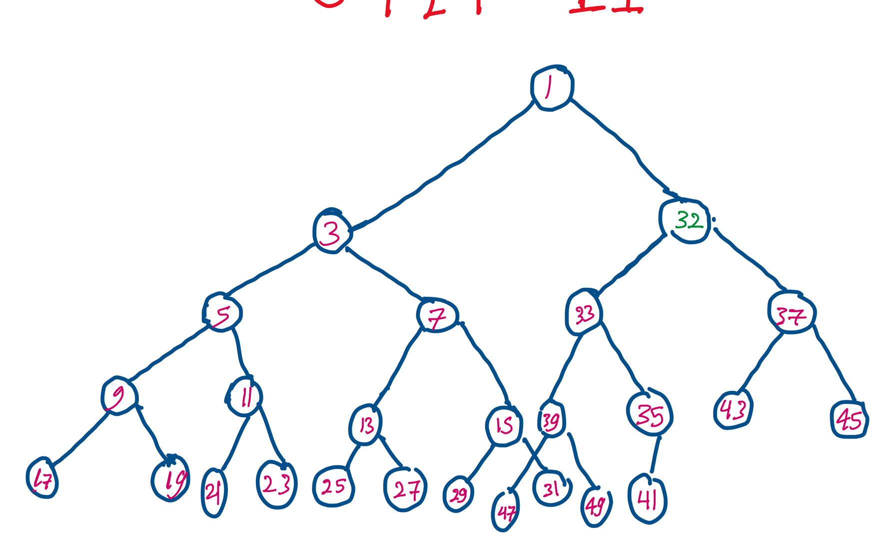
STEP-I



STEP-II



STEP-III



STEP-IV

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.

Algorithm isPermutation (A, B)

```
if length(A) ≠ length(B)
    return false
Count_map = new empty hash map

for element in A:
    if element in Count-map
        Count-map[element] = Count_map[element] + 1
    else
        Count_map[element] = 1

for element in B:
    if element not in Count-map or Count_map[element] = 0
        return 0
    else
        Count_map[element] = Count_map[element] - 1

return true
```