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:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 22 |  |  |  |  |  |  |  |  |  |
| 15 | 22 |  |  |  |  |  |  |  |  |
| 15 | 22 | 26 |  |  |  |  |  |  |  |
| 15 | 22 | 26 | 44 |  |  |  |  |  |  |
| 15 | 22 | 26 | 44 | 10 |  |  |  |  |  |
| 10 | 15 | 22 | 26 | 44 |  |  |  |  |  |
| 10 | 15 | 22 | 26 | 44 | 3 |  |  |  |  |
| 3 | 10 | 15 | 22 | 26 | 44 |  |  |  |  |
| 3 | 10 | 15 | 22 | 26 | 44 | 9 |  |  |  |
| 3 | 9 | 10 | 15 | 22 | 26 | 44 |  |  |  |
| 3 | 9 | 10 | 15 | 22 | 26 | 44 | 13 |  |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 44 |  |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 44 | 29 |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 29 | 44 |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 29 | 44 | 25 |
| 3 | 9 | 10 | 13 | 15 | 22 | 25 | 26 | 29 | 44 |

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

Answer:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 22 | 15 |  |  |  |  |  |  |  |  |
| 15 | 22 | 26 |  |  |  |  |  |  |  |
| 15 | 22 | 26 | 44 |  |  |  |  |  |  |
| 15 | 22 | 26 | 44 | 10 |  |  |  |  |  |
| 10 | 15 | 22 | 26 | 44 |  |  |  |  |  |
| 10 | 15 | 22 | 26 | 44 | 3 |  |  |  |  |
| 3 | 10 | 15 | 22 | 26 | 44 |  |  |  |  |
| 3 | 10 | 15 | 22 | 26 | 44 | 9 |  |  |  |
| 3 | 9 | 10 | 15 | 22 | 26 | 44 |  |  |  |
| 3 | 9 | 10 | 15 | 22 | 26 | 44 | 13 |  |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 44 |  |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 44 | 29 |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 29 | 44 |  |
| 3 | 9 | 10 | 13 | 15 | 22 | 26 | 29 | 44 | 25 |
| 3 | 9 | 10 | 13 | 15 | 22 | 25 | 26 | 29 | 44 |

R-2.10 Give an example of a worst-case sequence with n elements for insertion-sort runs in Ω(n2) time on such a sequence.

Answer:

In case the list is sorted by descending, (44, 29, 26, 25, 22, 15, 13, 10, 9, 3).

It takes number of swaps: 1 + 2 + 3 + … = n(n+1)/2, it means O(n2)

Besides, it takes time for sorting 2 numbers, 3 numbers, …, i.e. O(n2)

Therefore, it takes at least Ω(n2) time

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 a heap if the sorted order is ascending when the order always ensures the heap-order property

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).

Answer:

The key 32 would cause up-heap bubbling to a child of the root if the root has a child node whose value is greater than 32, therefore 33 must be a root’s child. Herewith is the heap:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|  | 1 | 3 | 33 | 5 | 7 | 35 | 37 | 9 | 11 | 13 | 15 | 39 | 41 | 43 | 45 | 17 | 19 | 21 | 23 | 25 | 27 | 29 | 31 | 47 | 49 |

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.

|  |  |
| --- | --- |
| Algorithm reportKeys(T, x)  Input: T is Vector prepresenting a heap n elements, x is a query value  Output: keys is a list for storing keys    keys <- new array to store keys  if T.isEmpty() then  return keys    n <- T.root()  if n.element() > x then  return keys  else  keys.insertLast(n.element())    listNodes <- new list to store both internal and external nodes  listNodes.insertLast(n.leftChild())  listNodes.insertLast(n.rightChild())    n <- listNodes.first()  stop <- false  while stop = false do  if n.isExternal() = false then  if n.element() <= x then  keys.insertLast(n.element())  else  listNodes.insertLast(n.leftChild())  listNodes.insertLast(n.rightChild())  if listNodes.size() = 0 then  stop = true  else  n <- listNodes.after(n)    return keys | O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(k\*2)  O(k\*2)  O(k\*2)  O(k)  O(k\*2)  O(k\*2)  O(k\*2)  O(1)  O(k\*2)  O(1)  Total: O(k) |

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 A.size() != B.size() then  return false    H1 <- heapSort(A)  H2 <- heapSort(B)    n <- A.size()  for i<-0 to n-1 do  p1 <- H1.removeMin()  p2 <- H2.removeMin()  if p1.element() != p2.element() then  return false    return true |

The worst case is the A is sorted by descending and the B is sorted by ascending, or vice versa, because when building the heap, it will take time to move smaller numbers to the root.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 7 | 3 | 4 | 1 | 6 | 2 | 9 | 8 | 0 |