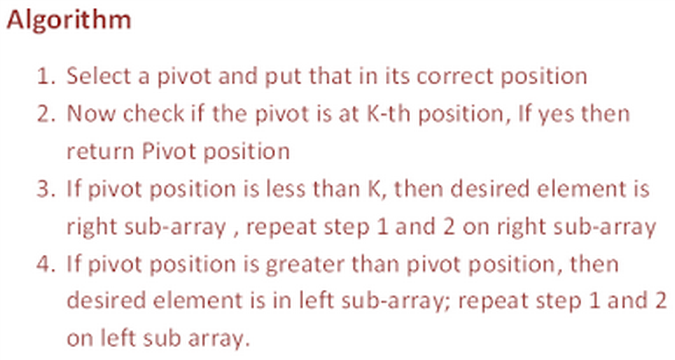
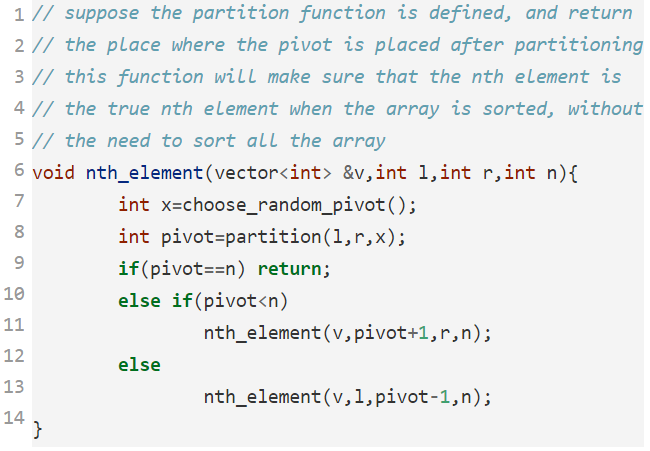
**Algorithm Permutation(arry, start, end)**

Input: arry is the charecter array of string, start and end is position of array

Output: array which contain all the permutation of array

|  |  |
| --- | --- |
| If start = end Then  print arry  Else  for j<-- start to j<=end do  Swap(arry[start], arry[j])  Permutation(arry, start +1, end)  Swap(arry[start], arry[j])  **Algorithm** Swap **(**a, b)  temp <-- a  a<-- b  b<--temp | O(1)  O(1)  O(1)  O(n)  O(n)  T(n(n-1(n-2....)))=O(n!)  O(n)  O(1)  O(1)  O(1)  Running time T(n) = O(n!) |





Partitioning has the complexity of O(n).

AVL tre or Not: prove

Height of the left and right subtrees may differ by at most 1.

Height of an empty tree is defined to be (–1).

The worst case time complexity of this method is O(n2), but it works in O(n) on average.

http://stackoverflow.com/questions/251781/how-to-find-the-kth-largest-element-in-an-unsorted-array-of-length-n-in-on.

1. .
2. .
3. .
4. .
5. .
6. .
7. .
8. .
9. .
10. .
11. . left child, right child, parrent
12. . left child, paren, right child