Assignment Submission

Name - Nirmit Chattoo
PRN - 2020BTEIT00001
File number 1: Algorithm and its explanation using code.

Quicksort Algorithm: QuickSort is a Divide and Conquer algorithm. A pivot element is picked and the array is partitioned around it. Different versions of quickSort pick pivot in different ways.
- Ensure that the first element is used as a pivot.
- Make the last element the pivot
- Choose a random element as a pivot.
- Pick the median as the pivot.

The key process in quickSort is partition(). Partitions aim to put x at the correct position in a sorted array, and put all smaller elements (smaller than x) before x, and all greater elements (greater than x) after x, given an array and an element x as the pivot. The entire process should take a linear amount of time.

Algorithm Explanation:

In the Main Code (Simple Recursive Calls)
1 - Base Case: if low(l) exceeds high(h)
2 - Partition the array and return the correct index of the pivot element
3 - Repeat for the left half
4 - Repeat for the right half

Partition Code
1 - For a given array in the range A[l…….h].
2 - Take the pivot element as A[l]
3 - Count all element smaller than pivot
4 - Place/Swap pivot element with A[l + count]
5 - Start checking all elements with i=l & j=h till i<pivot & j>pivot. If A[i]>pivot and A[j]<pivot then swap them.

Algorithm in form of pseudocode:

```
partition(arr[], l, h){
  pivot←arr[l];
   count←0;

 for  i←l+1 to h:
    if(arr[i]<=pivot):
      count++;

pivot_index ←l + count;
swap arr[s] and arr[pivot_index]

i←l , j←h;
while (i<pivot_index and j>pivot_index):
   if(arr[i]>arr[pivot_index] and arr[j]<=arr[pivot_index]):
      swap arr[i++] and arr[j - -]

   else if(arr[i] < arr[pivot_index]):
     i++;

   else:
    j–;

return pivot_index
}


quicksort(arr[], l, h){
if(l<h):
  partIndex = partition(arr[], l, h);
  quicksort(arr, l, partIndex - 1);
  quicksort(arr, partIndex+1, h);
}
```