CP312 Assignment #3

Memet R.
130951550

1. Statement is True.
For some arrays where the values are some what sorted insersion will be quicker.
The best case scenario for insersion sort is $\Omega(n)$ and for Quick sort is $\Omega(n \log n)$.
for most cases how ever Quick sort is more effiecent. $O(n^2)$ for ~~insert~~ insersion and $O(n \log n)$ for Quick sort.

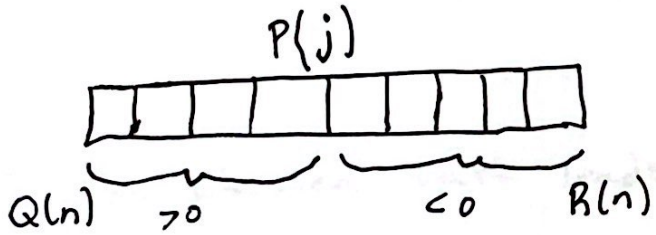
Example List [1,2,3,4] will run in Linear time with insersion sort but still need $n \log n$ time in Quick sort

```
i, j = 0

while j < n  //Where n is size of array

    if array[j] < 0
        if i != j

            i++
            swap(array[i], array[j])

        end
    end

    j++

end
```

2. Since the Algorithm does not require sorting, only that negative numbers come before positive the time complexity is $O(n)$. This is because the program only needs to Traverse once as in the Quick Sort partitioning with Zero being the piviot.

Proof

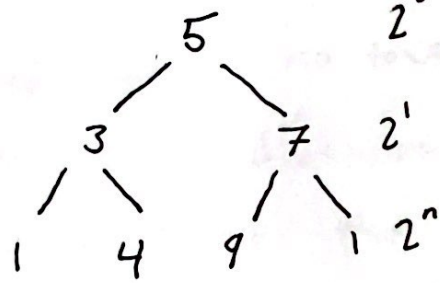
The piviot of array n $P(K) = 0$. There exists a point where all values $P(0)$ to $P(j)$ are negative.



These two subsets can vary in size and as we traverse the array all negative values are swapped into the Q(n) array and positive into R(n). Once we Traverse the entire array we combine them again to get the original P(n) array sorted.

3. Proof: If we view the problem as a decision tree were for every element of the array we compare is element A[i] = X. In the best case scenario we choose a path in the tree that contains the value X. The tree does not need to be sorted.

example



2^0 X = 4

2^1 Path = {5, 3, 4}

2^n height = log n

If the choosen path leads to A[i] = X, i.e. the best case, the path has the same height of a Binary tree.

~~complexity~~ complexity as

The height of a binary tree being Log(n). thus, the best case is $\Omega(\log n)$ for an array search.

```
r = range of tower
n = number of houses
h = array of houses

quickSort(h)

numTowers, i = 0

while i < max(h)
        for j, i to (i + 2r)
            if h.contains(j)
                numTowers++
            end
        end
    i += 2r
end
```

4. Proof of ~~Correctness~~ optimality

We can prove the solution by taking all the distances between houses that are less than the range of the tower. For example if a house is 2 units from the western most point and the next is 7 units with a cell tower of range 2, this difference can be ignored since no tower is needed for the gap. once the distance between houses is summed we get the total number of towers and multiple by range. if this number is greater than total distance we know the salution is valid. If we subtract range from the total it will become less than total distance. thus showing any less towers would result in not enough and being optimal.

Correctness

Since all elements are with in range of a tower by the ~~code~~ contains search they are all covered