Algorithms Lab

Assignment 3: Randomized Quick Sort

Aditya Badayalya

510819056

IT-Hy

Randomized Quick Sort Algorithm:

As opposed to other methods of implementation of the Quick Sort algorithm which involve selecting a fixed position as pivotal value, Randomized Quick Sort can prove to be more efficient in some cases making the average case time complexity equal to O(N*log(N)) but the worst case time complexity still remains as $O(N^2)$.

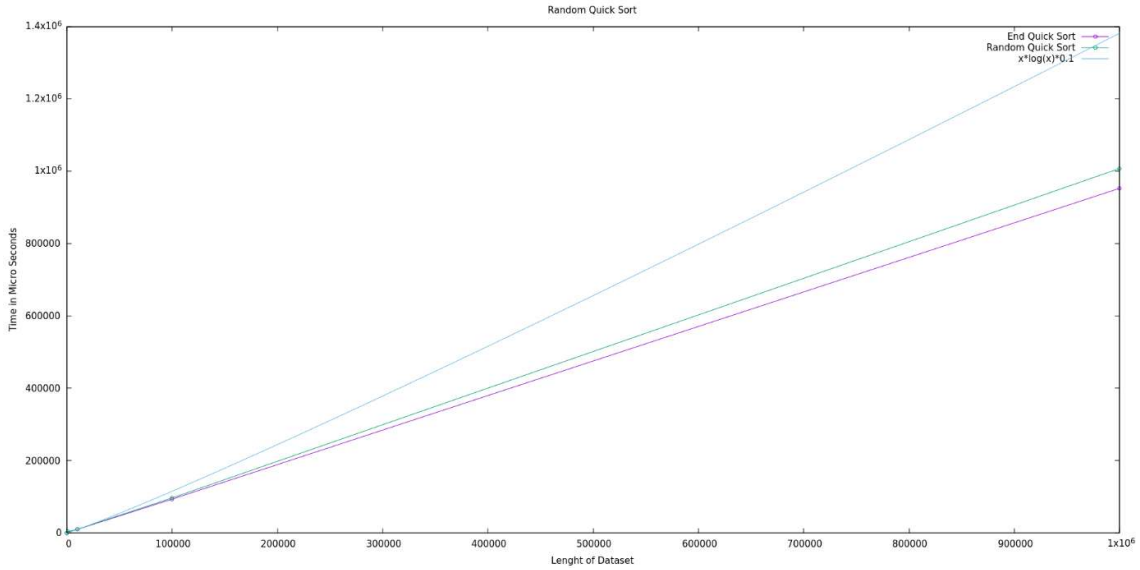
```
import random
class Solution:
    def partition(self, arr, l, r):
        temp = random.randint(l,r)
        pivot=arr[temp]
        arr[r],arr[temp]=arr[temp],arr[r]
        i =l-1
        for j in range(l,r):
            if(arr[j]<pivot):
                i+=1
                arr[j],arr[i]=arr[i],arr[j]
        i+=1
        arr[i],arr[r]=arr[r],arr[i]
        return i
    def quickSort(self, arr, l,r):
        if(l<r):
            pi = self.partition(arr,l,r)
            self.quickSort(arr,l,pi-1)
            self.quickSort(arr,pi+1,r)


if __name__ == '__main__':
    arr = [random.randint(1,500) for _ in range(100)]
    Solution().quickSort(arr,0,99)
    print(arr)
```

In the above algorithm, the pivot value picked is a random value making the chances of it being the highest or the lowest value in the array very low thus avoiding the worst-case complexity of $O(N^2)$

The above algorithm would give a recurrence relation as T(N) = T(N-K-1) + T(K)

The above relation can be solved to obtain the result of the Time complexity as O(N*log(N)) where N is the length of the dataset

Below are graphs depicting the behavior of Random Pivot v/s Fixed Pivot



Below is a comparison graph where the length of dataset is over the range 1000-100000