Lab 5

Student ID	Name
614760	Omkar Nath Chaudhary
614732	Sushil Subedi
614604	Rahul Niraula
614600	Shrawan Adhikari

Q1. Use QuickSort to sort the array. Use leftmost value as pivot each time.

[**1**, 6, 2, 4, 3, 5]

1 **6**, 2, 4, 3, 5

[ ] **2**, 4, 3, 5, 6

[ ] **4**, 3, 5

3 5

Sorted: 1, 2, 3, 4, 5, 6

Q2. Use QuickSort analysis to answer the following questions.

a) **Good pivots ?**

Sorting the given array A = [1, 1, **2, 3, 3, 4, 5**, 6, 7]

We only take the items that falls in ¾ = 75% part of the array. 9 of 75% 9*0.75 = 6.75 = ~7

Min = arrLength - 7 = 9 - 7 = 2
Max = 7

Min < All Items <= max
I.e All Items = [2, 3, 3, 4, 5]

b) **Is it true that at least half the elements are good pivots ?**

Using 75%, part of the array , we got [2, 3, 3, 4, 5] . i.e 5 elements
Therefore, out of 9 elements 5 are good pivots, 5/9 = 0.555.
Hence, more than 50% of the elements of A are good pivots.

Q3. Determine algorith to find element "m" in sorted array, and prove that algorithm runs in o(n) time.

To find the element m, in the sorted array. We can use binary search, to find whether value m is in Array A or not.
This binary search runs in $O(log\ n)$

Proof: Yes $O(log\ n)\ is\ little\ o(n)$

$$\lim_{n \to \infty} \frac{\log n}{n}$$

$$\frac{\frac{1}{n}}{1}$$

$$\frac{1}{n}$$

$$\frac{1}{\infty}$$

$$0$$

As f(n)/g(n) = 0, $logn\ is\ a\ little\ o(n)$. True

Q4. Devise a pivot-selection strategy that guarantee QuickSort running time of O(nlogn)

QuickSort sorts the array in $O(n^2)$ in worst case. But with good pivot selection, it can sort in runtime $O(n\ log\ n)$.

Assume, we have array = [7, 4, 9, 3, 2, 6, 5, 1, 8] . Of length 9.
Here, $n^{th}\ mid\ item\ =\ (0\ +\ 8)\ /\ 2\ =\ 4th\ item$

Using Quick-Select we can find the mid-item and take it as pivot.
I.e here we need to find 4th item, using quick select (**this runs in O(n)**) . We get **pivot 5.**
I.e [1, 4, 3, 2, **5**, 7, 9, 6, 8] .

With this, good pivot selection. It divided the array in exact half. Similarly for these halves,
[1, **2**, 4, 3] [6, **7**, 9, 8]
Now calculating mid-item and pivot again.
First half: **pivot 2**. Second half: **pivot 7.** O(n)
Similarly …….
…….

Therefore, each time array is half, and running time for each step recursion tree is $O(logn)$
And pivot selection for each steps is: constant * n . i.e $O(cn)$. This is $O(n)$
Hence, even for the worst-case QuickSort selection, the running time is $O(n\ log\ n)$

Q5. Perform QuickSelect to find the median of the given array.

[1, 12, 8, 7, -2, -3, 6]

n = 7
To find the median, we first need to find the mid position = (n + 1) / 2 = 8/2 = 4th

Using Quick-Select to find the **4th** item
K = 4th, [1, 12, 8, 7, -2, -3, 6]