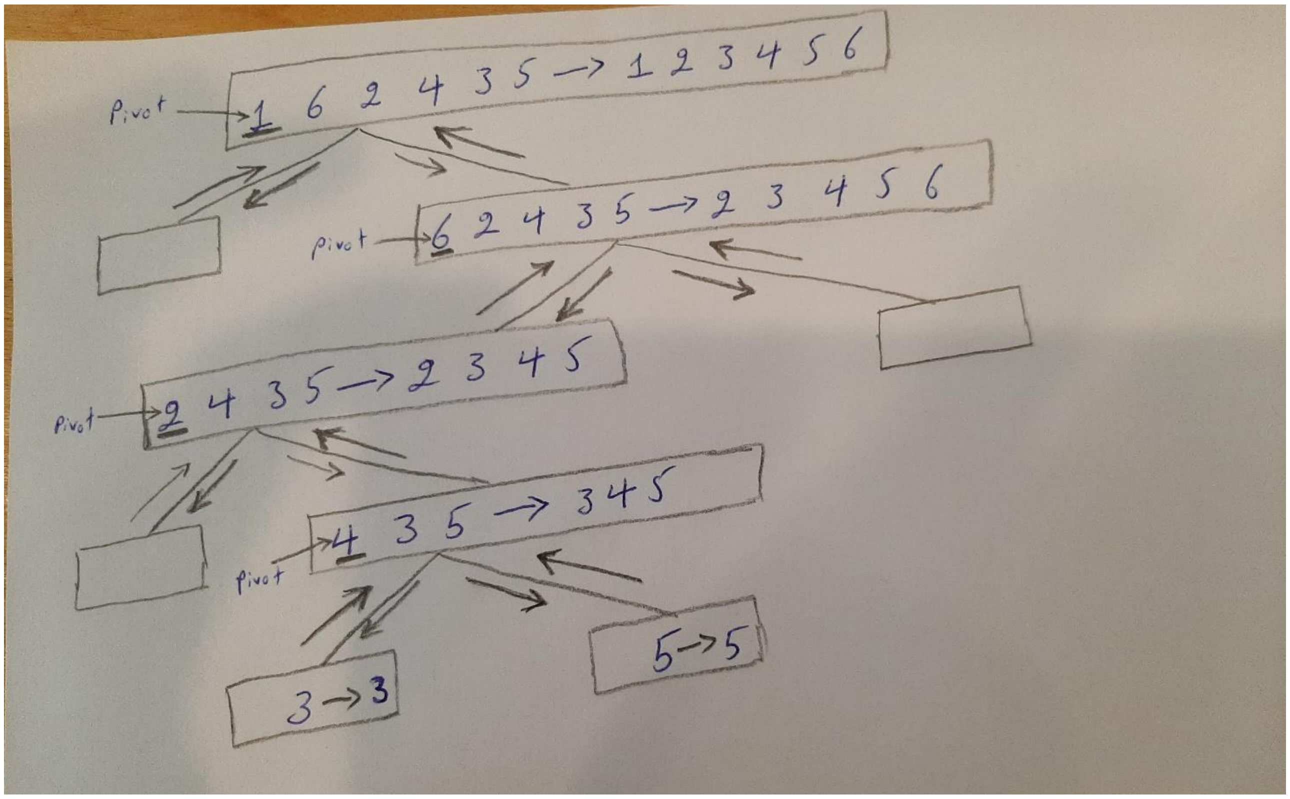
Lab05

Question1:

Show all steps of QuickSort in sorting the array [1, 6, 2, 4, 3, 5]. Use leftmost values as pivots at each step.



Question2:

In our average case analysis of QuickSort, we defined a *good self-call* to be one in which the pivot *x* is chosen so that number of elements < x is less than 3n/4, and also the number of elements > x is less than 3n/4. We call an x with these properties a *good pivot.* When n is a power of 2, it is not hard to see that at least half of the elements in an n-element array could be used as a good pivot (exactly half if there are no duplicates). For this exercise, you will verify this property for the array A = [5, 1, 4, 3, 6, 2, 7, 1, 3] (here, n = 9). Note: For this analysis, use the version of QuickSort in which partitioning produces 3 subsequences *L, E, R* of the input sequence *S.*

a. Which x in A are good pivots? In other words, which values x in A satisfy:

i. the number of elements < x is less than 3n/4, and also

ii. the number of elements > x is less than 3n/4

Answer:

A = [5, 1, 4, 3, 6, 2, 7, 1, 3], n=9, **3n/4 = 6.75**

Good pivots:

1- 5 will divide the array into L,G sub-arrays with sizes 6, 2. [1,4,3,2,1,3], [6,7]
2- 4 will divide the array into L,G sub-arrays with sizes 5, 3. [1,3,2,1,3], [5,6,7]
3- 3 will divide the array into L,E,G sub-arrays with sizes 3, 2, 4. [1 ,2,1], [3,3], [5,4,6,7]
4- 2 will divide the array into L,G sub-arrays with sizes 2, 6. [1,1], [5,4,3,6,7,3]
5- 3 will divide the array into L,E,G sub-arrays with sizes 3, 2, 4. [1 ,2,1], [3,3], [5,4,6,7]

b. Is it true that at least half the elements of A are good pivots?

Answer:

We found that 5 elements out of 9 (55%) are considered as good pivots.

<u>Question3:</u>

Give an o(n) (“little-oh”) algorithm for determining whether a sorted array A of distinct integers contains an element m for which A[m] = m. You must also provide a proof that your algorithm runs in o(n) time.

<u>Answer:</u>

For this problem, we can use a binary search algorithm(recursive) because array is already sorted. The base will examine if Array[mid] == mid and if so, returns Array[mid]. And induction case will be if Array[mid] > mid, use binary search in left side otherwise search the right side. If lower > upper(if no such element found), return -1.

This solution solves the problem is O(logn) times.

Now, suppose f(n) = O(logn) and g(n) = o(n).

To show this algorithm runs in o(n),

Lg   f/g   should be 0.
n-> ∞

Using L’Hopital rule, we have,

$$\underset{n\to\infty}{\mathrm{lt}} \frac{f}{g} \Rightarrow \underset{n\to\infty}{\mathrm{lt}} \frac{\log n}{n} \Rightarrow \underset{n\to\infty}{\mathrm{lt}} \frac{\frac{1}{n}}{1} = \frac{1}{\infty} = 0$$

<u>Question4:</u>

Devise a pivot-selection strategy for QuickSort that will guarantee that your new QuickSort has a worst-case running time of O(nlog n).

<u>Answer:</u>

For this problem, we can use QuickSelect algorithm having worst case running time of O(n) to select pivot elements on each recursive pivot selection. This adds O(k) running time whenever section of the area has length k, so has the same cost as the partition step.

Using this algorithm, guarantees that all pivot elements are good pivots, so the recursion tree has height O(log n) and running time is O(nlog n) in the worst case.

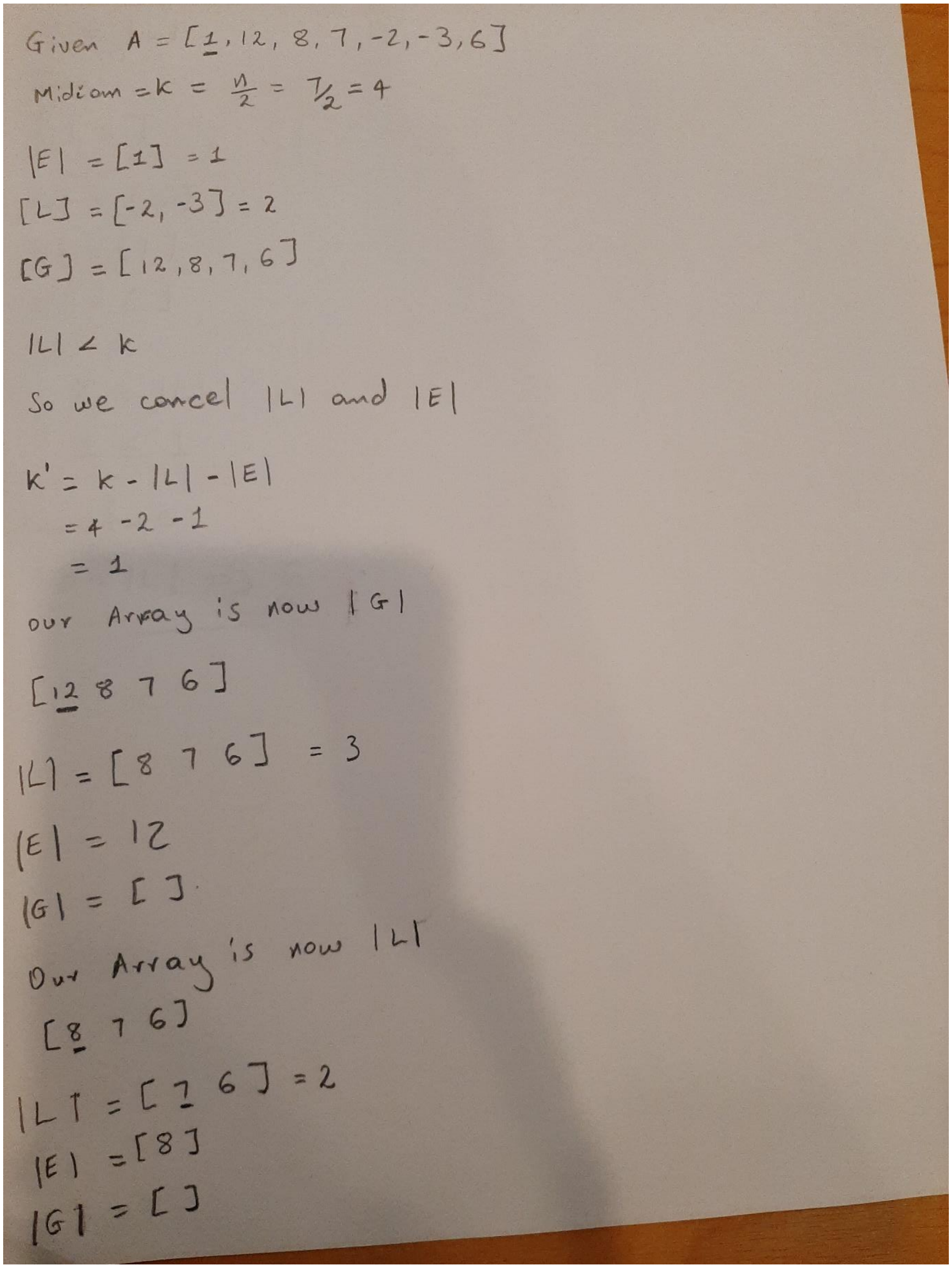
```
Algorithm check(S, lower)
    Input sorted sequence S with n integers and number lower
    Output m if A[m]=m otherwise null
    mid←n/2
    if(n≤0) then
        return null
    if(S[mid]=mid+lower) then
        return S[mid]
    else if(S[mid]<mid+lower) then
        S1← S.copyRange(mid, n-1)
        return check(S1, mid+lower)
    else
        S2← S.copyRange(0, mid-1)
        return check(S2, lower)
```

Proof: In the worst case when m = 0 where A[m] = m, the number of recursive calls are equal to the number of terms in sequence S: n/2, n/4, n/8, ……………, n/2m (= 1) [where m = logn]. Hence, the running time for this algorithm in worst case is ▯(m) or ▯(logn). And we know that logn is o(n).

<u>Question5:</u>

Show the steps performed by QuickSelect as it attempts to find the median of the array [1, 12, 8, 7, -2, -3, 6]. (The median is the element that is less than or equal to n/2 of the elements in the array. Since n is odd in this case, it is the element whose position lies exactly in the middle. Hint: The median is 6.) For pivots, always use the leftmost element of the current array.

<u>Answer:</u>



Given $A = [\underline{1}, 12, 8, 7, -2, -3, 6]$

Midiom $= k = \frac{n}{2} = \frac{7}{2} = 4$

$|E| = [1] = 1$

$[L] = [-2, -3] = 2$

$[G] = [12, 8, 7, 6]$

$|L| < k$

So we cancel $|L|$ and $|E|$

$k' = k - |L| - |E|$

$= 4 - 2 - 1$

$= 1$

our Array is now $|G|$

$[\underline{12}\ 8\ 7\ 6]$

$|L| = [8\ 7\ 6] = 3$

$|E| = 12$

$|G| = [\ ]$.

Our Array is now $|L|$

$[\underline{8}\ 7\ 6]$

$|L| = [\underline{7}\ 6] = 2$

$|E| = [8]$

$|G| = [\ ]$

$k \le |L|$

Then

$[1 \; 6]$

$|L| = [6]$

$|E| = [7]$

$|G| = [\;]$

$\therefore \; k \le |L|$

$1 \le 1$

Therefore

median $= |L| \Rightarrow 6$