1. Sorted Array of distinct integers $A[1,\dots,n]$. Find whether there is an i s.t. $A[i]=i$. Divide + Conquer that runs in $O(\log n)$. Describe + justify.

Algorithm:

1. Entire sorted Array
2. At each step find the middle element
3. Compare the middle elements value with its index
   - If $A[mid] == mid$, return mid
   - If $A[mid] > mid$, search the left subArray
   - If $A[mid] < mid$, search the right subArray
4. Repeat steps 2+3 until array size is 0 or $A[mid] == mid$

Correctness:

- At each step the alg discards half of the array from consideration
- If $A[mid] == mid$, we have an index where $A[i]=i$
- If $A[mid] > mid$, we move to the left subarray since all indices to the right are larger.
- If $A[mid] < mid$, we move to the right subarray for similar reasons

Running Time:

- At each step we divide the array size by 2 $O(1)$
- After dividing, we recursively search in one of the sub arrays. the subarray sized is halved to the previous step each recursion. $O(T(n/2))$

$O(T(n)) = O(1) + O(T(n/2)) \Rightarrow O(\log n)$

2. Sort the array $A = \{\overset{i}{7^a}, 3^a, 5, 11^a, 1, 9, 2, 6, 11^b, \overset{11^a}{7^b}, 12, 3^b, 4, \underset{\text{pivot}}{\underline{8}}\}$ using quicksort.

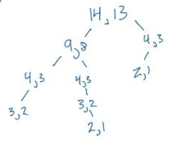
a) After Partition is called Once

$7^a\ 3^a\ 5\ 1\ 9\ 2\ 6\ 7^b\ 3^b\ 4\ 8\ 11^b\ 11^a\ 12$

b)

After the first partition, the relative positions of $11^a + 11^b$ swapped.

c)



d) 9

e) 5

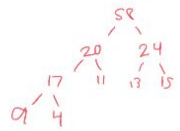
f) $14+9+4+4+4+2+3+3+2 = \underline{45}$

3.) Sort $A = <\overset{1}{9}, \overset{2}{4}, \overset{3}{13}, \overset{4}{20}, \overset{5}{11}, \overset{6}{24}, \overset{7}{15}, \overset{8}{17}, \overset{9}{58}>$ using Heapsort.

a) Array after Build-Max-Heap is returned

$<58, 20, 24, 17, 11, 13, 15, 9, 4>$

or



b) $O(9)$

c) $<4, 9, 11, 13, 15, 17, 20, 24, 58>$

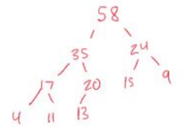
d)

Max-Heap $<22, 13\underline{a}, 13\underline{b}, 10, 9, 8, 7>$

$\downarrow$ heapsort

Final $<7, 8, 9, 10, 13\underline{b}, 13\underline{a}, 22>$

e) Insert 35



4.

1. Keep 3 pointers, RedEnd, BlueStart, BlueEnd.
2. Iterate through array from left to right, At each step using color(i).
3. If ball is red, swap it with RedEnd[swap(i, RedEnd)], then increment RedEnd.
4. If ball is blue, just increment the iterator, as blue should already be in place.
5. If ball is green, swap it with the ball at BlueEnd [swap(i, blueEnd)], then decrement BlueEnd.
6. Repeat process until iterator reaches BlueEnd.

Correctness:

- All red balls preceed blue balls due to whenever a red ball is encountered it is swapped with redEnd
- All blue balls preceed green balls due to whenever a green ball is encountered it is swapped with blueEnd.
- The algorithm ends as soon as it is finished iterating

Time Complexity: $O(n)$

- The algorithm iterates through the array only once
- The time is proportional to the number of elements in the array "n"
- color(i) + swap(i, j) take the same amount of time regardless of the input size