In this exam, you are asked to complete the function definitions to sort the given array **arr** with **ascending** order.

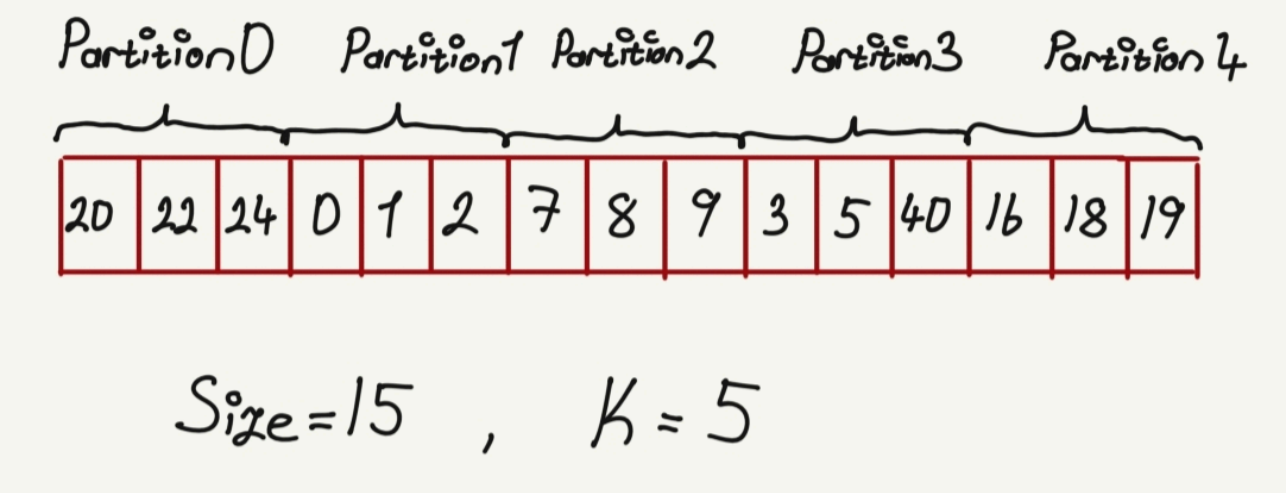
- **kWayMergeSortWithHeap()** should count the number of **comparison** and **swap** executed during sorting process (Comparisons are only between the values to be sorted during insertion sort and heapify process) and returns the total number of calls of **kWayMergeSortWithHeap()**.

K Way Merge Sort With Heap algortihm (**kWayMergeSortWithHeap()**) is as follows:

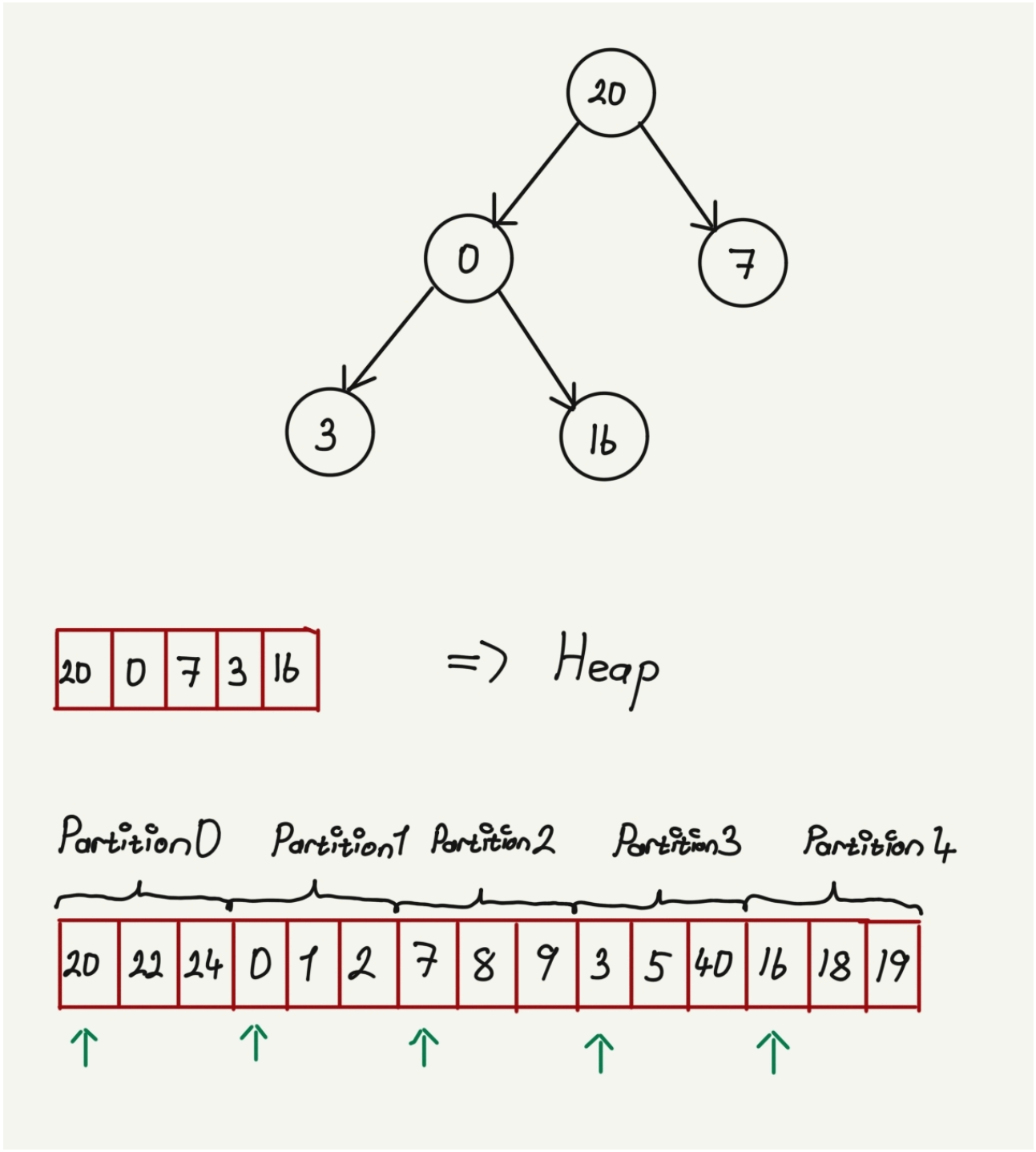
- If the size of the array is less than K, then sort the array by using insertion sort.(You can use the insertion sort algorithm given to you in THE0.)
- Otherwise, split the array into K sub-arrays and do K recursive calls to sort the partitions.
    - Then, merge K sorted arrays.
    - When merging K sorted-arrays, you should use a Binary Min Heap to select the minimum element between the minimum elements of K partition arrays.
    - When creating the array of the heap,
        - Firstly, generate a linear array whose elements are the minimum elements of the K partition arrays. At the beginning, the position of the each element is determined by the belonging partition. For example, the element coming from partition 0 is placed to heap_array[0] and the element coming from partition 1 is placed to heap_array[1] etc.
        - Then, heapify the initial array.
    - After finding the minimum element, you should insert a new element from the related partition to the Min Heap.
        - Read the minimum element in the heap and record it.
        - Then, replace the minimum element with a new element from the partition that has the last minimum element. (New element insertion is not a swap operation. Swap has to be counted only inside the heap or insertion sort.)
        - Then, heapify the current array.

- In case of equality during heapify and insertion sort, do not swap the elements.

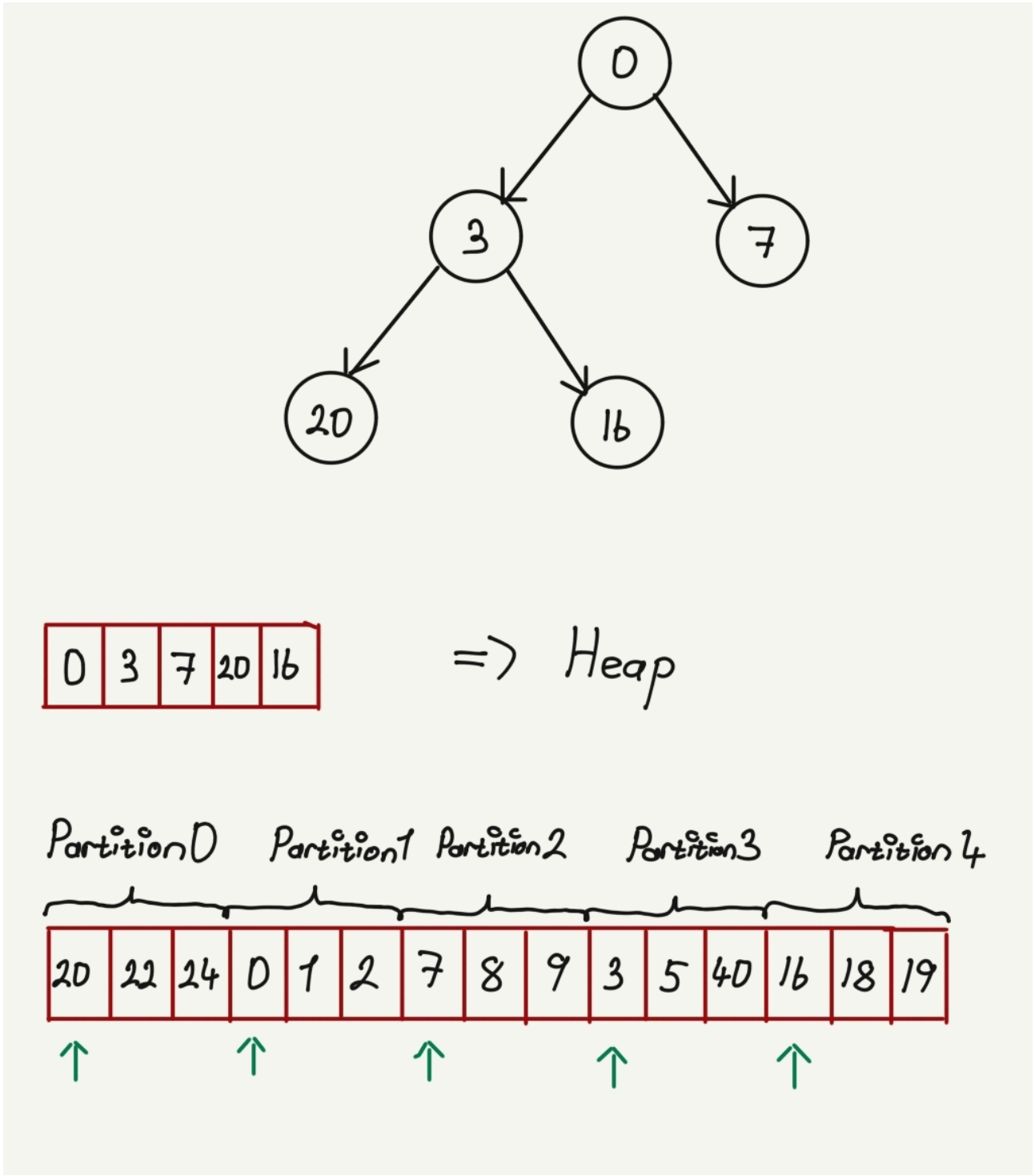
- Count the comparison and swap between any 2 elements of the array H in both insertion sort and heapify, such as H[i]>H[j]
- Return the total number of **kWayMergeSortWithHeap()** calls.
- Let's have an example case:
  - Let's say in some point the array is as follows:
-



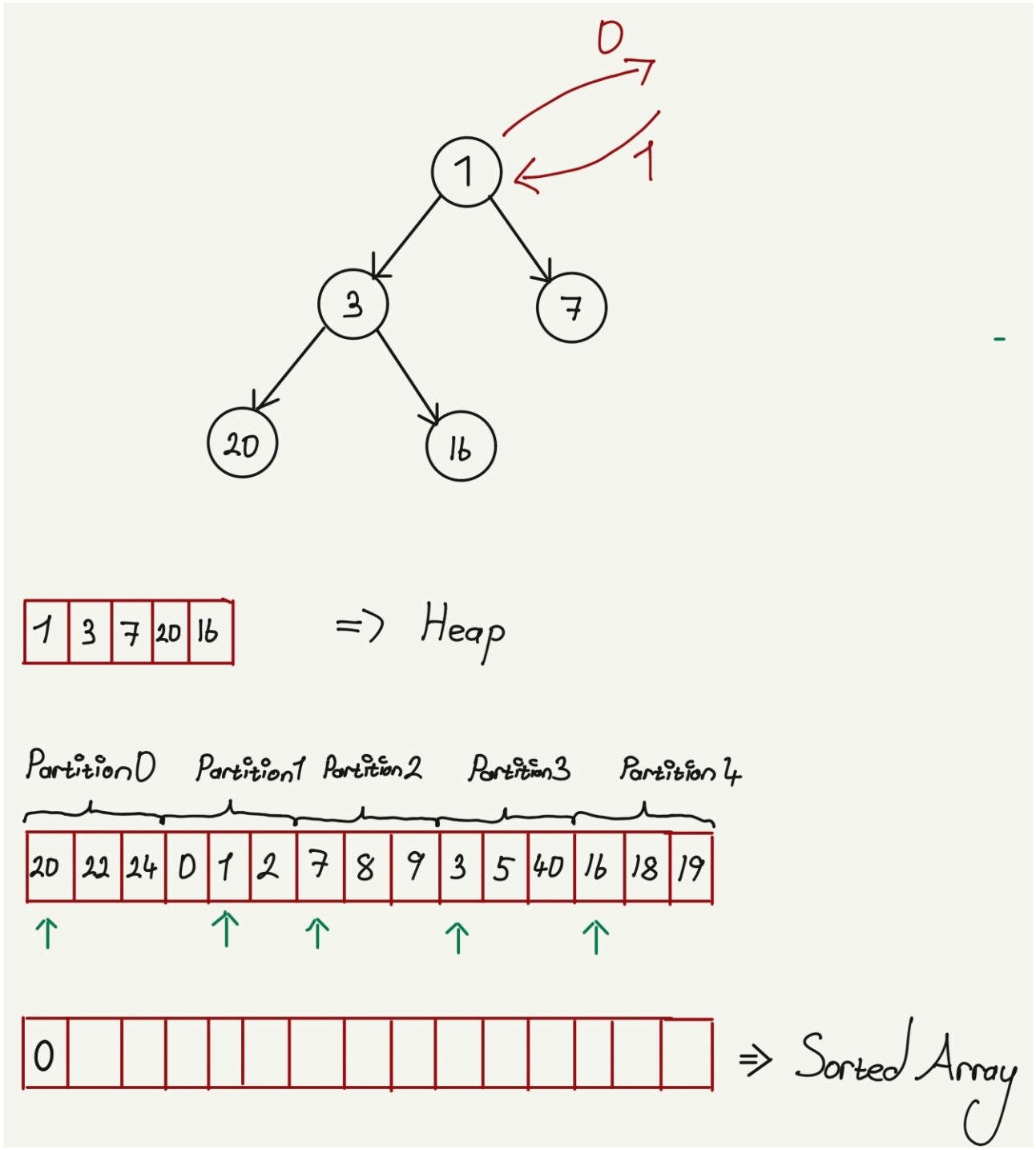
-
- Create a heap array and place the first elements of the partitions



- Heapify the array(6 comparisons and 2 swaps are required.)



- Record the minimum and insert a new element(It is not counted as swap.)



- Then, heapify again.