we build a binary heap in-place bottom-up within an array. A variant of a binary heap is a quaternary heap which is a heap where nodes can have (at most) 4 children.

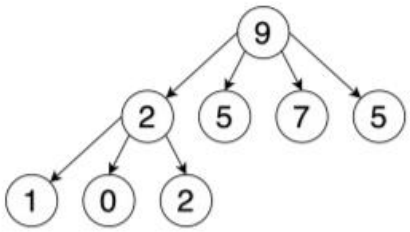
In this question, you will implement in-place quaternary heapsort. This should build a quaternary heap in-place bottom-up, then use this to sort the array in ascending order.

(a) First, you should implement the helper function quaternaryDownheap in QuaternaryHeapsort.java. This should perform a downheap operation on a quaternary max heap within an array.

(b) Using this helper function, implement quaternaryHeapsort within QuaternaryHeapsort.java.

(c) State the worst case time and space complexity (in Big-O notation) of all your methods in their Javadoc.

Below is an example visualisation of a quaternary max heap. Note that it is a complete quaternary tree (leaf nodes of bottommost level are as far left as possible). The array representation of the below heap would be [9, 2, 5, 7, 5, 1, 0, 2].



Here are some examples of quaternaryDownheap. It works on the array representation of a heap and performs the downheap in-place.

• Input heap [0, 10, 20, 30, 40] size 5 and start 0 would result in [40, 10, 20, 30, 0].

• Input heap [1, 0, 2, 3, 4, 10, 20, 30, 40], size 9 and start index 1 would result in [1, 40, 2, 3, 4, 10, 20, 30, 0]. Note that higher levels (the root node in this case) are not changed, regardless of whether they satisfy the heap property.

• Input heap [10, 20, 1, 2, 3, 11, 12, 13, 14], size 9 and start index 0 would continue the downheap through the entire heap, resulting in [20, 14, 1, 2, 3, 11, 12, 13, 10].

• Input heap [10, 20, 1, 2, 3, 11, 12, 13, 14], size 5 and start index 0 would only consider the first 5 items in the array, resulting in [20, 10, 1, 2, 3, 11, 12, 13, 14]. Values at index ≥ size are not modified.
Notes:

• You should create and use private helper methods to improve readability of your code.

• You will be given no marks for any variant of heapsort that doesn't utilise a quaternary heap.