HEAP SORT

HEAP SORT

USES A HEAP TO HELP SORT ELEMENTS IN ASCENDING OR DESCENDING ORDER

FIRST CONVERTS THE UNSORTED LIST OR ARRAY INTO A HEAP - THIS CAN BE DONE IN PLACE

USE THE HEAP TO ACCESS THE MAXIMUM ELEMENT AND PUT IT IN THE RIGHT POSITION IN THE ARRAY

HEAP SORT

HEAPIFY

FIRST CONVERTS THE UNSORTED LIST OR ARRAY INTO A HEAP - THIS CAN BE DONE IN PLACE

TAKE A PORTION OF THE ARRAY - MAKE ALL ELEMENTS IN THAT PORTION SATISFY THE HEAP PROPERTY

KEEP ADDING ADDITIONAL ELEMENTS INTO THE HEAP PORTION ENSURING THAT THESE ADDITIONAL ELEMENTS ALSO SATISFY THE HEAP PROPERTY

THE HEAP WILL GROW TO ENCOMPASS ALL ELEMENTS IN THE ARRAY

SORT

USE THE HEAP TO ACCESS THE MAXIMUM ELEMENT AND PUT IT IN THE RIGHT POSITION IN THE ARRAY

A HEAP OFFERS O(1) ACCESS TO THE LARGEST OR THE SMALLEST ELEMENT

REMOVE THE LARGEST ELEMENT FROM THE HEAP AND POSITION IT AT THE END OF THE SORTED ARRAY

THE SORTED ARRAY WILL GROW TO ENCOMPASS ALL ELEMENTS IN THE ARRAY

HEAP SORT

HEAPIFY

WE'LL USE A **MAXIMUM HEAP**
SO WE CAN ALWAYS ACCESS THE
LARGEST ELEMENT IN O(1) TIME

A HEAP CAN BE REPRESENTED
USING AN ARRAY

HEAPIFY IS THE OPERATION TO CONVERT THE
UNSORTED ARRAY TO A HEAP

WE USE THE SAME ARRAY WITH NO
ADDITIONAL SPACE TO DO THE HEAPIFY

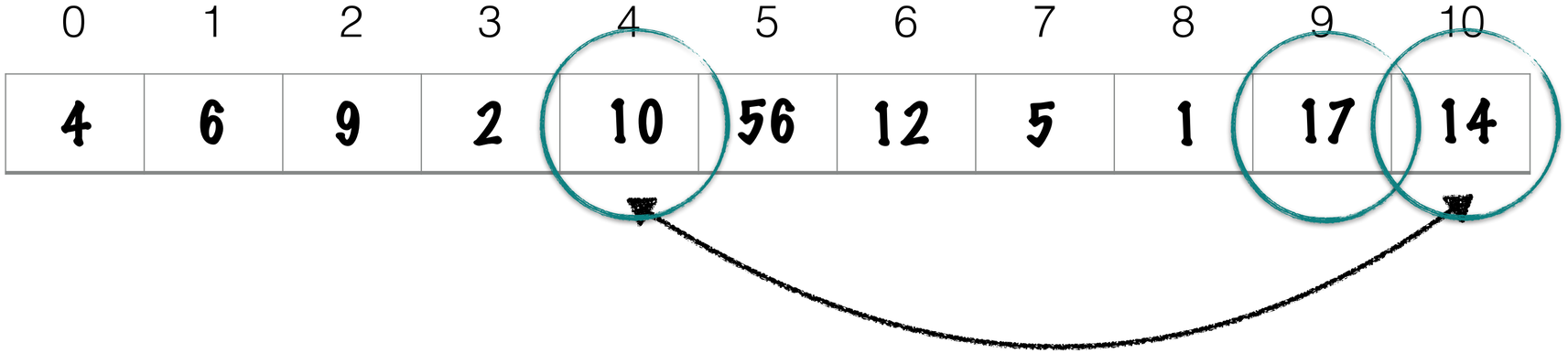
HEAP SORT

HEAPIFY

HEAP SORT

HEAPIFY

START WITH THE PARENT OF THE LAST ELEMENT IN THIS UNSORTED ARRAY



ENSURE THAT ELEMENT AT INDEX 4 AND ALL CHILDREN BELOW IT SATISFY THE HEAP PROPERTY

NODE AT INDEX: `i`

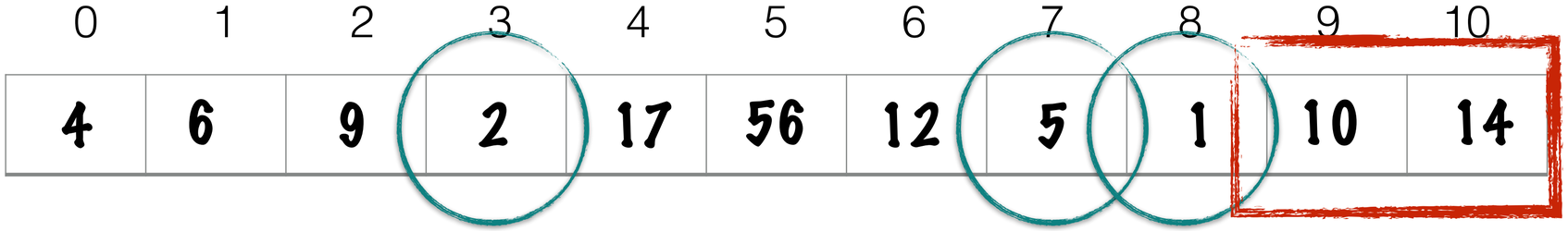
HAS PARENT AT INDEX: `(i - 1)/2`

HEAP SORT

HEAPIFY

MOVE TO THE PREVIOUS PARENT AT INDEX 3

THERE ARE NO FURTHER CHILDREN TO CONSIDER



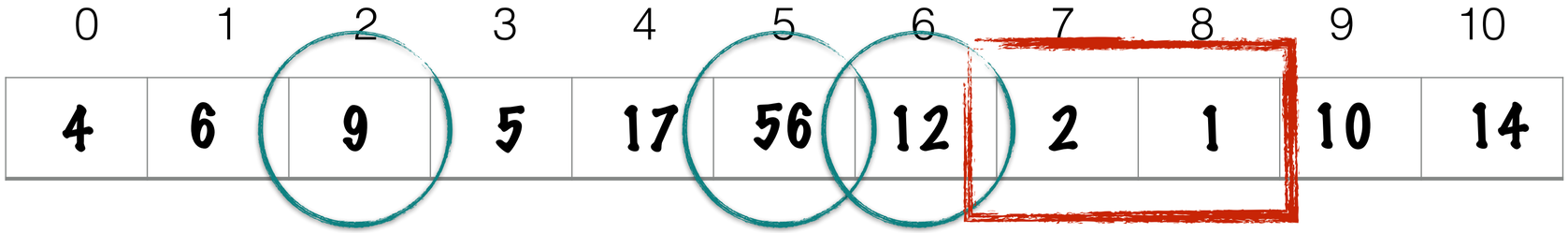
ENSURE THAT ELEMENT AT INDEX 3 AND ALL CHILDREN BELOW IT SATISFY THE HEAP PROPERTY

HEAP SORT

HEAPIFY

MOVE TO THE PREVIOUS PARENT AT INDEX 2

THERE ARE NO FURTHER CHILDREN TO CONSIDER



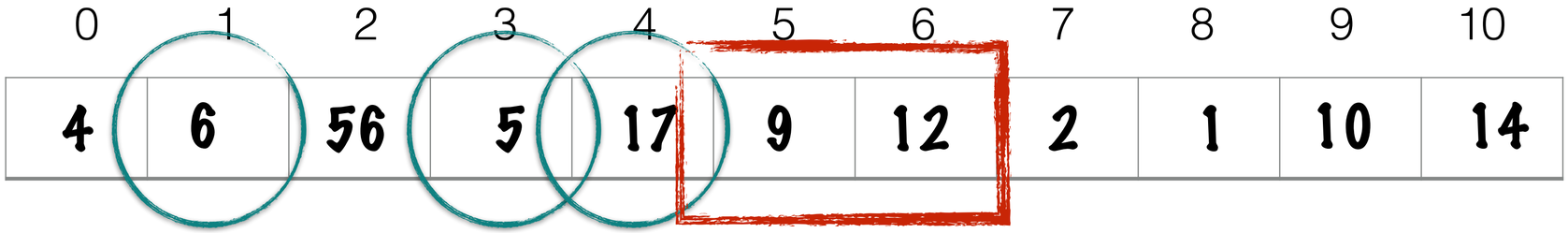
ENSURE THAT ELEMENT AT INDEX 2 AND ALL CHILDREN BELOW IT SATISFY THE HEAP PROPERTY

HEAP SORT

HEAPIFY

MOVE TO THE PREVIOUS PARENT AT INDEX 1

THERE ARE NO FURTHER CHILDREN TO CONSIDER

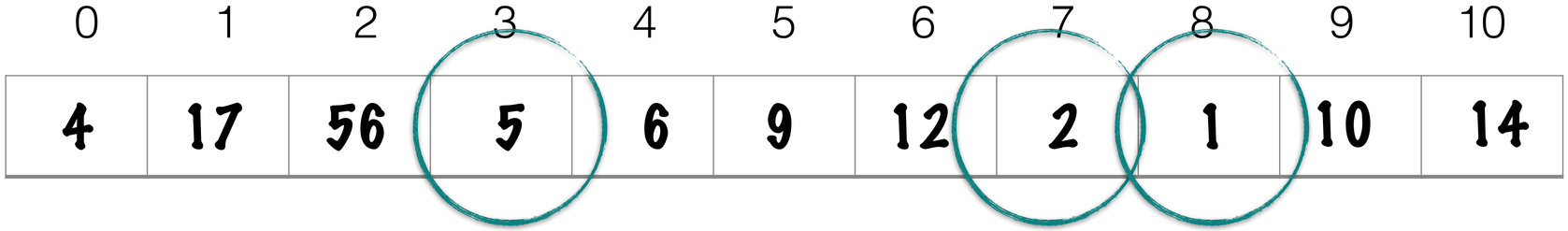


ENSURE THAT ELEMENT AT INDEX 1 AND ALL CHILDREN BELOW IT SATISFY THE HEAP PROPERTY

HEAP SORT

HEAPIFY

CONSIDER THE CHILDREN OF ELEMENTS AT INDEX 3 AND 4 AS WELL, STARTING WITH 3

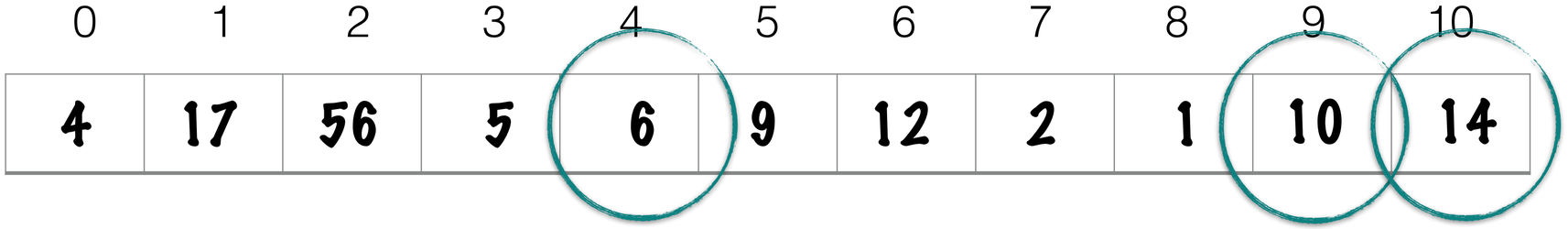


THE NODE REPRESENTED BY INDEX 3 SATISFIES THE HEAP PROPERTY

HEAP SORT

HEAPIFY

NOW FOR THE NODE AT INDEX 4

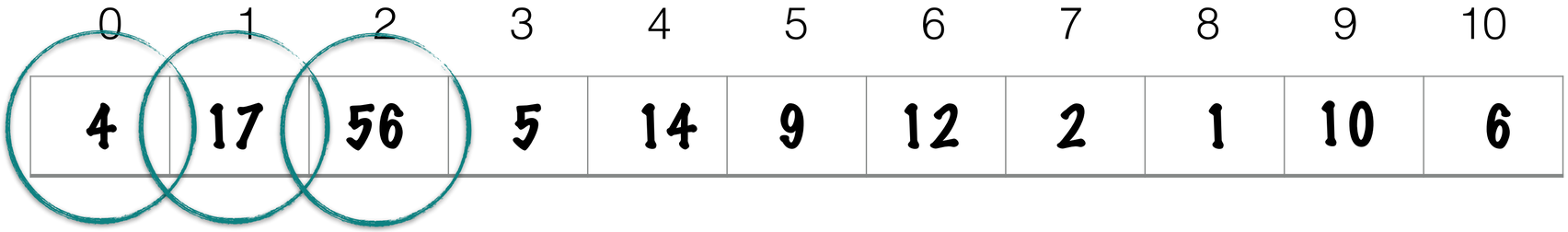


THIS NEEDS TO BE HEAPIFIED FURTHER - THE MAX HEAP PROPERTY IS NOT SATISFIED

HEAP SORT

HEAPIFY

MOVE TO THE PREVIOUS PARENT AT INDEX 0

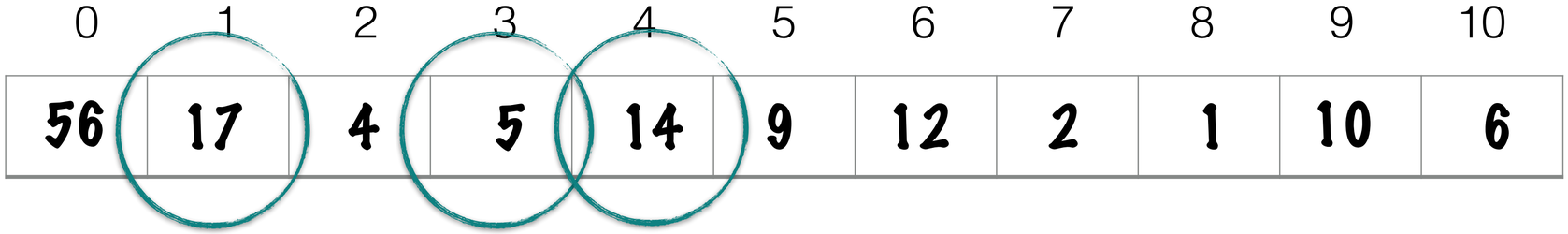


ENSURE THAT ELEMENT AT INDEX 0 AND ALL CHILDREN BELOW IT SATISFY THE HEAP PROPERTY

HEAP SORT

HEAPIFY

CONTINUE CHECKING THE NODES AT INDEX 1 AND 2 STARTING WITH 1

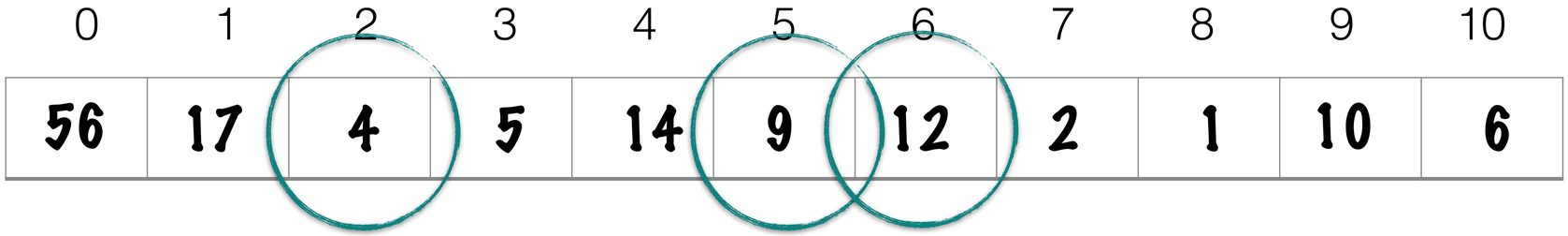


THE ELEMENT AT INDEX 1 SATISFIES THE MAX HEAP PROPERTY

HEAP SORT

HEAPIFY

CHECK INDEX 2



THIS DOES NOT SATISFY
THE MAX HEAP PROPERTY
AND HAS TO BE HEAPIFIED
FURTHER

HEAP SORT

HEAPIFY

0	1	2	3	4	5	6	7	8	9	10
56	17	12	5	14	9	4	2	1	10	6

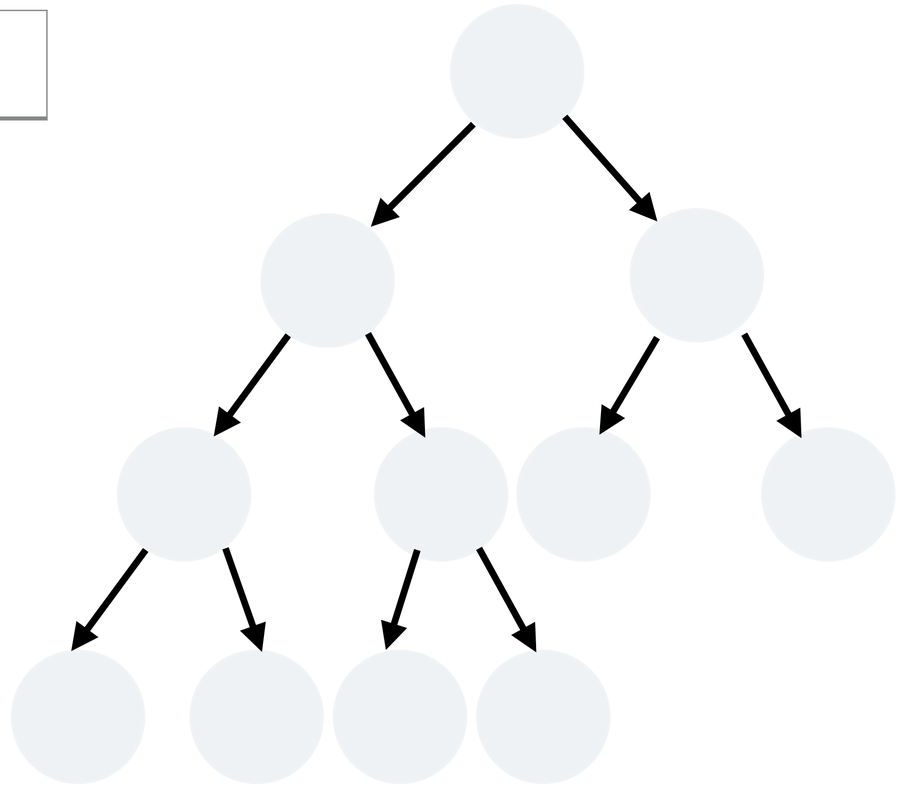
THE HEAP PROPERTY IS SATISFIED

NO FURTHER CHECKS NEED TO BE DONE - THIS IS A MAXIMUM HEAP!

HEAP SORT

HEAPIFY

0	1	2	3	4	5	6	7	8	9	10
56	17	12	5	14	9	4	2	1	10	6



HEAP SORT

HEAPIFY

0	1	2	3	4	5	6	7	8	9	10

A MAXIMUM HEAP USING
IN-PLACE HEAPIFY!