THE BINARY HEAP
IMPLEMENTATION

THE LOGICAL STRUCTURE OF A BINARY HEAP IS A TREE - SO THEORETICALLY WE COULD REPRESENT A HEAP JUST AS WE WOULD A TREE

EACH NODE WOULD HAVE A POINTER TO THE LEFT AND RIGHT CHILD

THE OPERATIONS TYPICALLY PERFORMED ON A HEAP REQUIRES US TO:

1. TRAVERSE DOWNWARDS FROM THE ROOT TOWARDS THE LEAF NODES
2. TRAVERSE UPWARDS FROM THE LEAF NODES TOWARDS THE ROOT

ON A HEAP WE WANT TO BE ABLE TO:

GET LEFT CHILD GET RIGHT CHILD GET PARENT

THE BINARY HEAP IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT

A NODE WOULD NEED 2 CHILD POINTERS AND A PARENT POINTER

THIS IS A LOT OF EXTRA SPACE

HEAPS CAN BE REPRESENTED MUCH MORE EFFICIENTLY BY USING AN ARRAY AND HAVING AN IMPLICIT RELATIONSHIP TO DETERMINE THE PARENT, LEFT AND RIGHT CHILD OF A NODE

EVERY LEVEL OF THE BINARY TREE IN A HEAP IS FILLED EXCEPT PERHAPS THE LAST

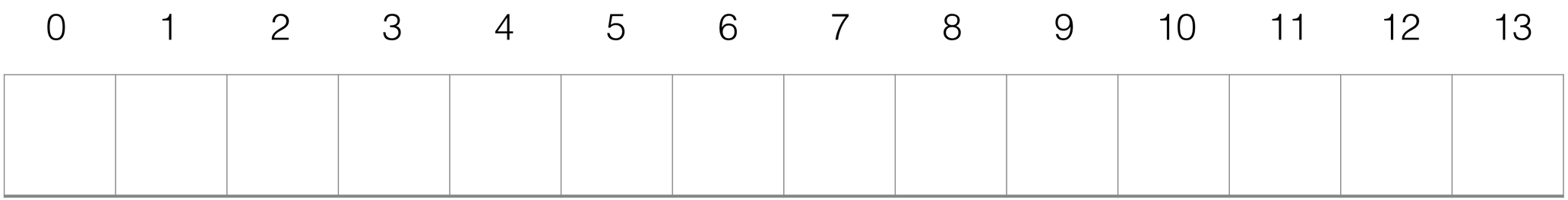
THIS MEANS CONTIGUOUS SLOTS IN AN ARRAY CAN BE USED TO REPRESENT BINARY TREE LEVELS

THE BINARY HEAP
IMPLEMENTATION

GET PARENT

NODE AT INDEX 0

LEFT CHILD AT INDEX 1
RIGHT CHILD AT INDEX 2



NODE AT INDEX: `i`

GET LEFT CHILD HAS A LEFT CHILD AT INDEX: `2i + 1`

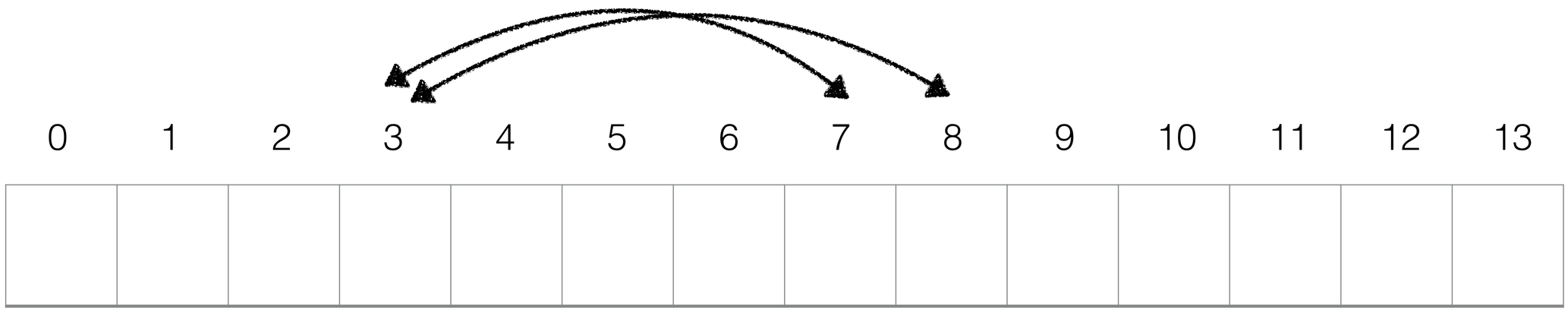
GET RIGHT CHILD HAS A RIGHT CHILD AT INDEX: `2i + 2`

THE BINARY HEAP
IMPLEMENTATION

GET PARENT

NODE AT INDEX 3

LEFT CHILD AT INDEX 7
RIGHT CHILD AT INDEX 8



NODE AT INDEX: `i`

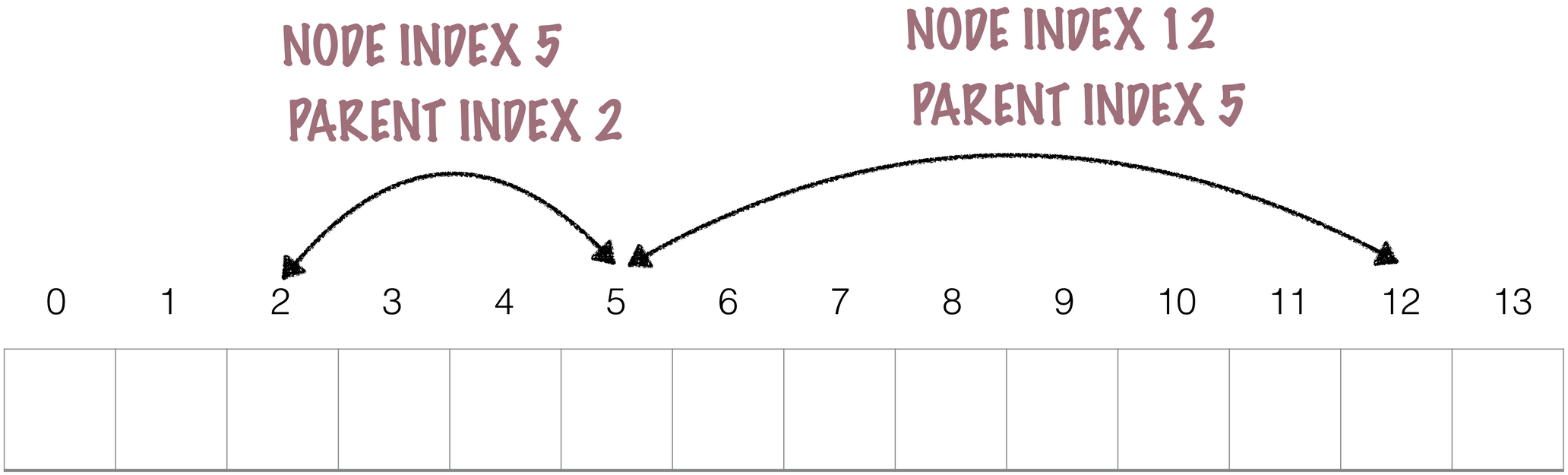
GET LEFT CHILD HAS A LEFT CHILD AT INDEX: `2i + 1`

GET RIGHT CHILD HAS A RIGHT CHILD AT INDEX: `2i + 2`

THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD



NODE AT INDEX: `i`

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

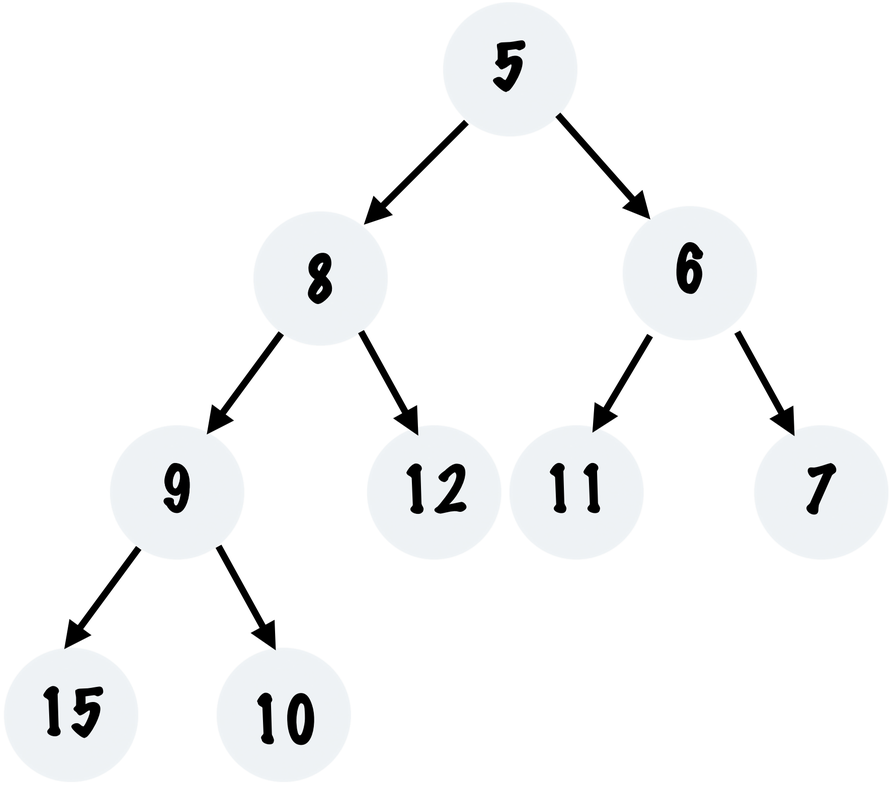
THE BINARY HEAP

IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT

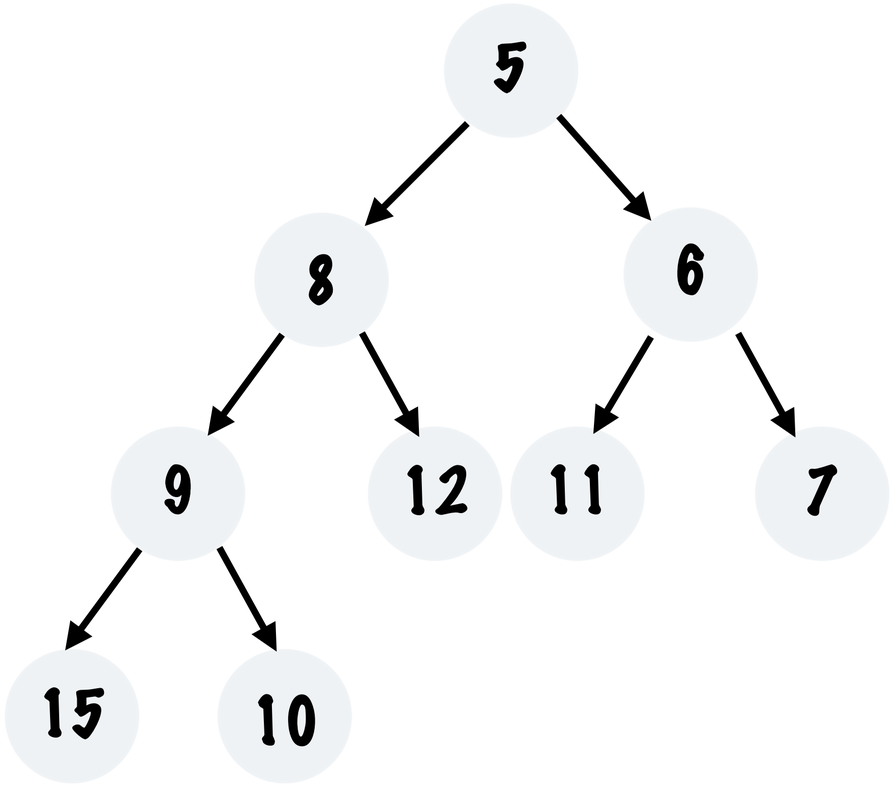


THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT



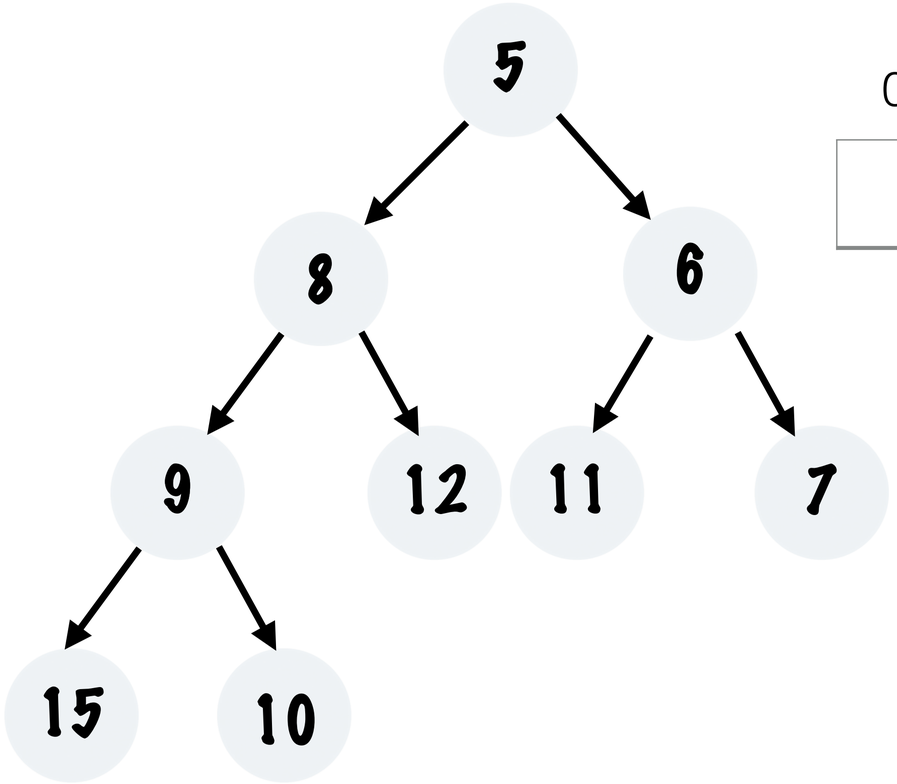
THE BINARY HEAP

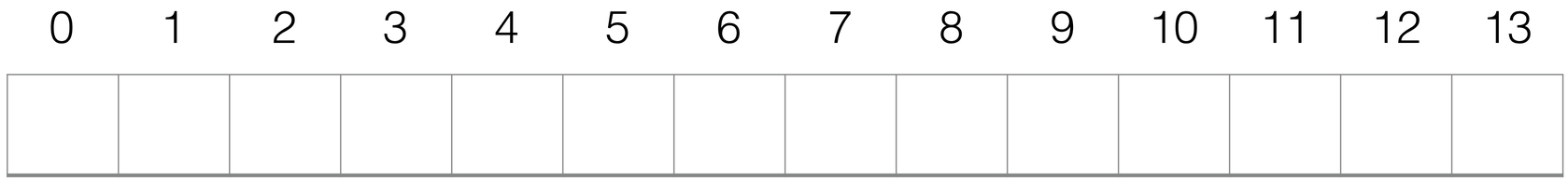
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT



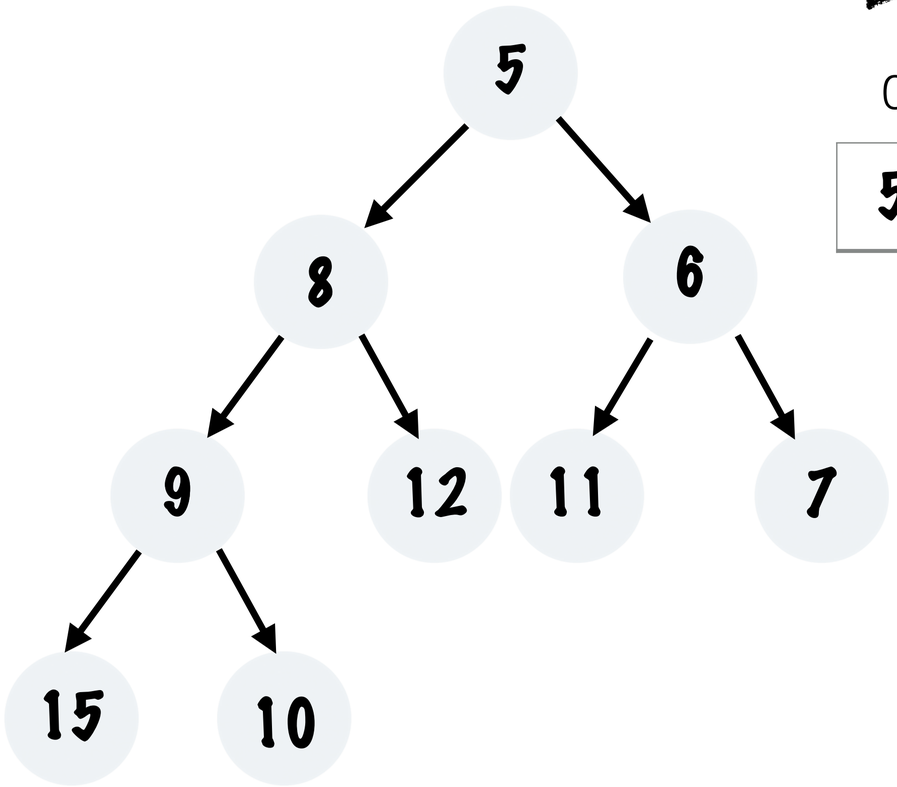


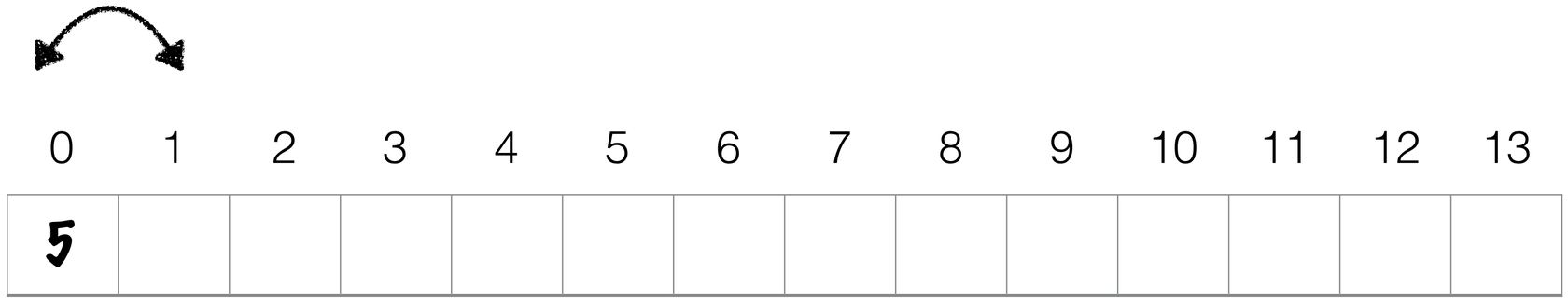
THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT





NODE AT INDEX: `i`

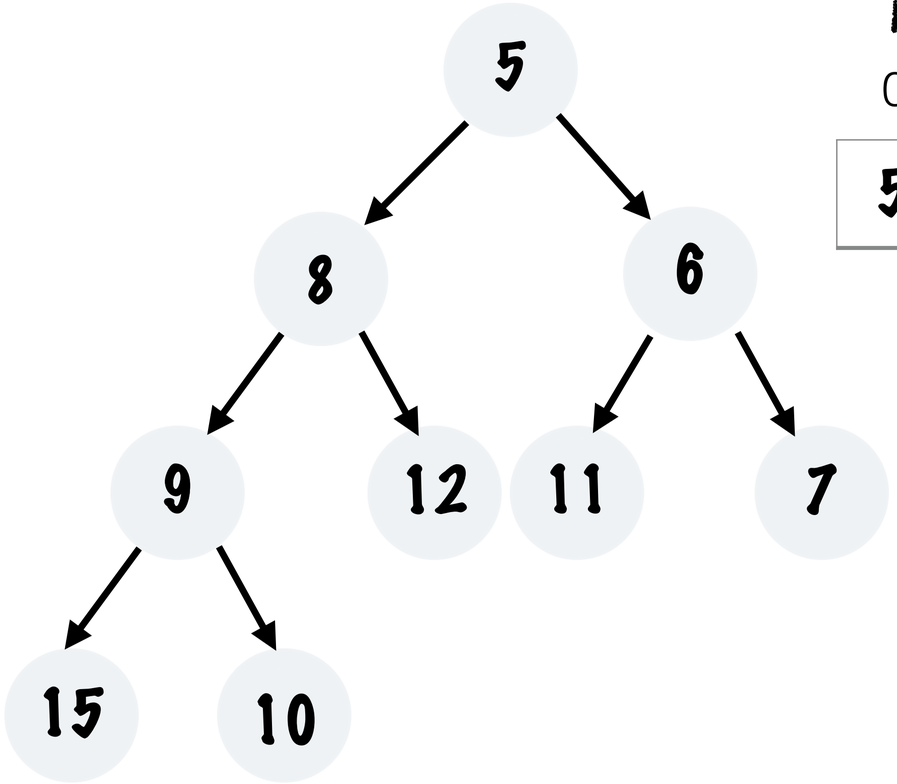
HAS A LEFT CHILD AT INDEX: `2i + 1`

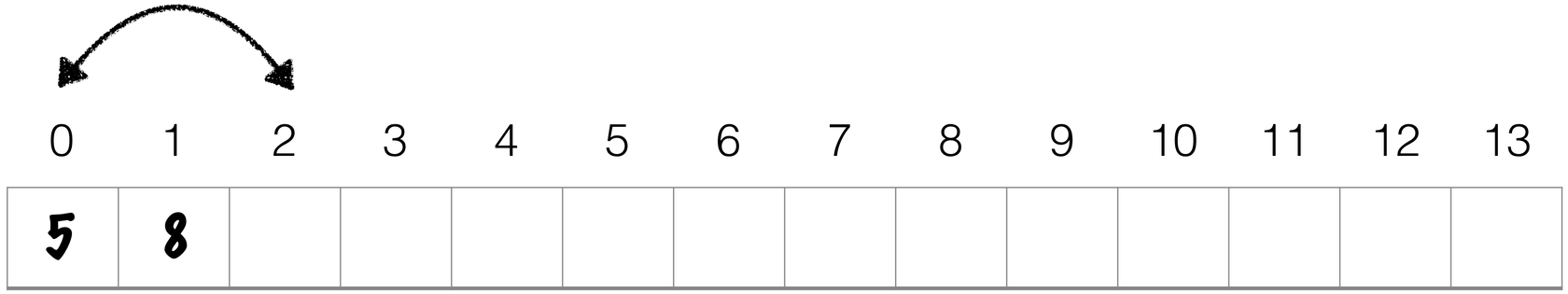
THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT





NODE AT INDEX: `i`

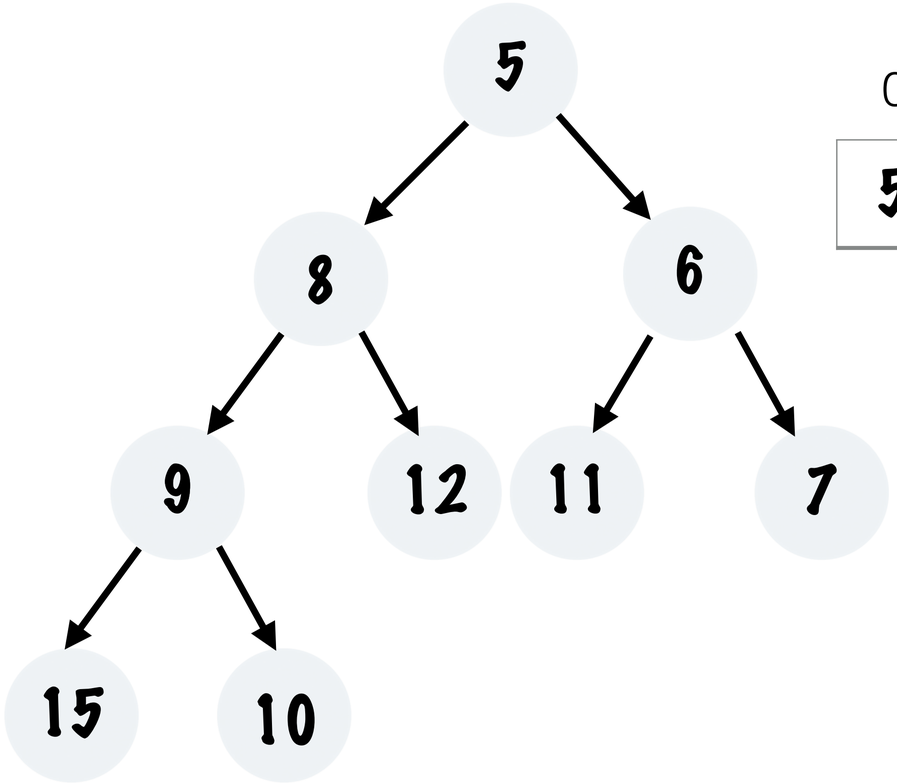
HAS A RIGHT CHILD AT INDEX: `2i + 2`

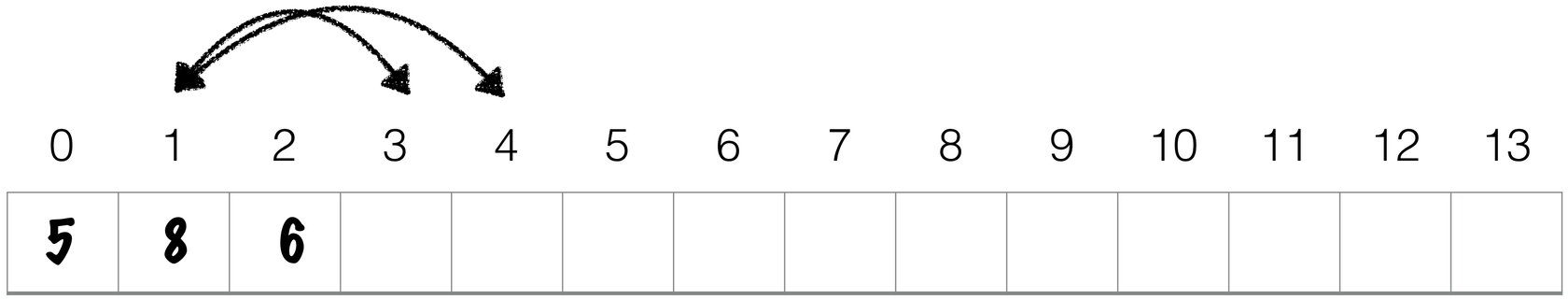
THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT





NODE AT INDEX: `i`

HAS A LEFT CHILD AT INDEX: `2i + 1`

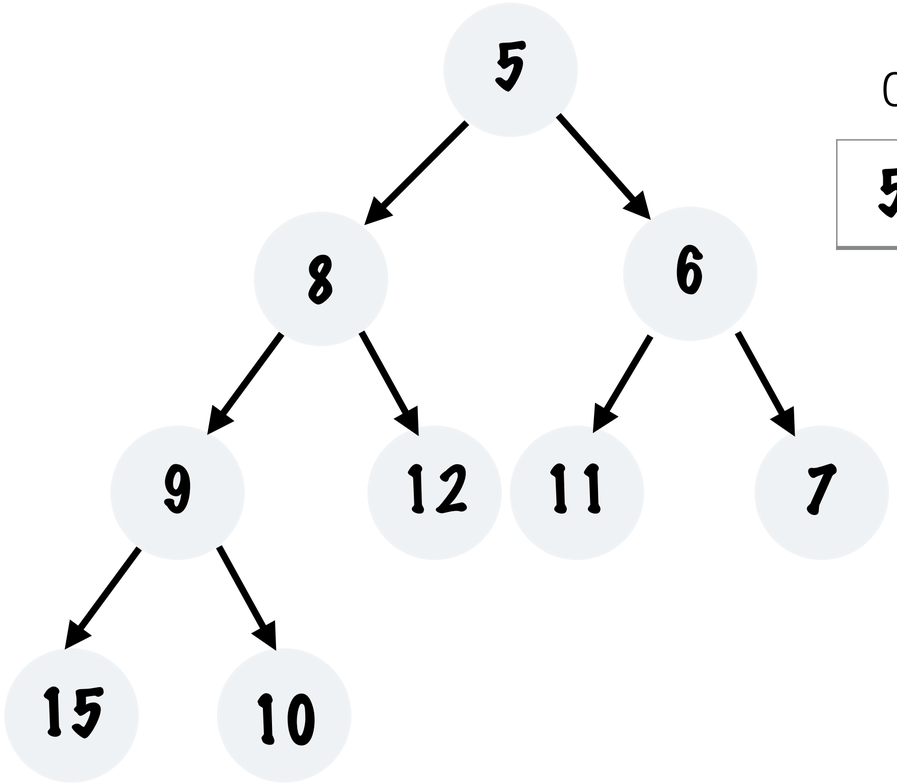
HAS A RIGHT CHILD AT INDEX: `2i + 2`

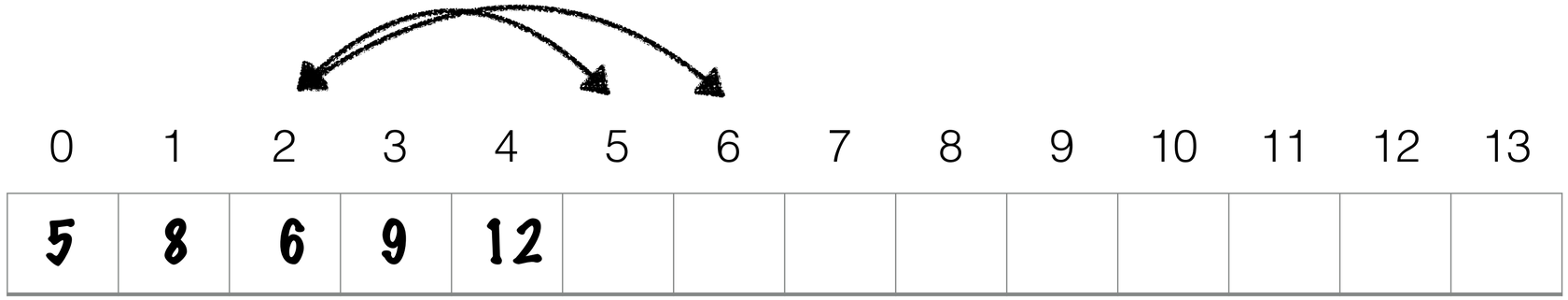
THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT





NODE AT INDEX: `i`

HAS A LEFT CHILD AT INDEX: `2i + 1`

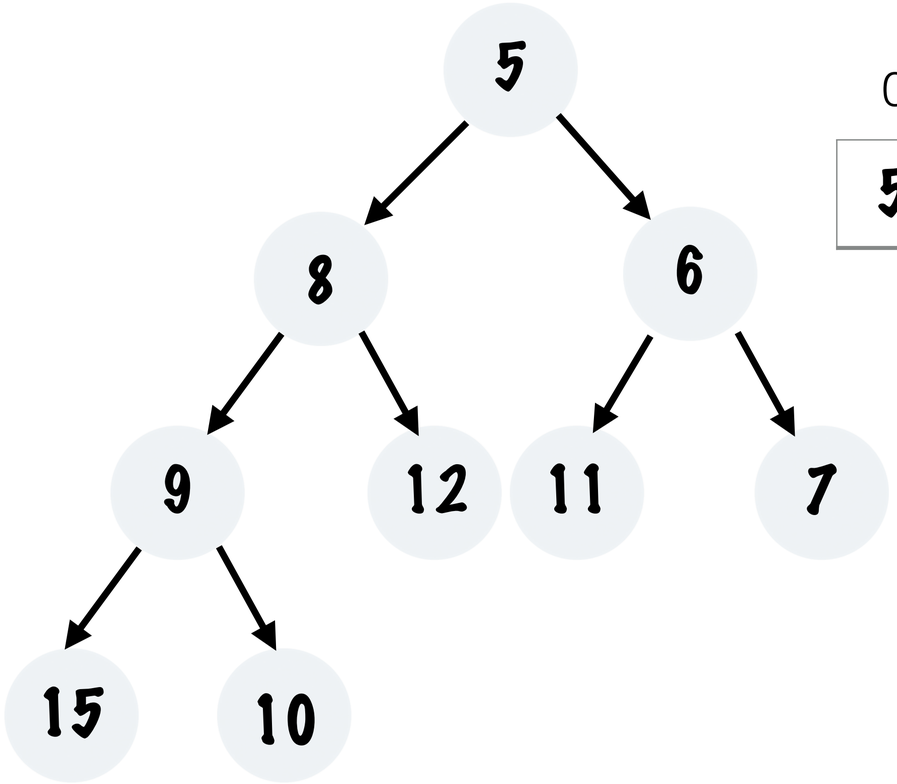
HAS A RIGHT CHILD AT INDEX: `2i + 2`

THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT



0	1	2	3	4	5	6	7	8	9	10	11	12	13
5	8	6	9	12									

NODE AT INDEX: `i`

HAS A LEFT CHILD AT INDEX: `2i + 1`

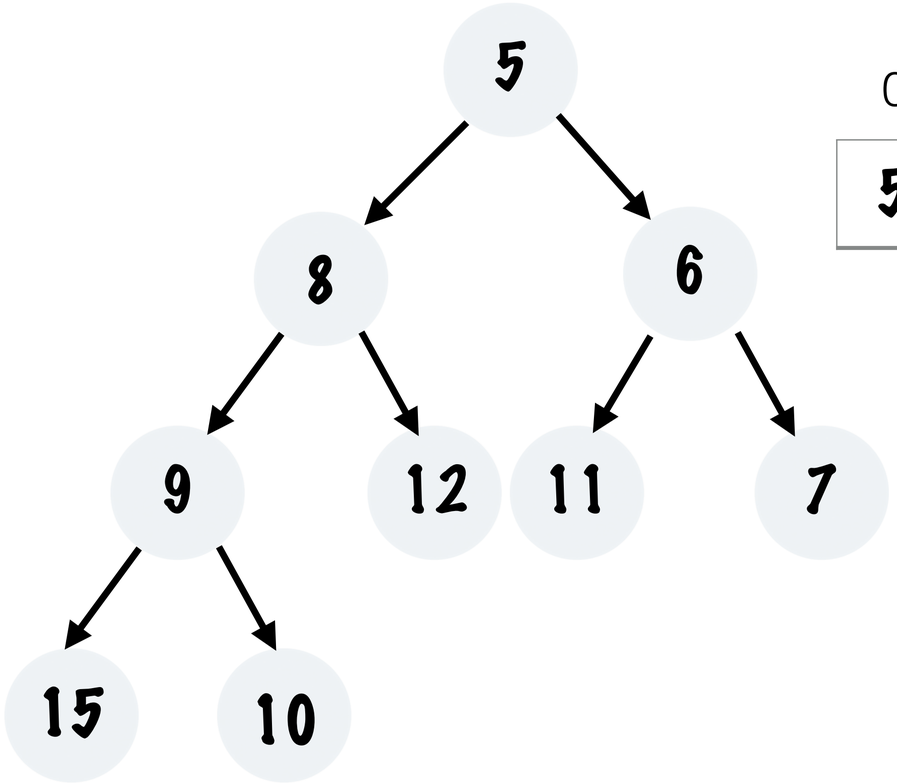
HAS A RIGHT CHILD AT INDEX: `2i + 2`

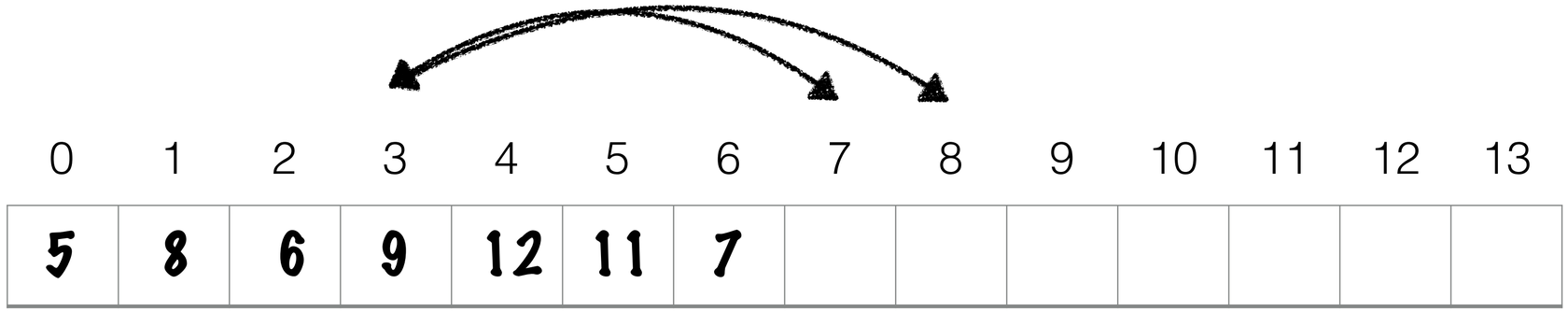
THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT





NODE AT INDEX: `i`

HAS A LEFT CHILD AT INDEX: `2i + 1`

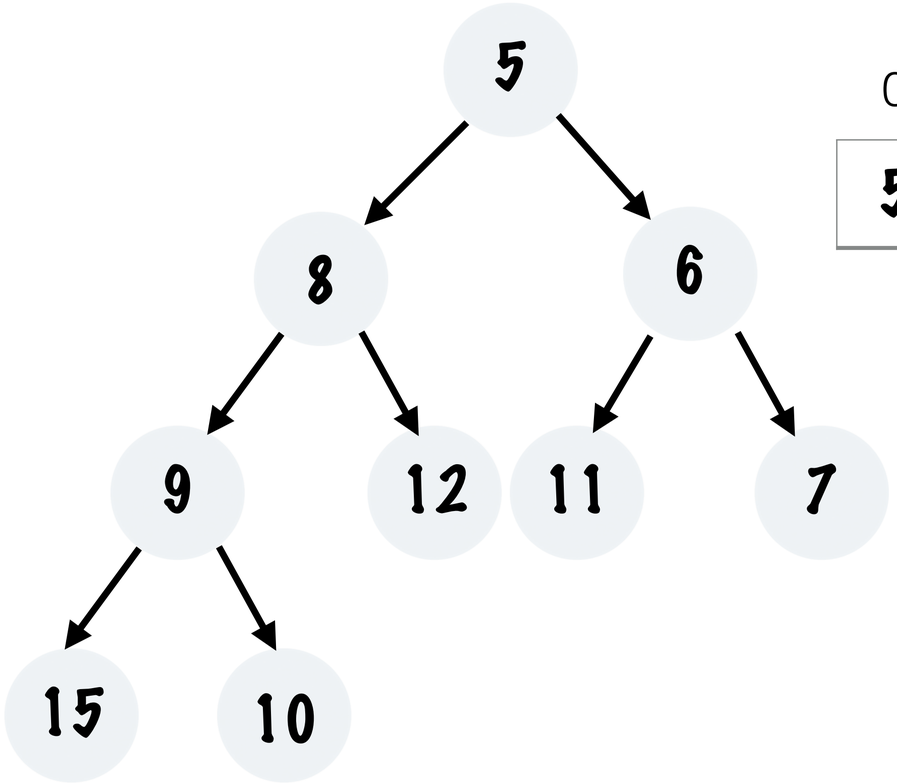
HAS A RIGHT CHILD AT INDEX: `2i + 2`

THE BINARY HEAP
IMPLEMENTATION

GET LEFT CHILD

GET RIGHT CHILD

GET PARENT



0	1	2	3	4	5	6	7	8	9	10	11	12	13
5	8	6	9	12	11	7	15	10					

NODE AT INDEX: `i`

HAS A LEFT CHILD AT INDEX: `2i + 1`

HAS A RIGHT CHILD AT INDEX: `2i + 2`