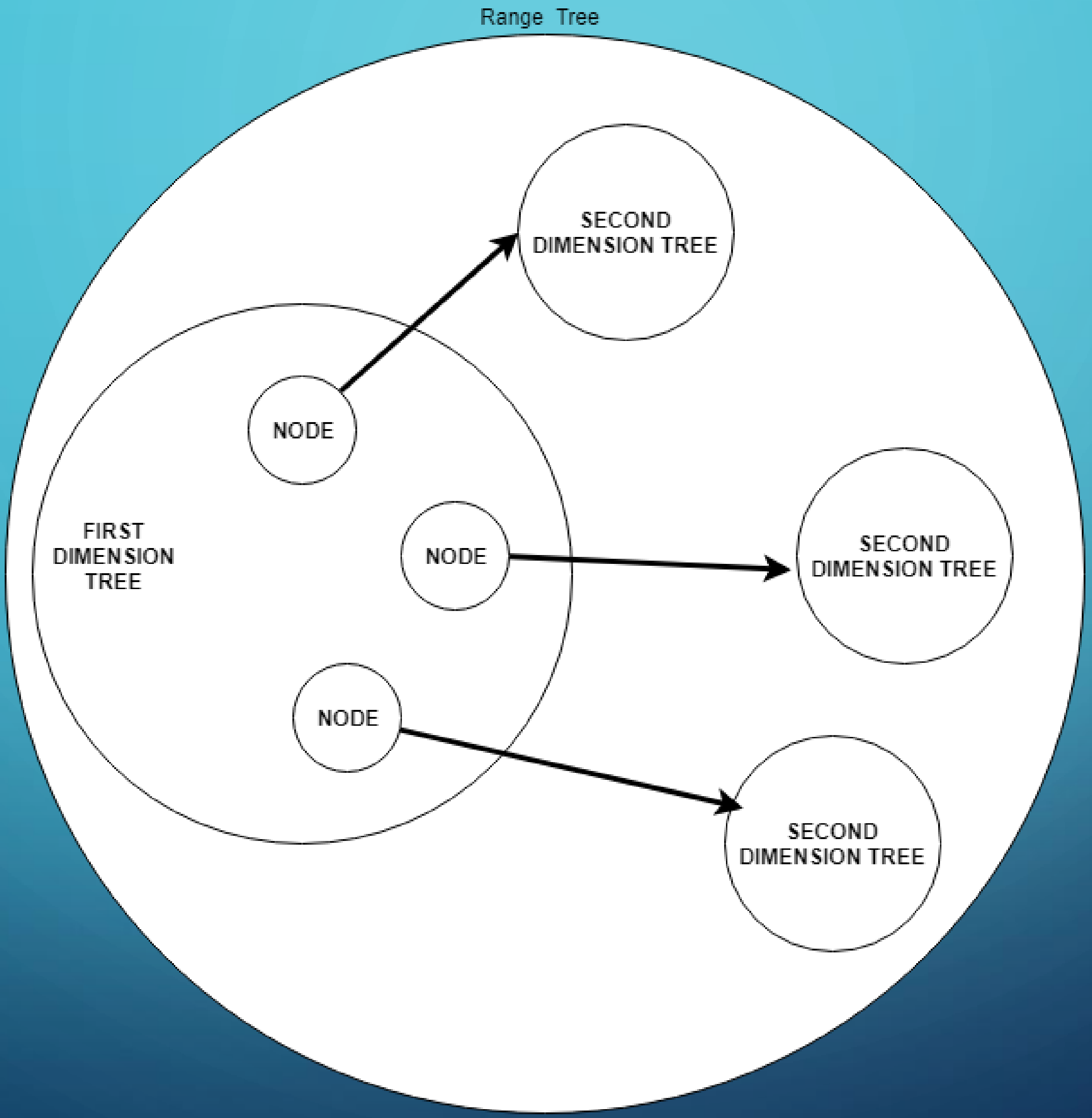
RANGE TREE

IOANNIS-DIMITRIOS STEFANOU

CLASSES


Range Tree
FIRST DIMENSION TREE
NODE
NODE
NODE
SECOND DIMENSION TREE
SECOND DIMENSION TREE
SECOND DIMENSION TREE

FIRST WE SORT OUR DATA ON THE FIRST DIMENSION

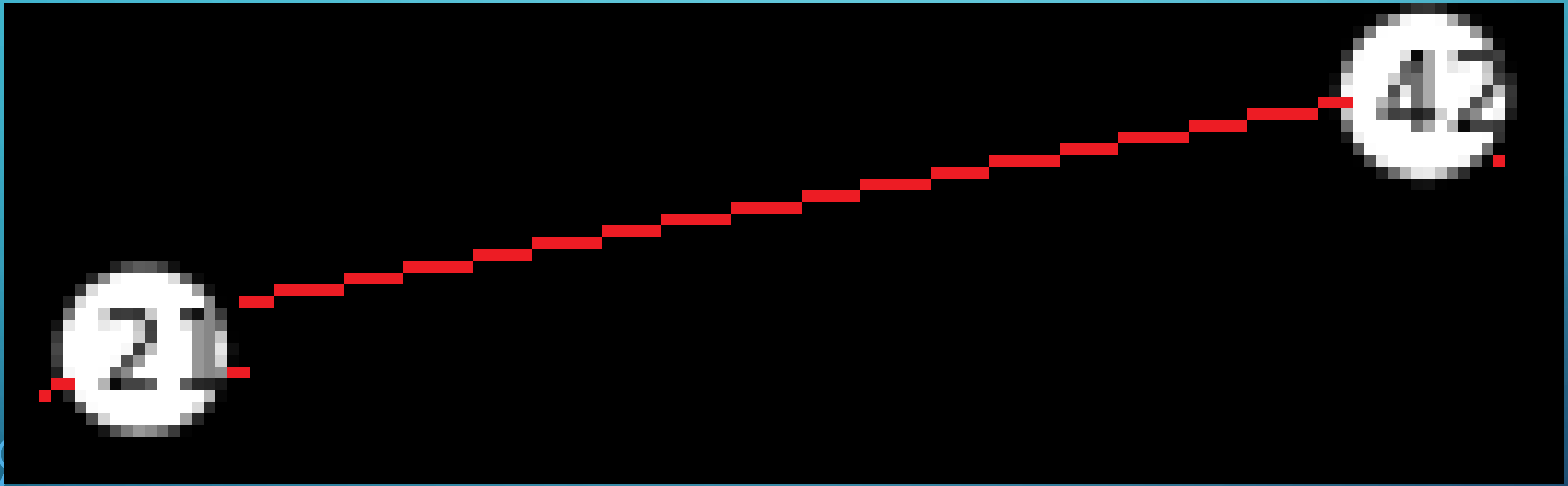
6	15	17	21	24	33	42	51	52	57	65	73	78
0	1	2	3	4	5	6	7	8	9	10	11	12

Then we find the middle element and make it our root (42)

(Left + right) / 2 = 6

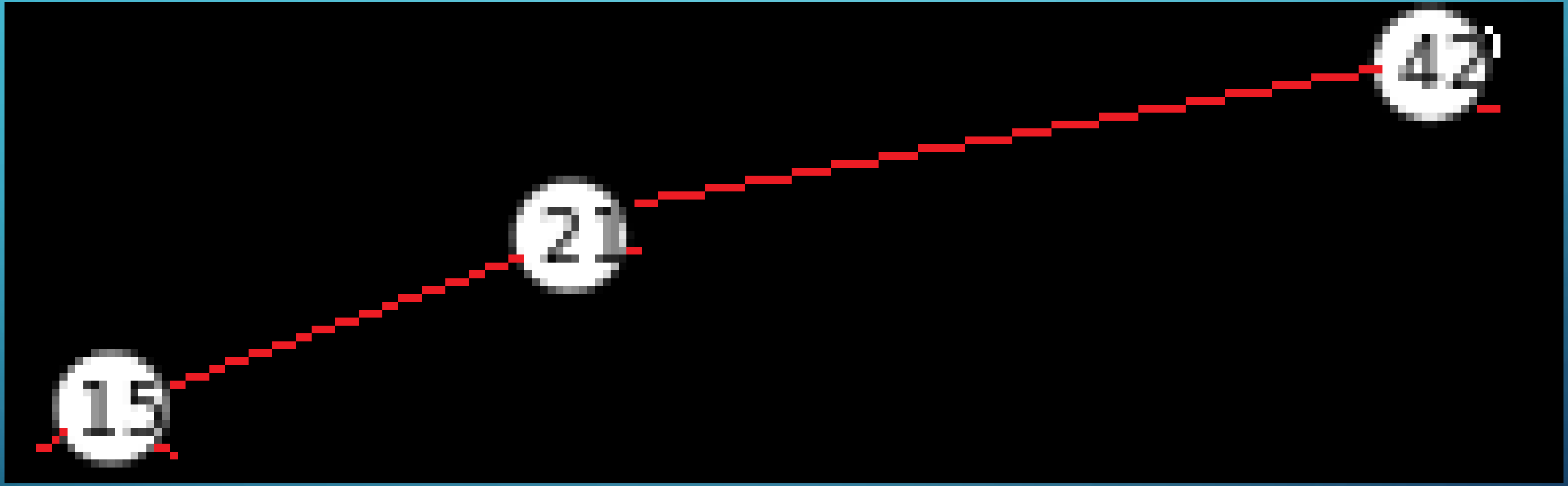
6	15	17	21	24	33	42	51	52	57	65	73	78
0	1	2	3	4	5	6	7	8	9	10	11	12

Then we find the middle element of the left part and it becomes the left child (21)



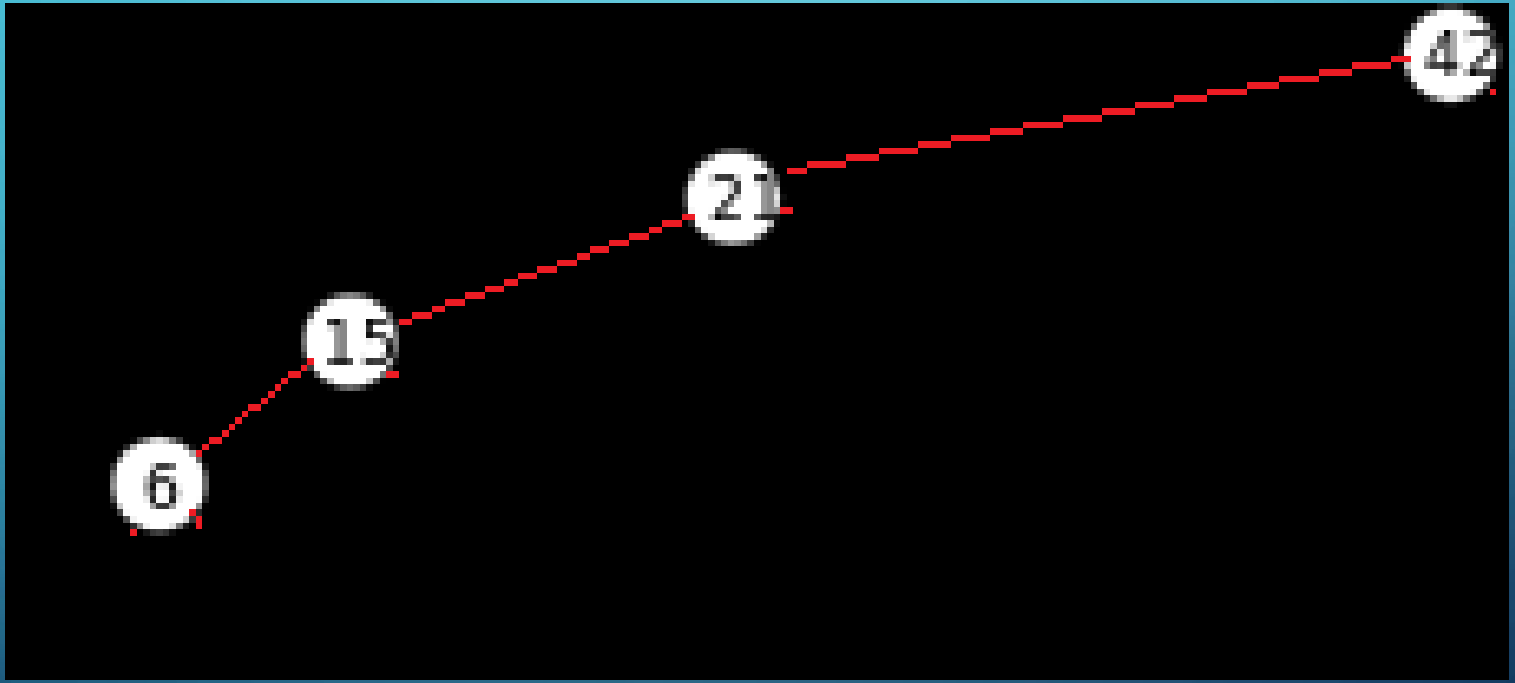
6	15	17	21	24	33	42	51	52	57	65	73	78
0	1	2	3	4	5	6	7	8	9	10	11	12

Following this procedure until we reach the left and right pointers are the equal



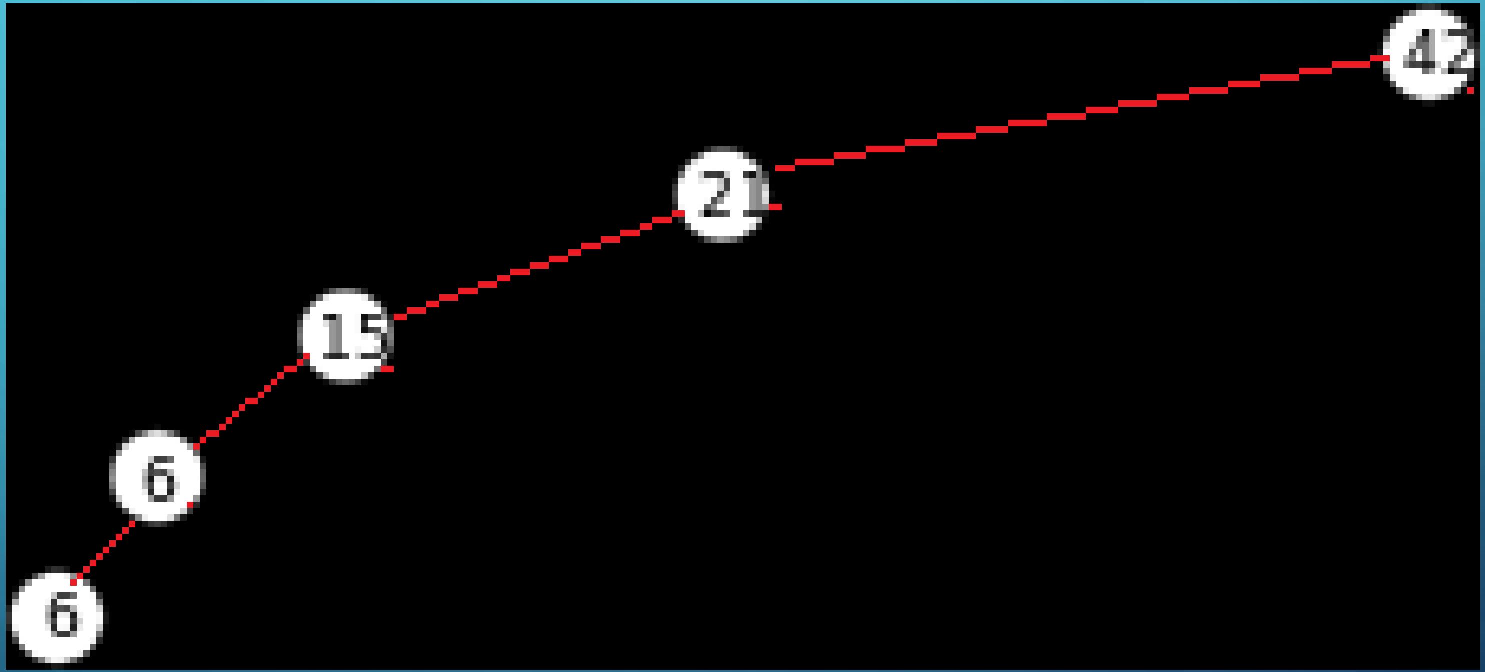
6	15	17	21	24	33	42	51	52	57	65	73	78
0	1	2	3	4	5	6	7	8	9	10	11	12

Following this procedure until we reach the left and right pointers are the equal



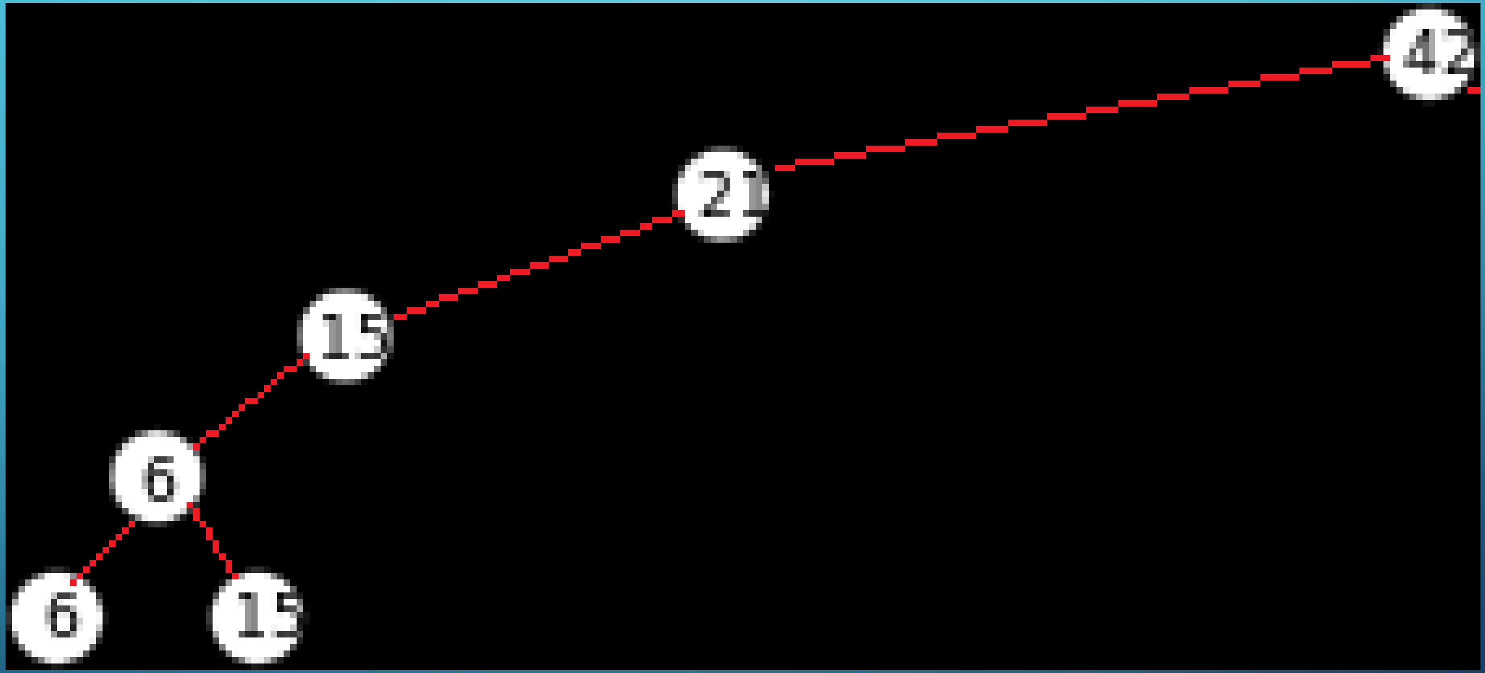
6	15	17	21	24	33	42	51	52	57	65	73	78
0	1	2	3	4	5	6	7	8	9	10	11	12

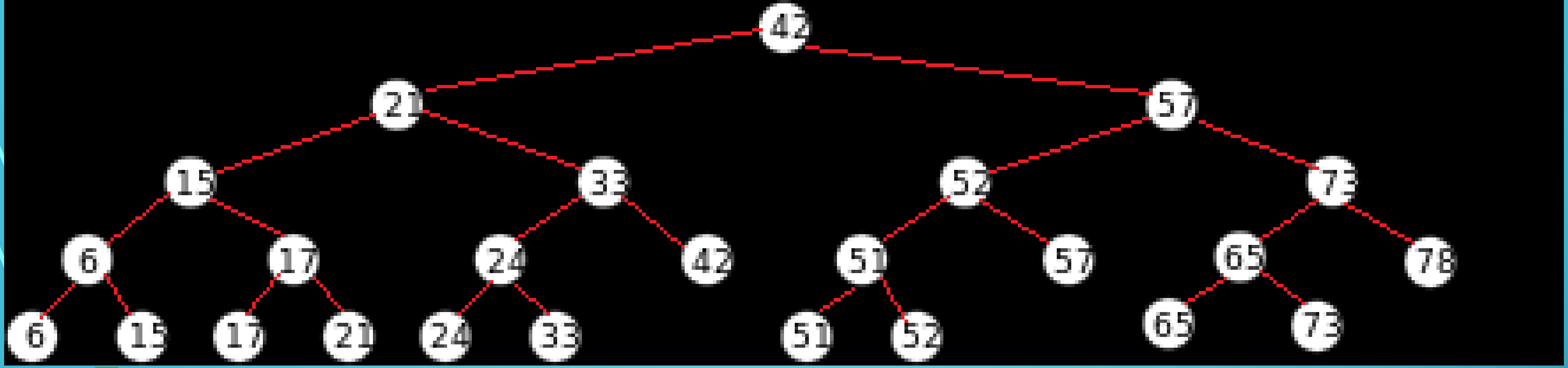
After reaching a leaf if it is a left child we backtrack and create the right node of the parent

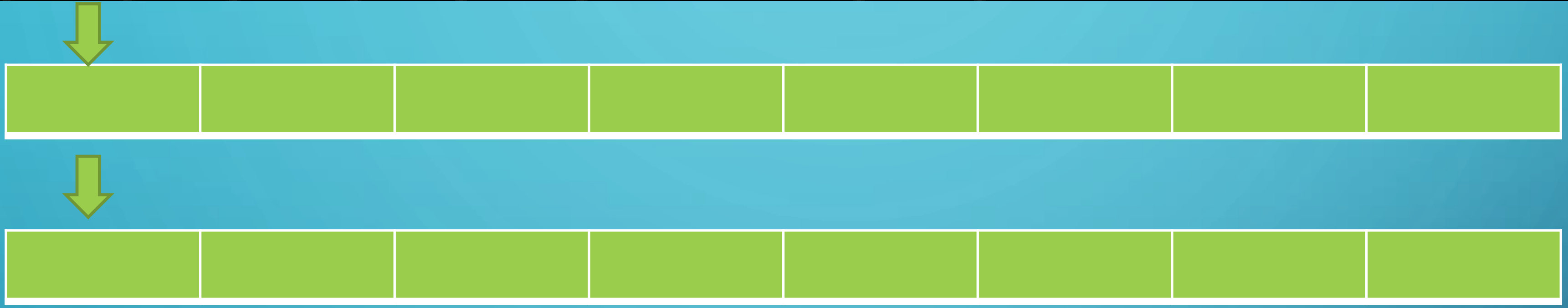


6	15	17	21	24	33	42	51	52	57	65	73	78
0	1	2	3	4	5	6	7	8	9	10	11	12

After reaching a leaf if it is a right child we backtrack and create subtrees for the other dimensions



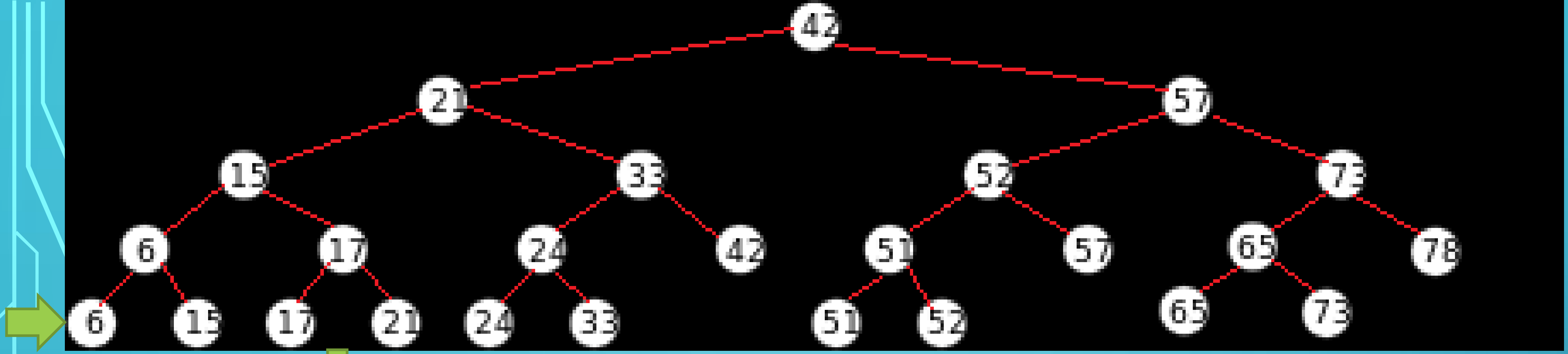




NODE_CONTENTS
SIZE N

node_sorting_helper
Size Log<base2> n

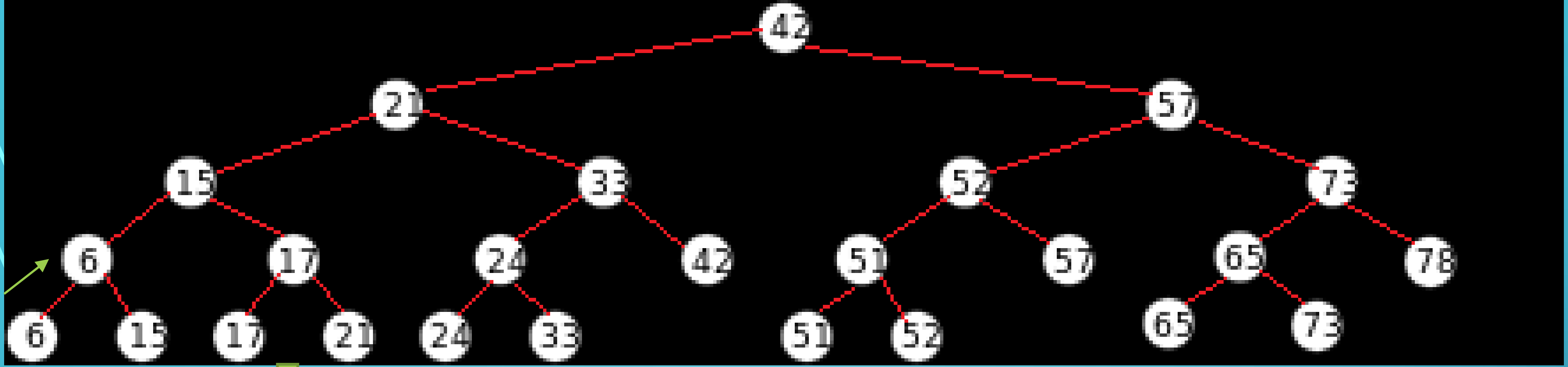
node_content_counter=0 node_sorting_helper_counter=0 amount_of_numbers=0



Node 6								NODE_CONTENTS
1								node_sorting_helper

node_content_counter=1 **node_sorting_helper_counter=1** amount_of_numbers=0

When we reach a leaf we save the node to the contents and the value 1 in the sort helper



Node 6							

NODE_CONTENTS

1							

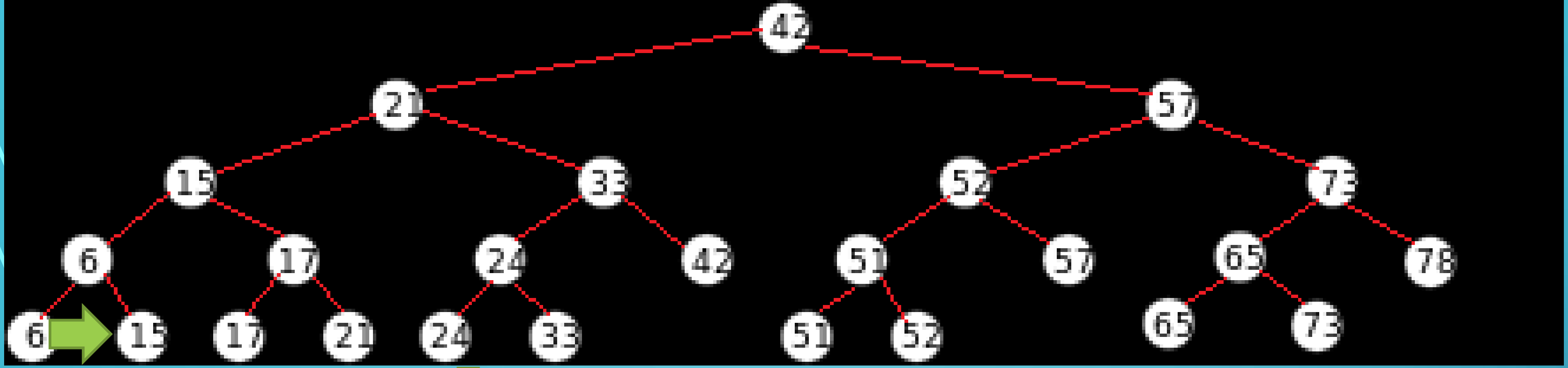
node_sorting_helper

node_content_counter=1

node_sorting_helper_counter=1

amount_of_numbers=0

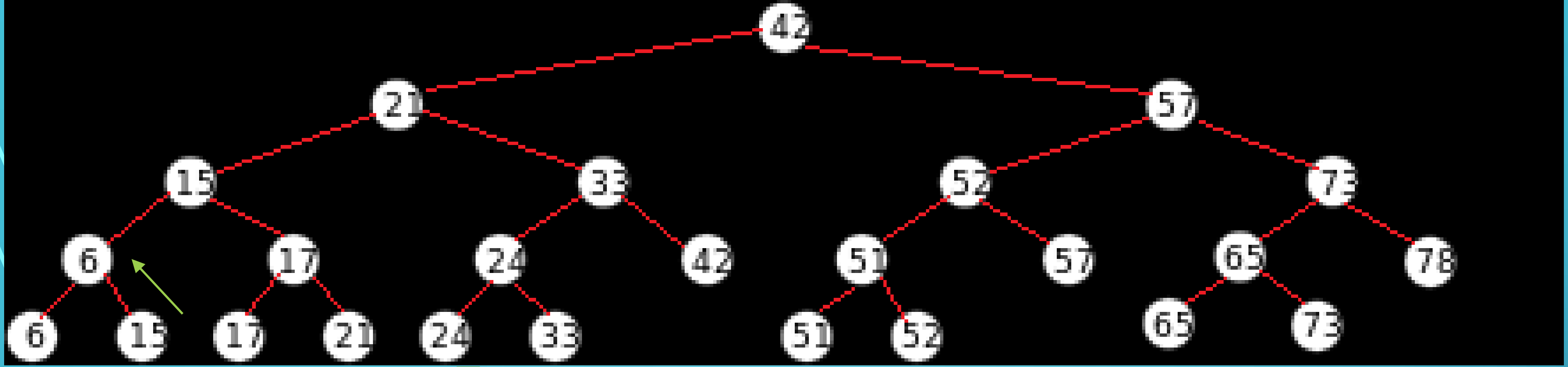
Also increase the pointers values



Node 6	**Node 15**							NODE_CONTENTS

1	**1**							node_sorting_helper

node_content_counter=2 **node_sorting_helper_counter=2** amount_of_numbers=0



Node 6	Node 15						

NODE_CONTENTS

1	1						

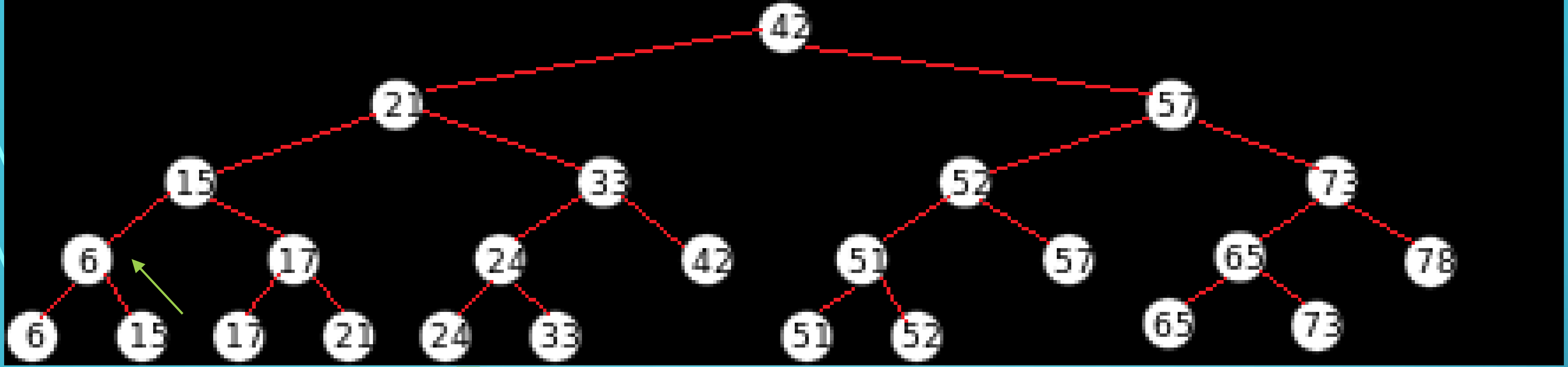
node_sorting_helper

node_content_counter=2

node_sorting_helper_counter=2

amount_of_numbers=node_sorting_helper(counter-2)+(counter-1)

When going up from a right child we want to sort the leaves of this node



Node 6	Node 15							NODE_CONTENTS
1	1							node_sorting_helper

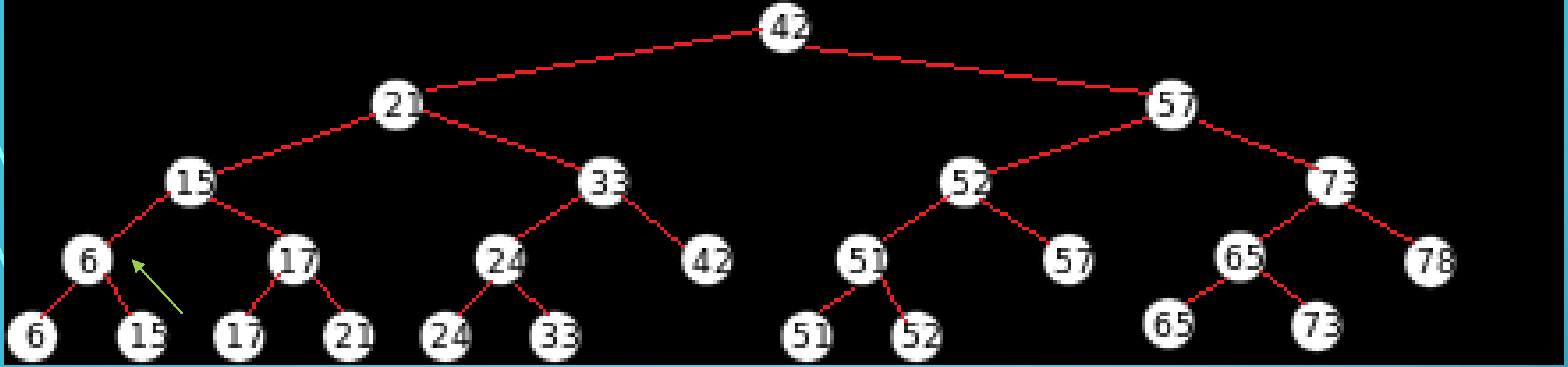
node_content_counter=2

node_sorting_helper_counter=2

amount_of_numbers=2

Sort Node contents (content_counter-amount of numbers) – (content_counter-1)

Sort Node contents (0) – (1) and then construct a tree using the specific values



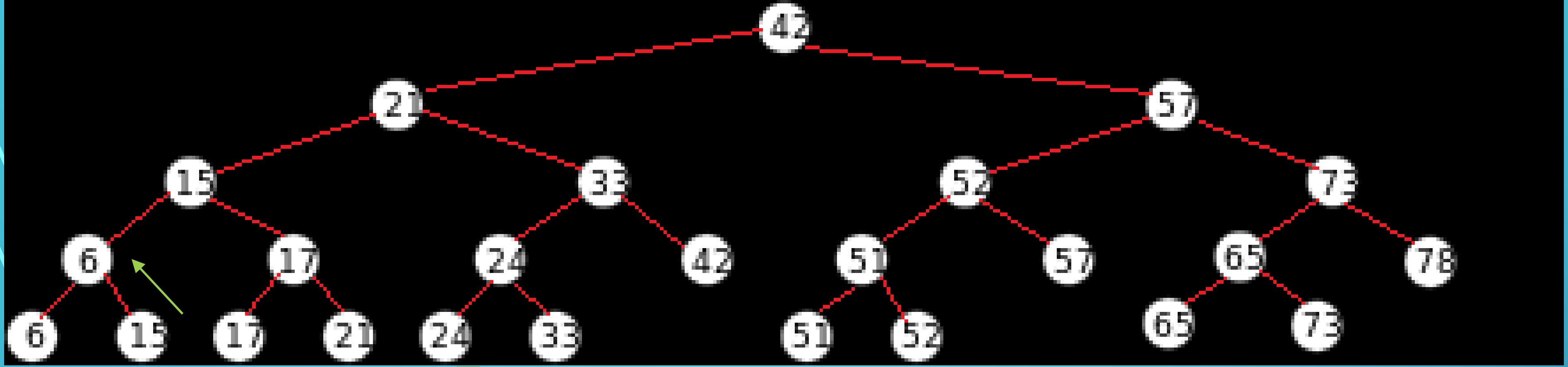
Node 6	Node 15							NODE_CONTENTS
2	1							node_sorting_helper

node_content_counter=2

node_sorting_helper_counter=2

amount_of_numbers=2

Save the summary to 2 slots behind the pointer (2 – 2)



NODE_CONTENTS

Node 6	**Node 15**						

node_sorting_helper

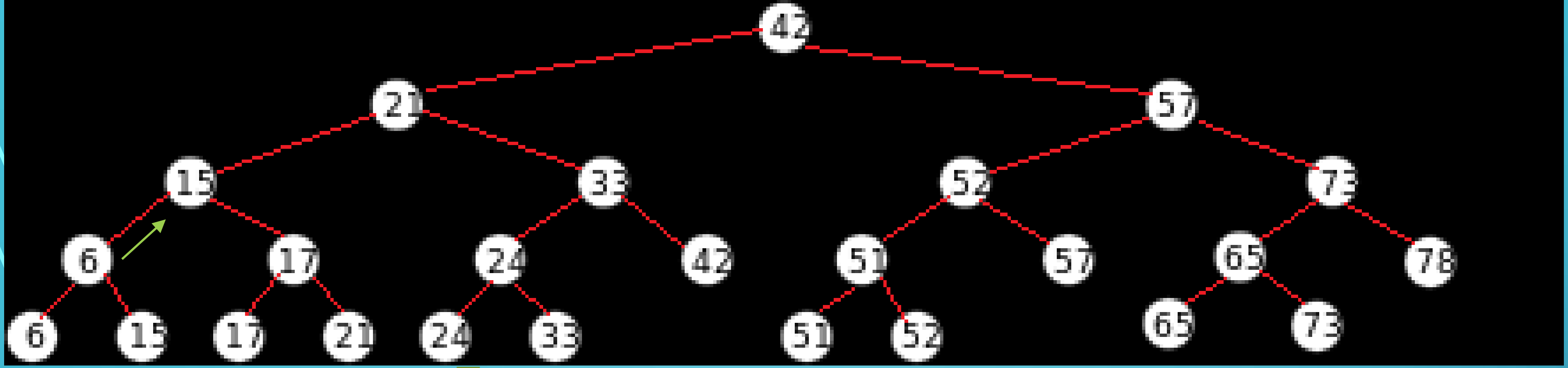
2	**1**						

node_content_counter=2

node_sorting_helper_counter=1

amount_of_numbers=2

Decrease the sorting pointer by 1



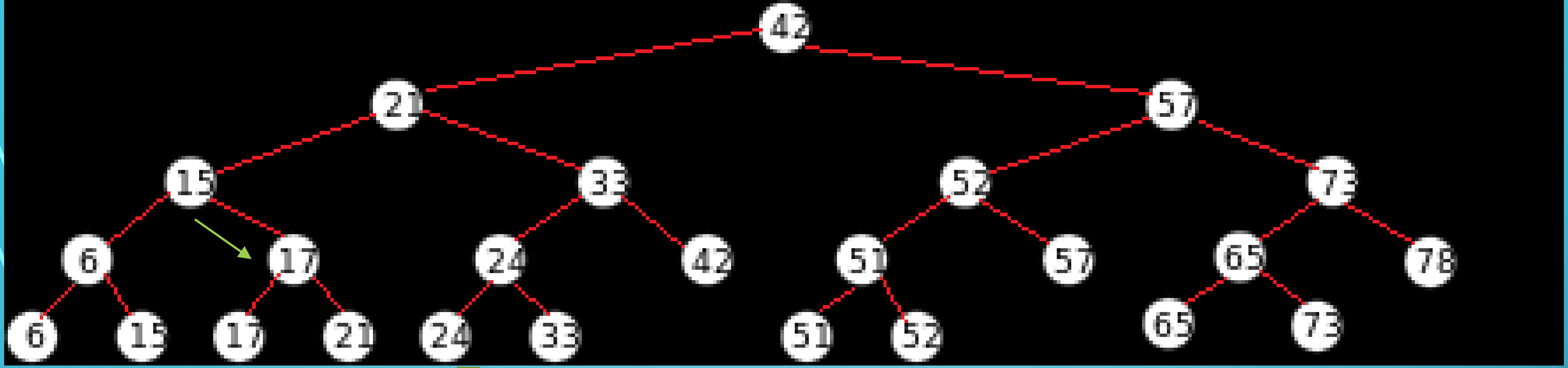
Node 6	Node 15							NODE_CONTENTS

2	1							node_sorting_helper

node_content_counter=2

node_sorting_helper_counter=1

amount_of_numbers=2

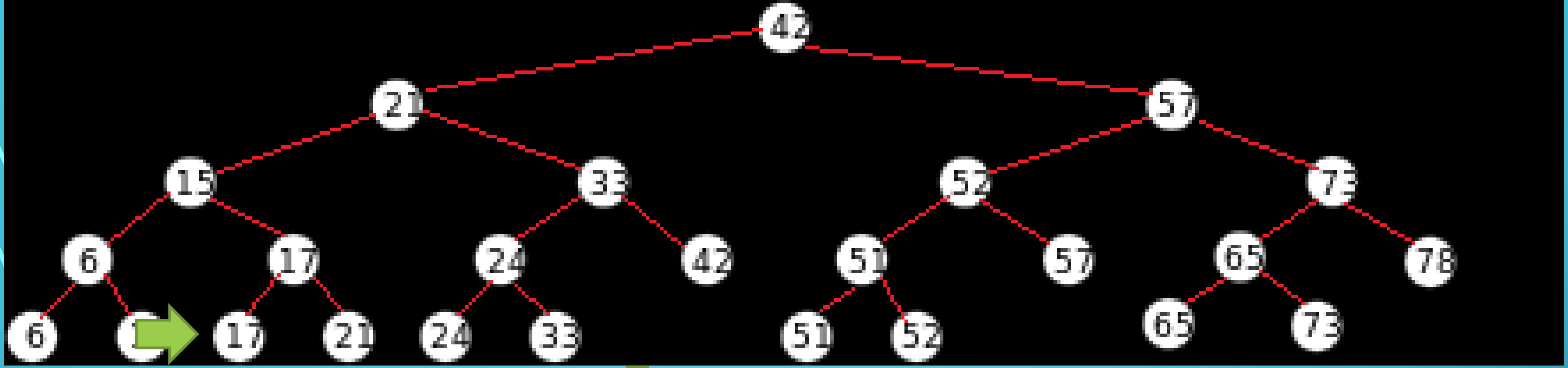


Node 6	Node 15							NODE_CONTENTS
2	1							node_sorting_helper

node_content_counter=2

node_sorting_helper_counter=1

amount_of_numbers=2



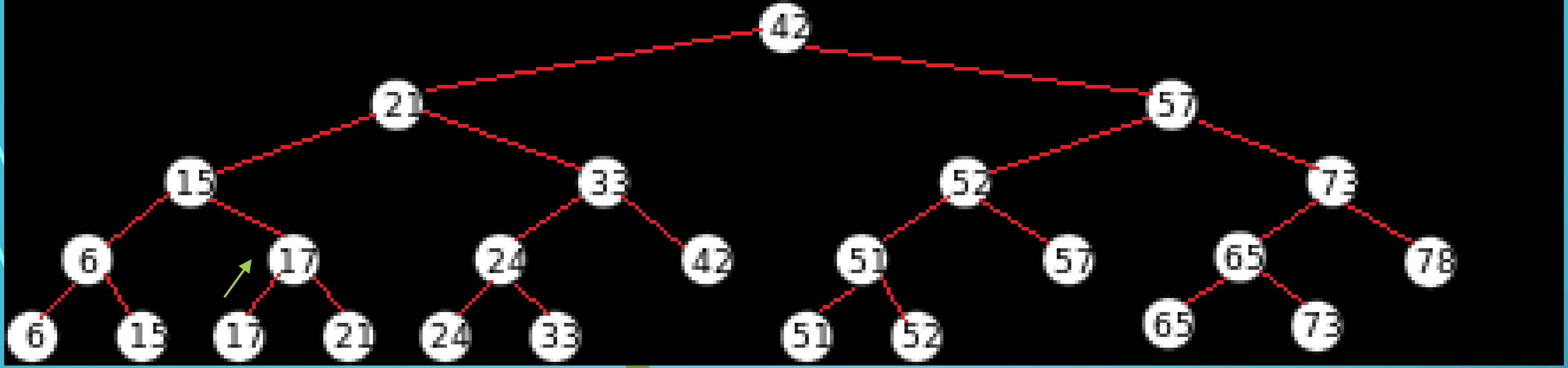
Node 6	Node 15	Node 17						NODE_CONTENTS

2	1	1						node_sorting_helper

node_content_counter=3

node_sorting_helper_counter=2

amount_of_numbers=2

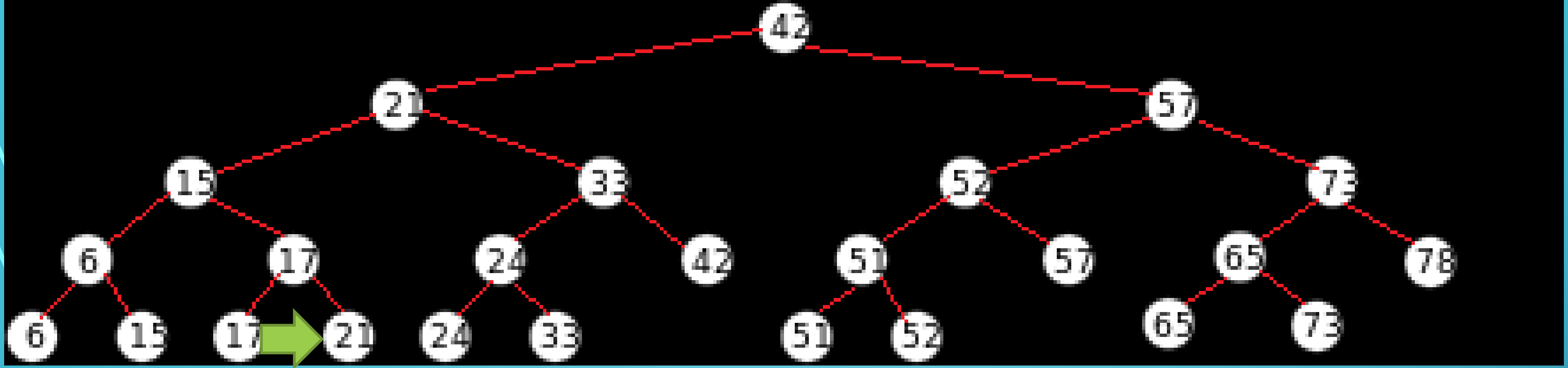


Node 6	**Node 15**	**Node 17**						NODE_CONTENTS
2	**1**	**1**						node_sorting_helper

node_content_counter=3

node_sorting_helper_counter=2

amount_of_numbers=2



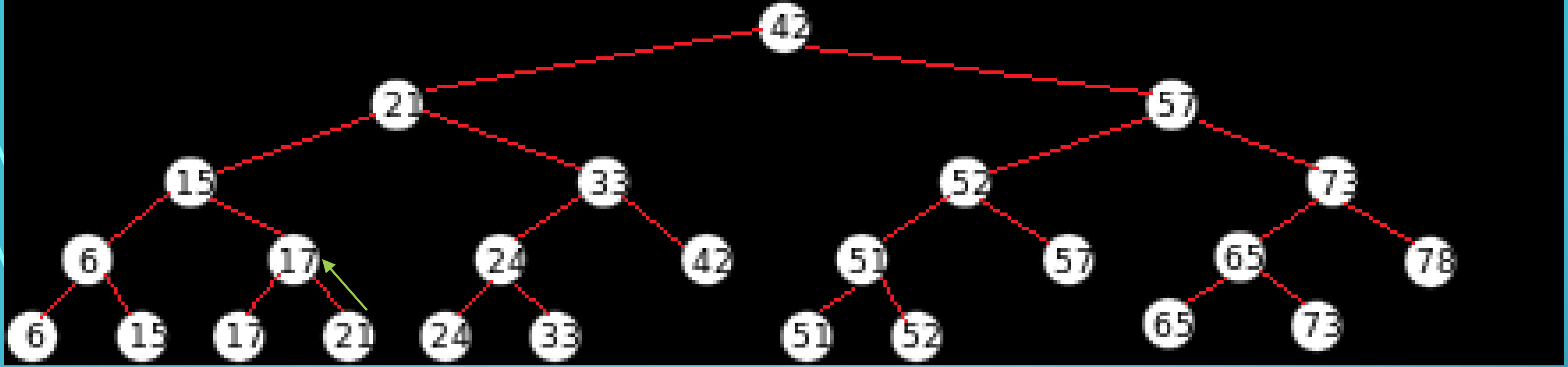
Node 6	**Node 15**	**Node 17**	**Node 21**					NODE_CONTENTS

2	**1**	**1**	**1**					node_sorting_helper

node_content_counter=4

node_sorting_helper_counter=3

amount_of_numbers=2



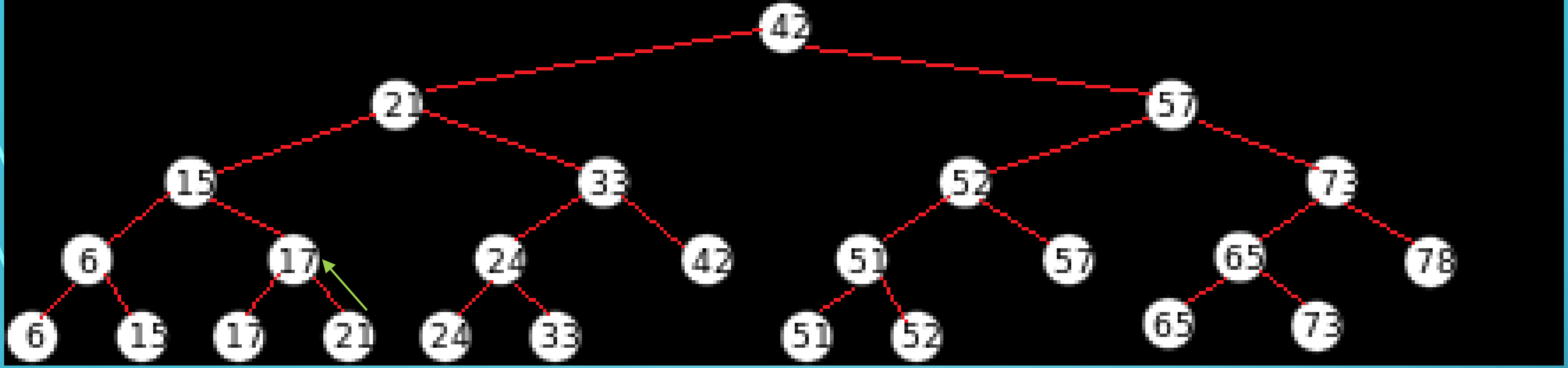
Node 6	Node 15	Node 17	Node 21					NODE_CONTENTS
2	1	1	1					node_sorting_helper

node_content_counter=4

node_sorting_helper_counter=3

amount_of_numbers=2

Sort Node contents (4-2) – (4-1)

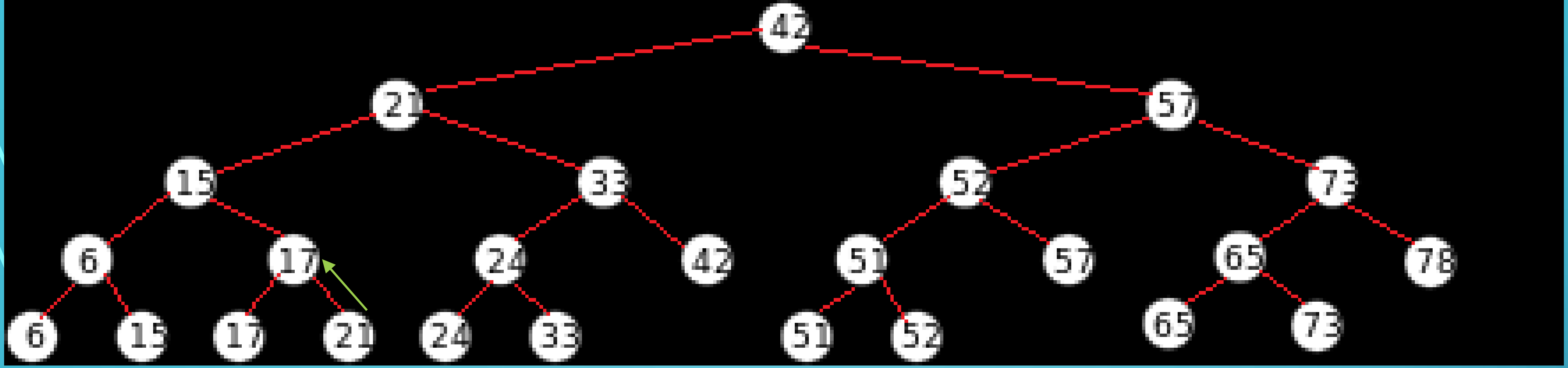


Node 6	Node 15	Node 17	Node 21					NODE_CONTENTS
2	1	1	1					node_sorting_helper

node_content_counter=4

node_sorting_helper_counter=3

amount_of_numbers=2

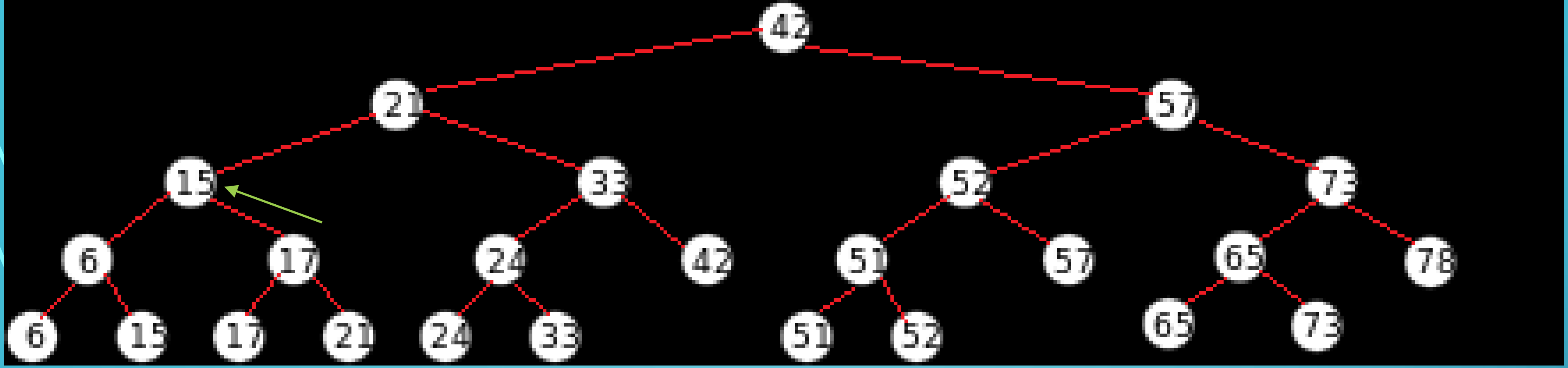


Node 6	Node 15	Node 17	Node 21					NODE_CONTENTS
2	2	1	1					node_sorting_helper

node_content_counter=4

node_sorting_helper_counter=2

amount_of_numbers=2



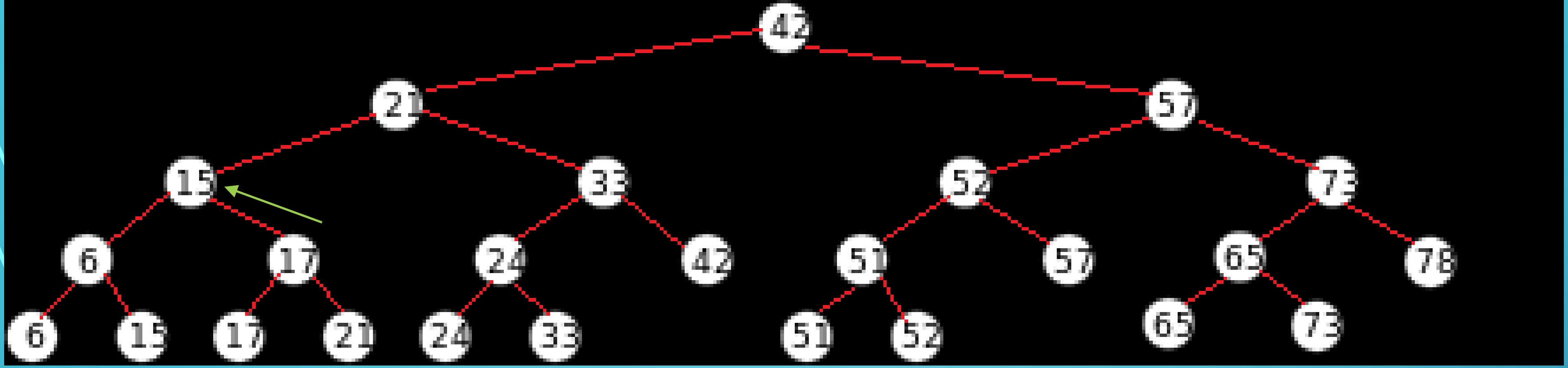
Node 6	Node 15	Node 17	Node 21					NODE_CONTENTS
2	2	1	1					node_sorting_helper

node_content_counter=4

node_sorting_helper_counter=2

amount_of_numbers=4

Sort Node contents (4-4) – (4-1)



Node 6	**Node 15**	**Node 17**	**Node 21**					NODE_CONTENTS

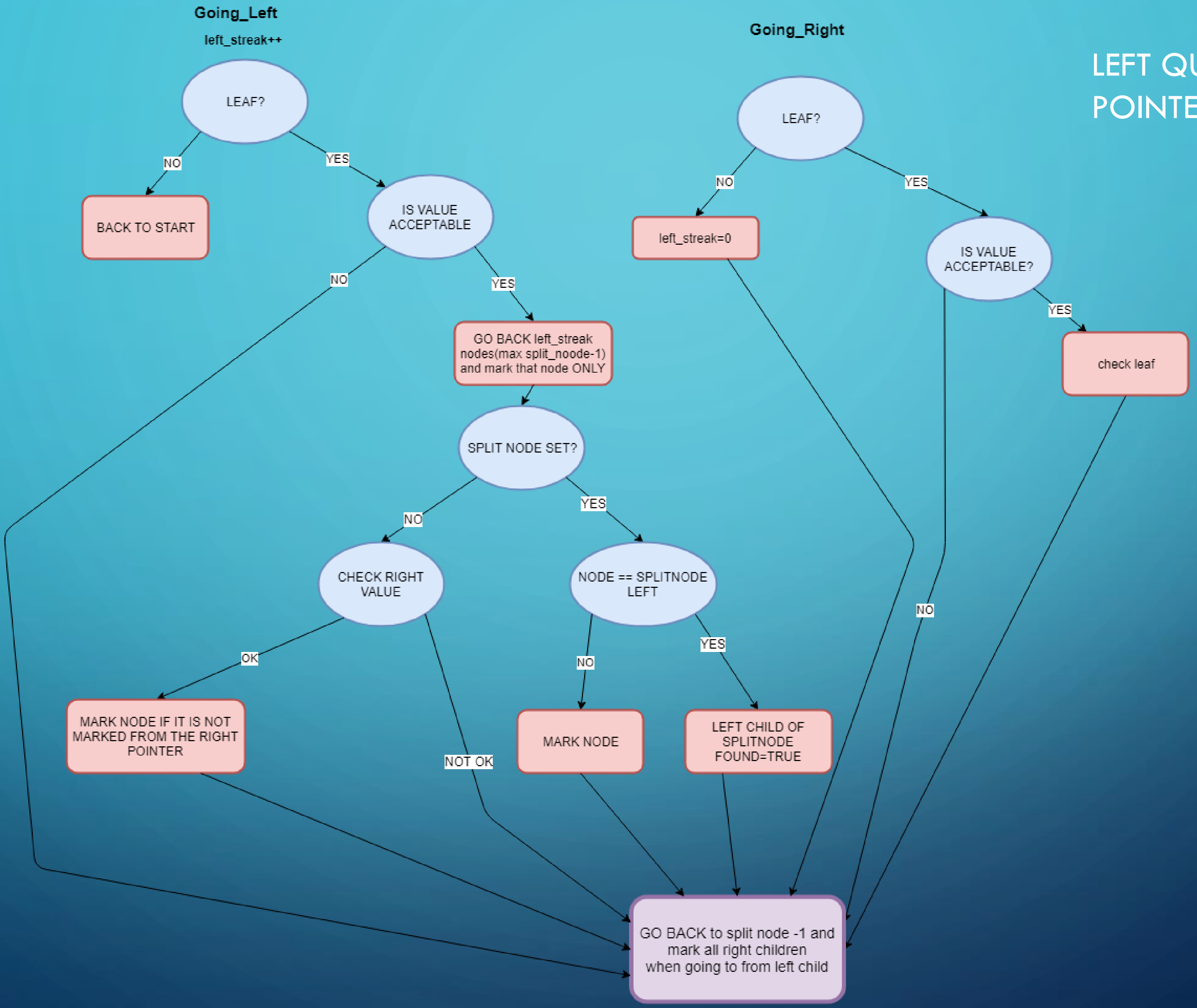
4	**2**	**1**	**1**					node_sorting_helper

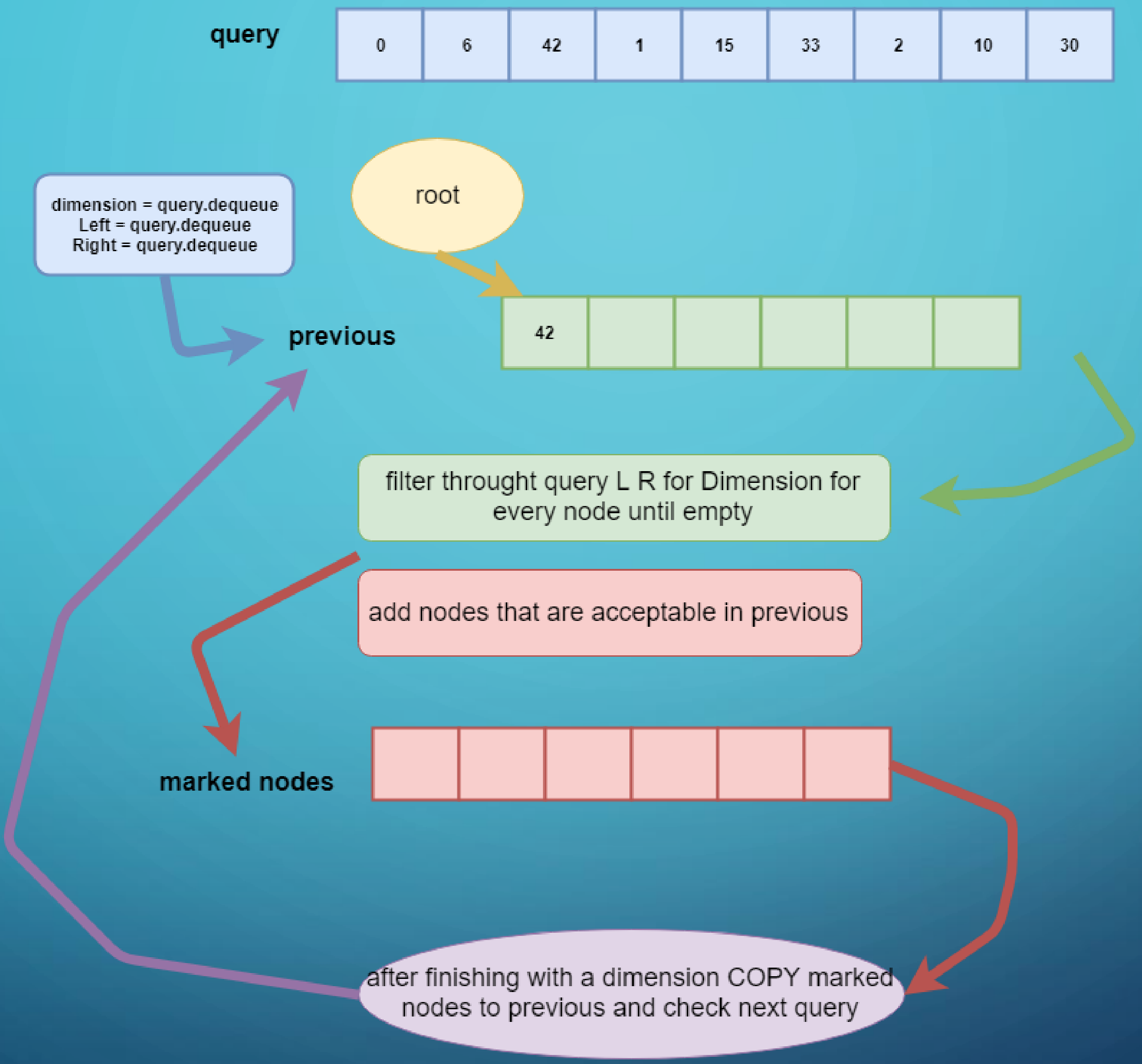
node_content_counter=4

node_sorting_helper_counter=1

amount_of_numbers=4

LEFT QUERY POINTER DIAGRAM


Going_Left
left_streak++
LEAF?
NO
BACK TO START
YES
IS VALUE ACCEPTABLE
NO
YES
GO BACK left_streak nodes(max split_noode-1) and mark that node ONLY
SPLIT NODE SET?
NO
YES
CHECK RIGHT VALUE
NODE == SPLITNODE LEFT
OK
NOT OK
NO
YES
MARK NODE IF IT IS NOT MARKED FROM THE RIGHT POINTER
MARK NODE
LEFT CHILD OF SPLITNODE FOUND=TRUE
Going_Right
LEAF?
NO
YES
left_streak=0
IS VALUE ACCEPTABLE?
YES
NO
check leaf
GO BACK to split node -1 and mark all right children when going to from left child

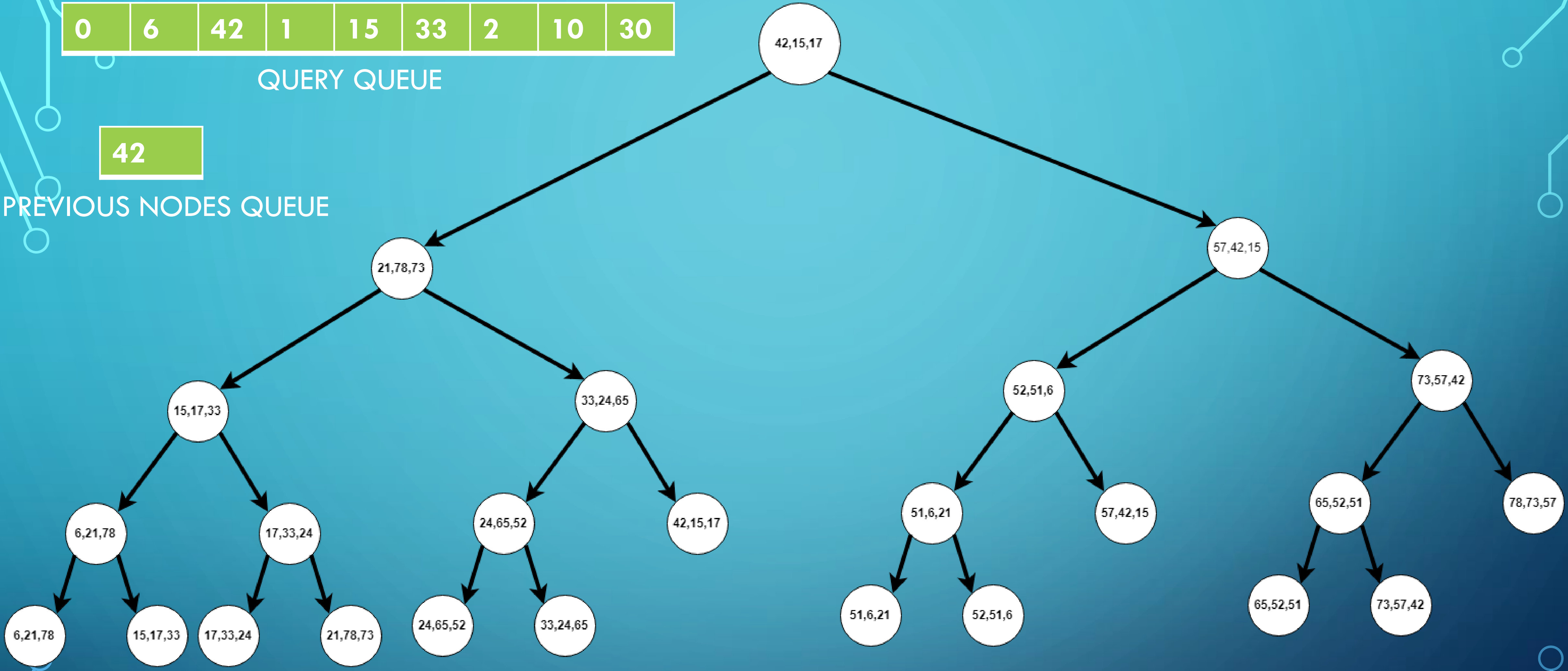

query
0
6
42
1
15
33
2
10
30
root
dimension = query.dequeue
Left = query.dequeue
Right = query.dequeue
previous
42
filter throught query L R for Dimension for every node until empty
add nodes that are acceptable in previous
marked nodes
after finishing with a dimension COPY marked nodes to previous and check next query

0	6	42	1	15	33	2	10	30

QUERY QUEUE

42

PREVIOUS NODES QUEUE



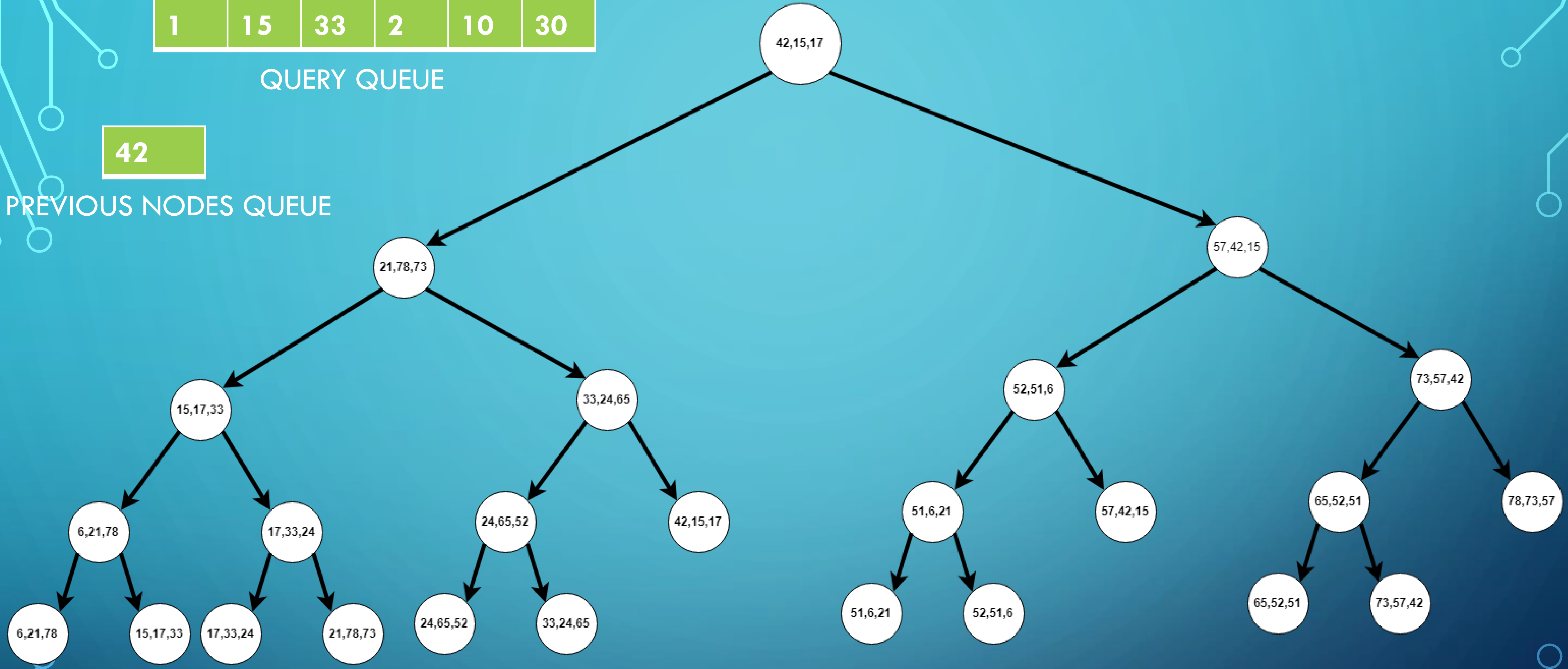
QUERY (6-42,15-33,10-30)

1	15	33	2	10	30

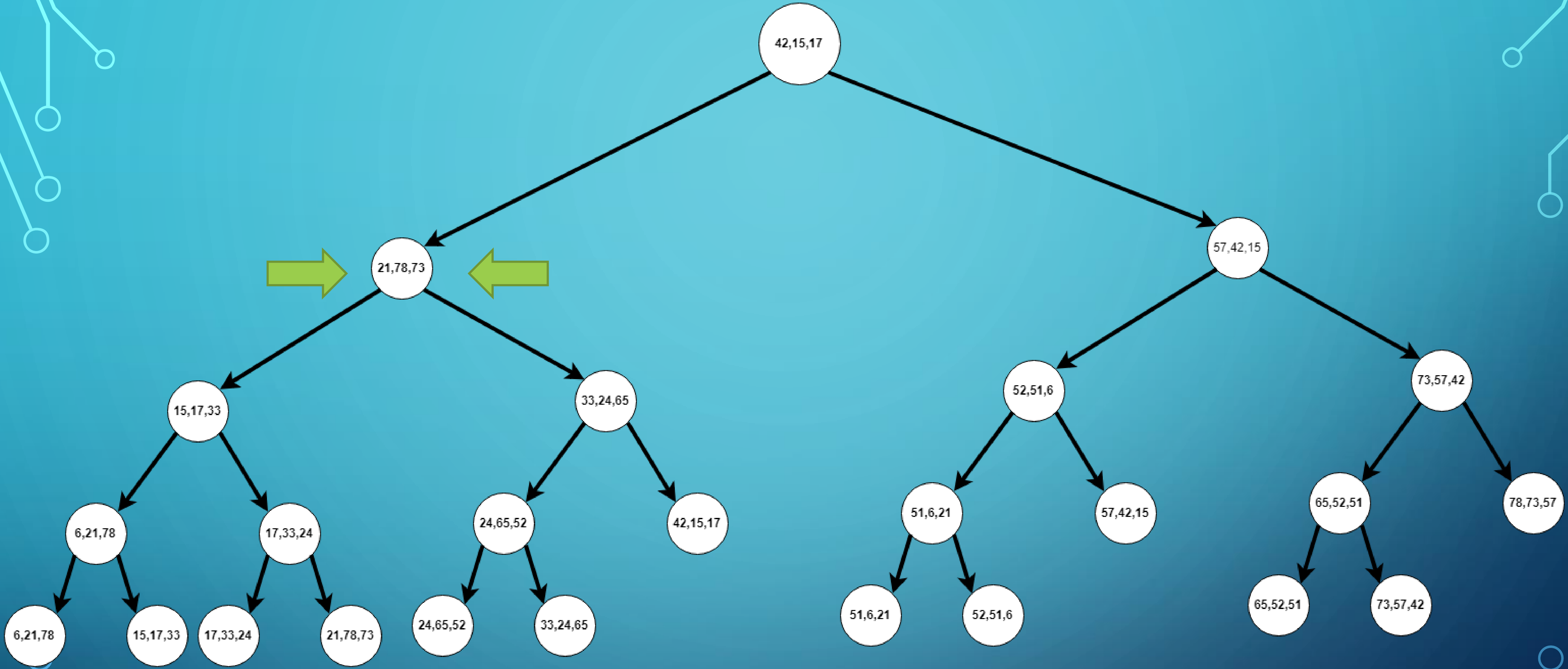
QUERY QUEUE

42

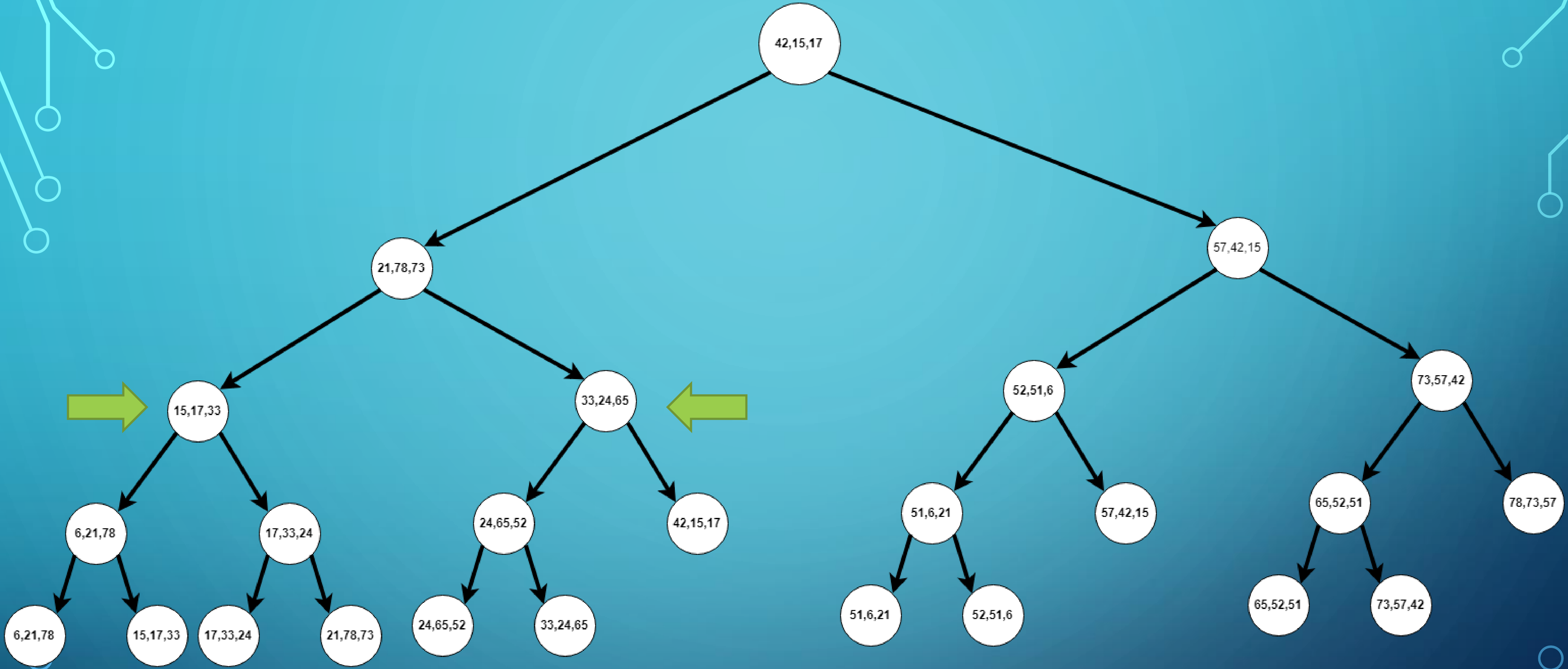
PREVIOUS NODES QUEUE



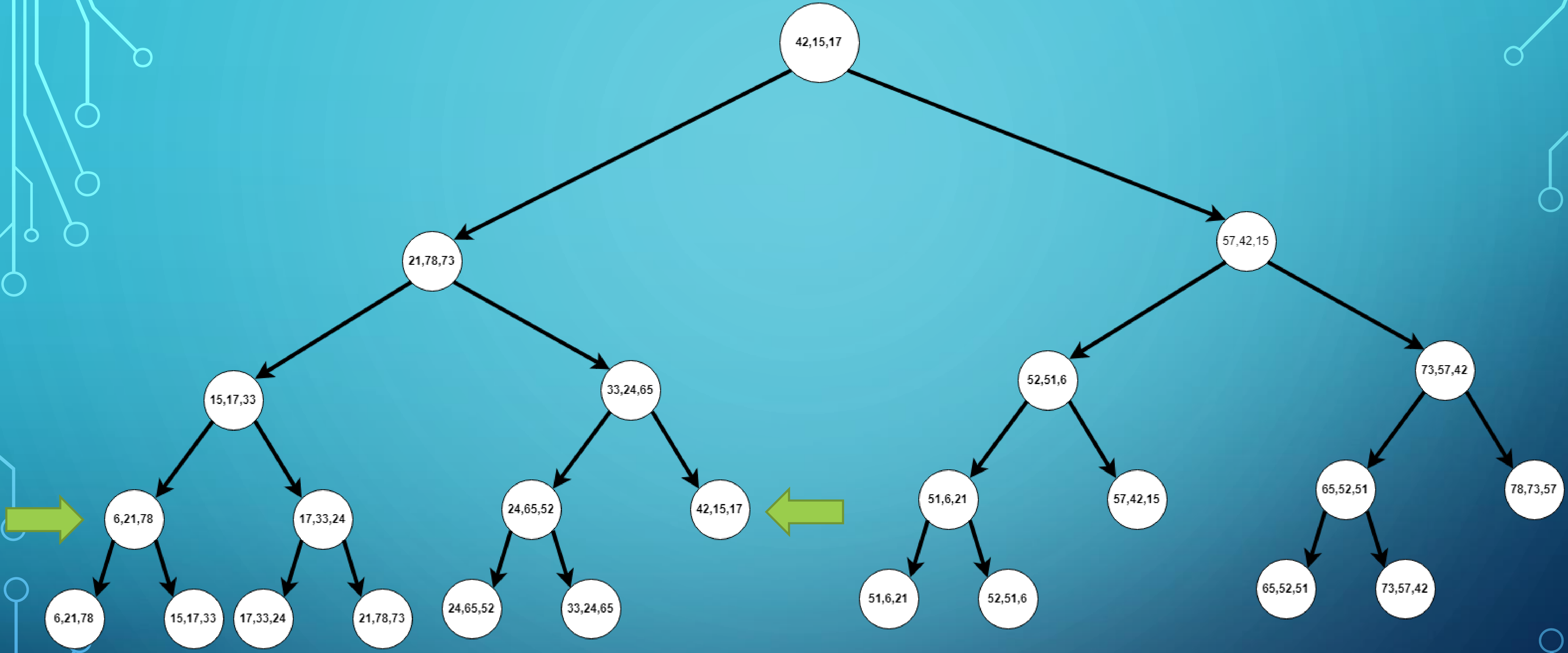
QUERY (**6-42**,15-33,10-30)



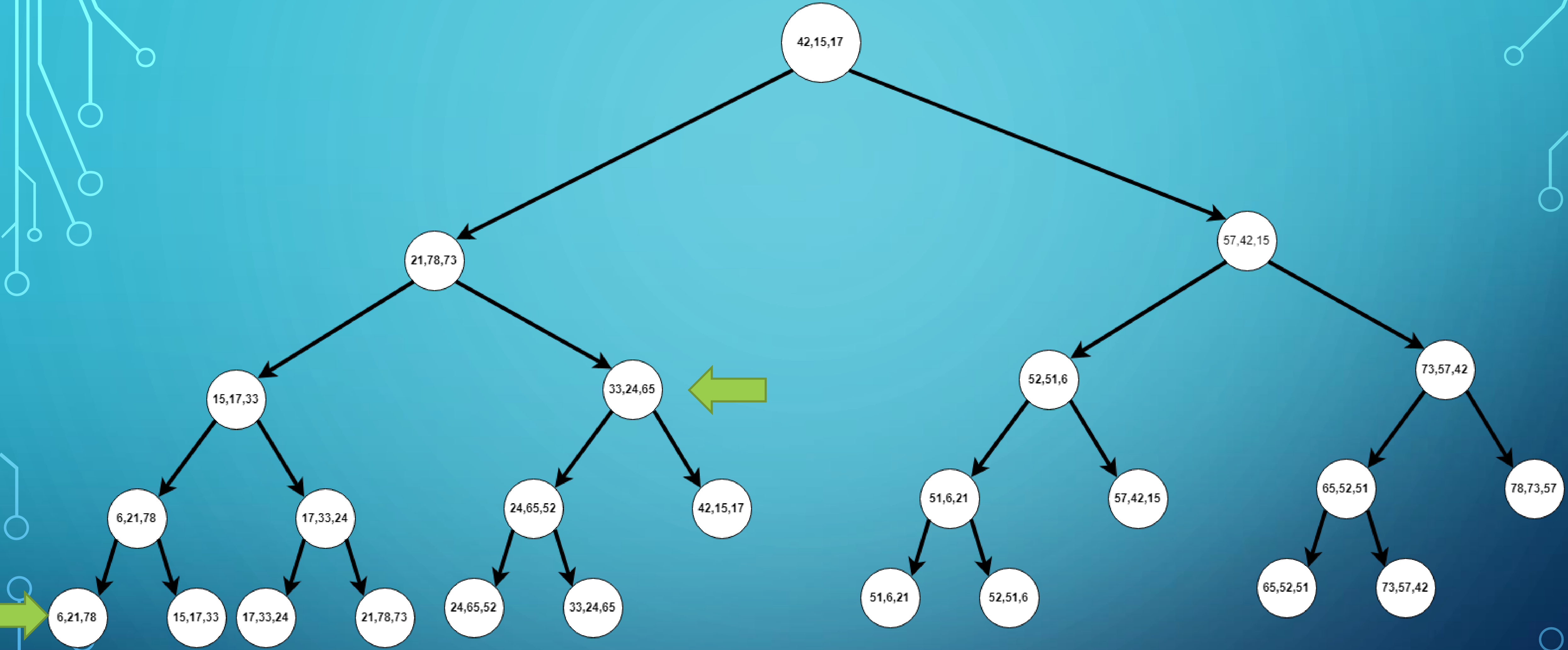
QUERY (6-42,15-33,10-30)



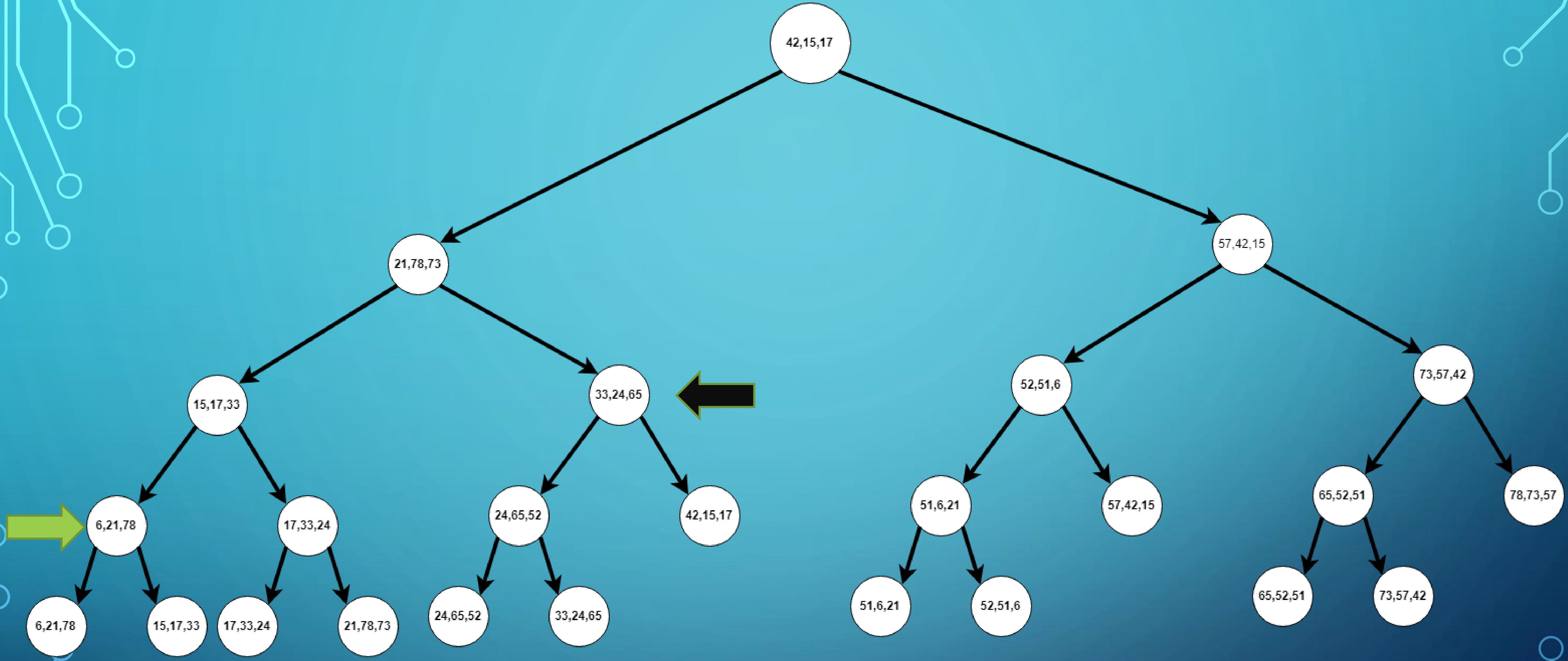
QUERY (6-42,15-33,10-30)



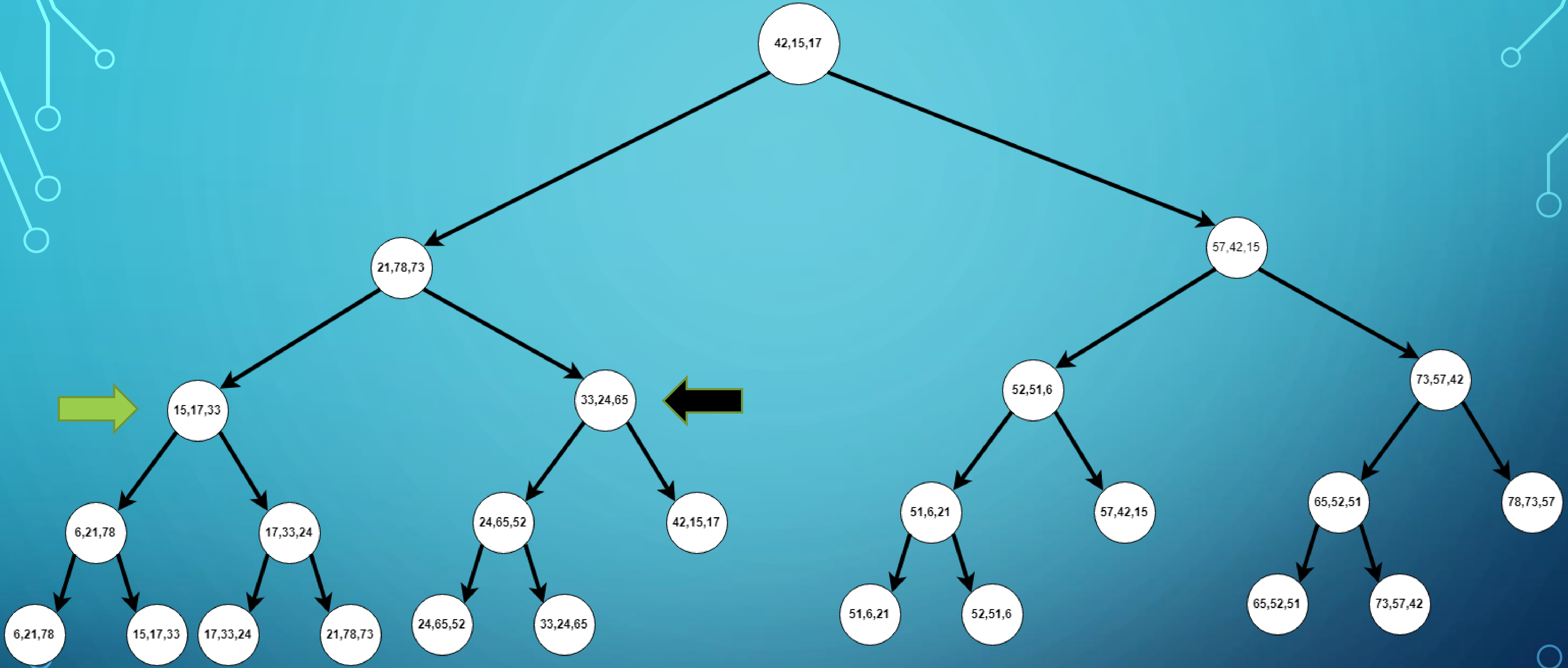
QUERY (6-42,15-33,10-30)



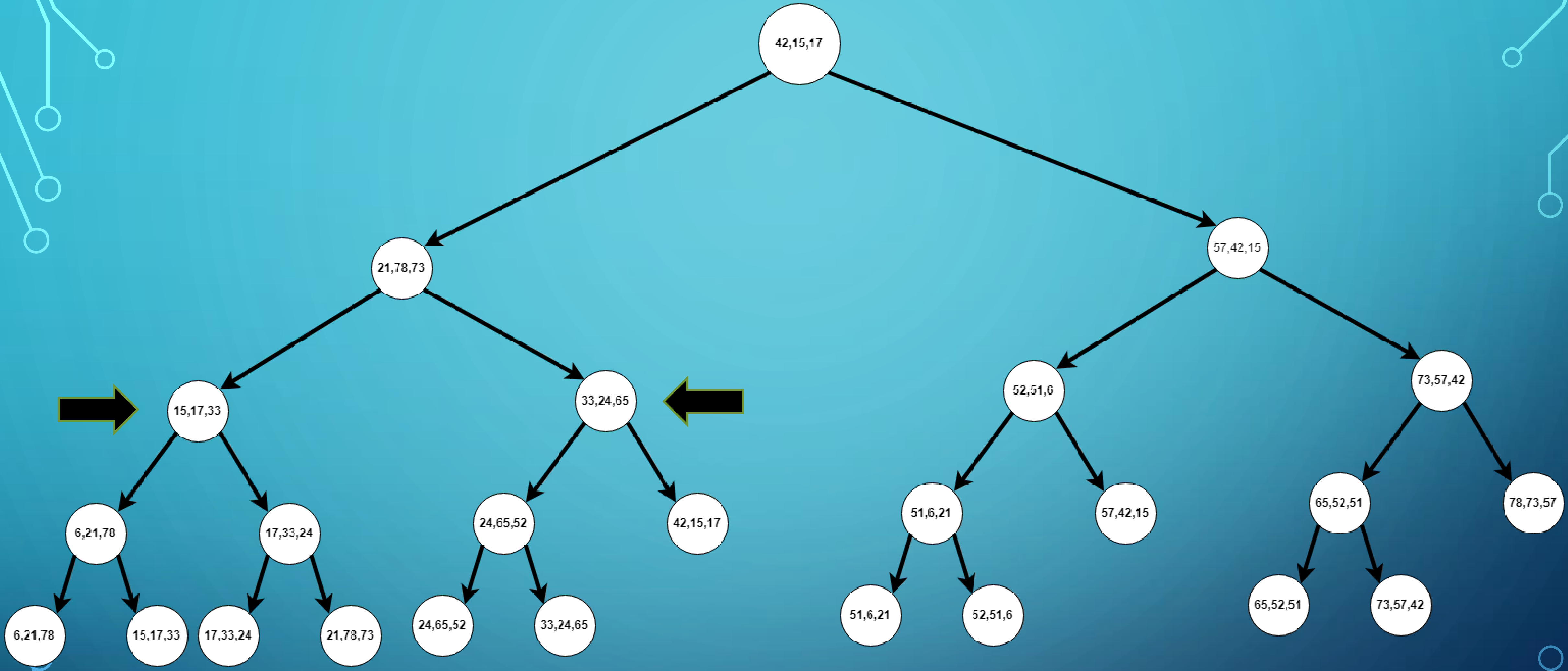
QUERY (6-42,15-33,10-30)



QUERY (6-42,15-33,10-30)



QUERY (6-42,15-33,10-30)

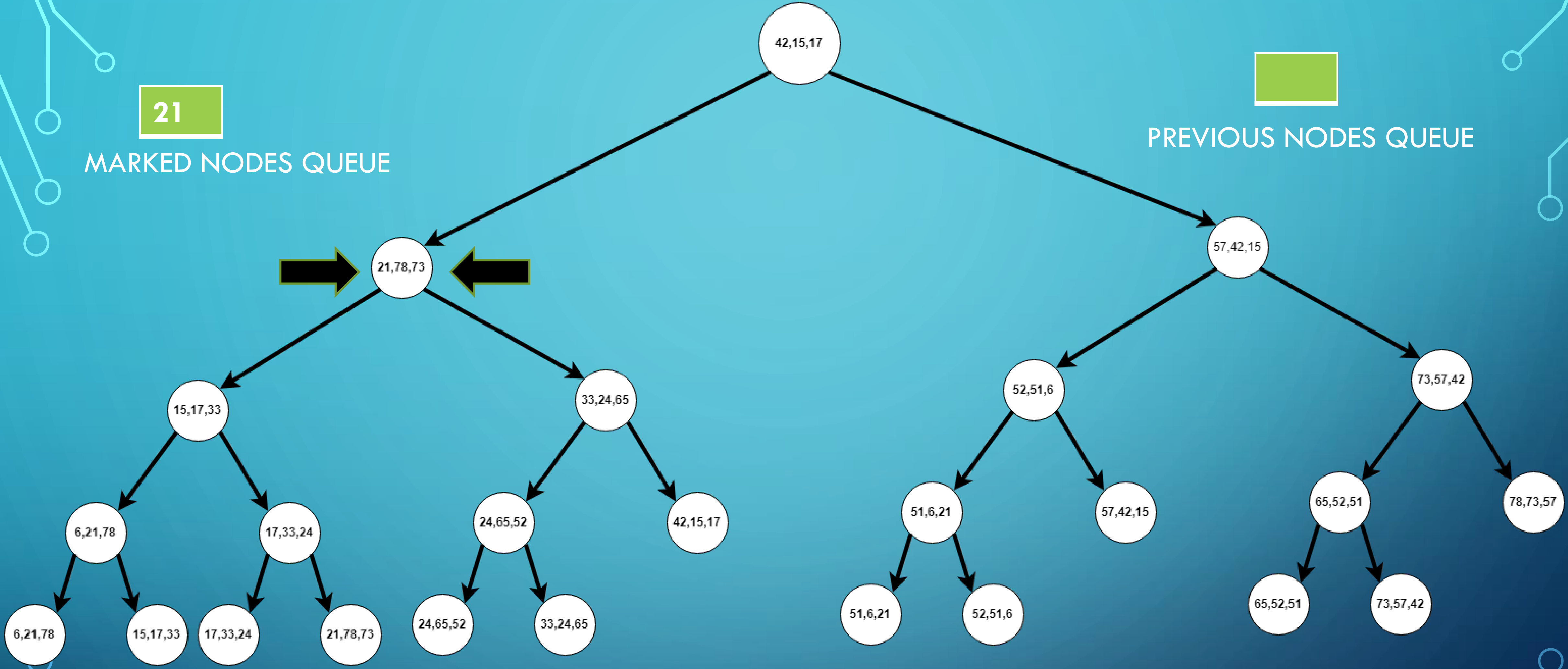


QUERY (6-42,15-33,10-30)

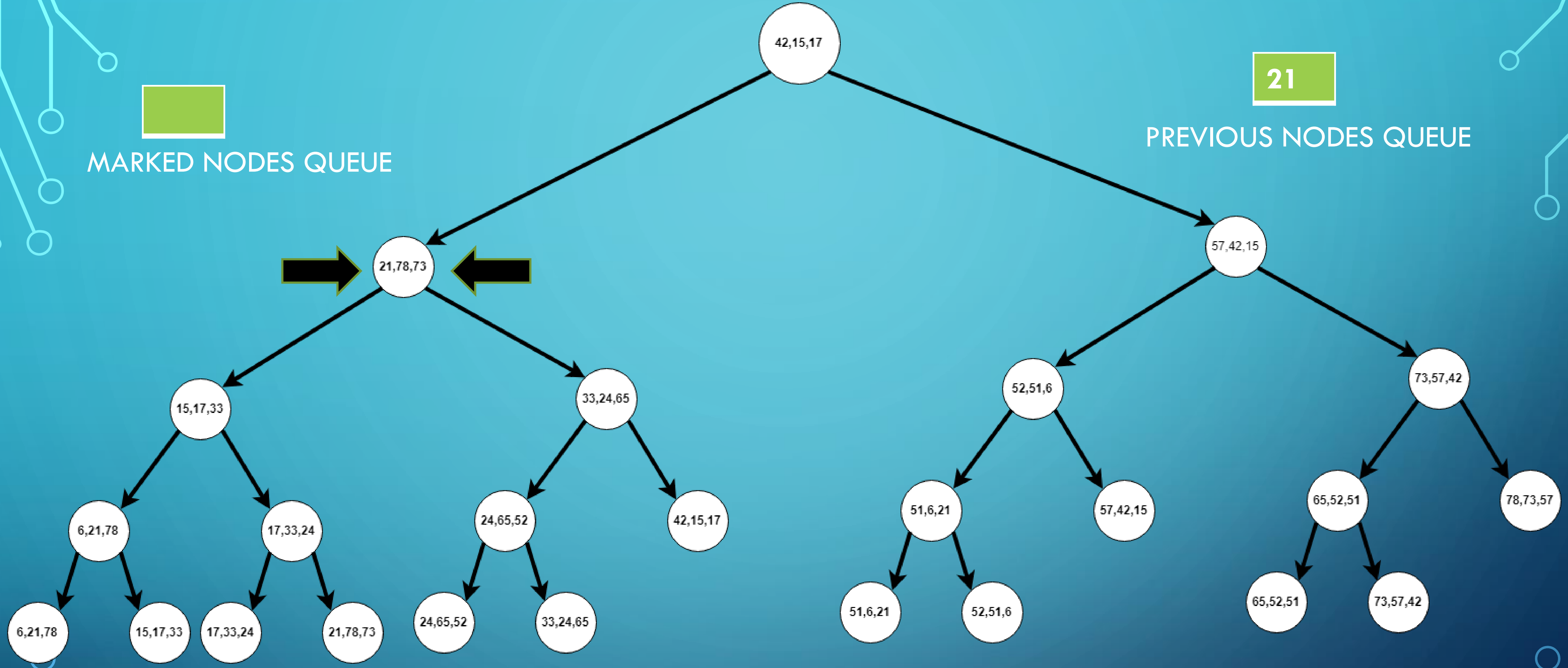
21

MARKED NODES QUEUE

PREVIOUS NODES QUEUE



QUERY (6-42,15-33,10-30)



MARKED NODES QUEUE

21

PREVIOUS NODES QUEUE

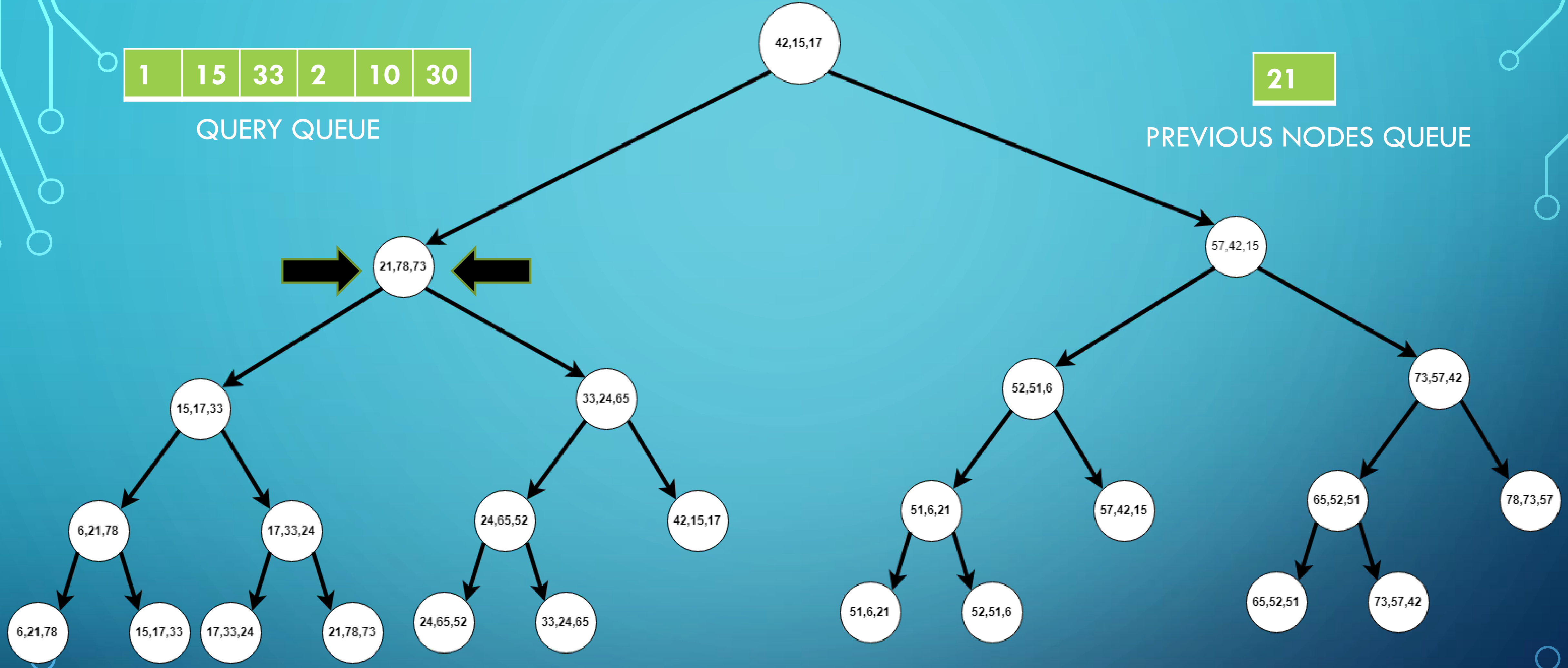
QUERY (6-42,15-33,10-30)

1	15	33	2	10	30

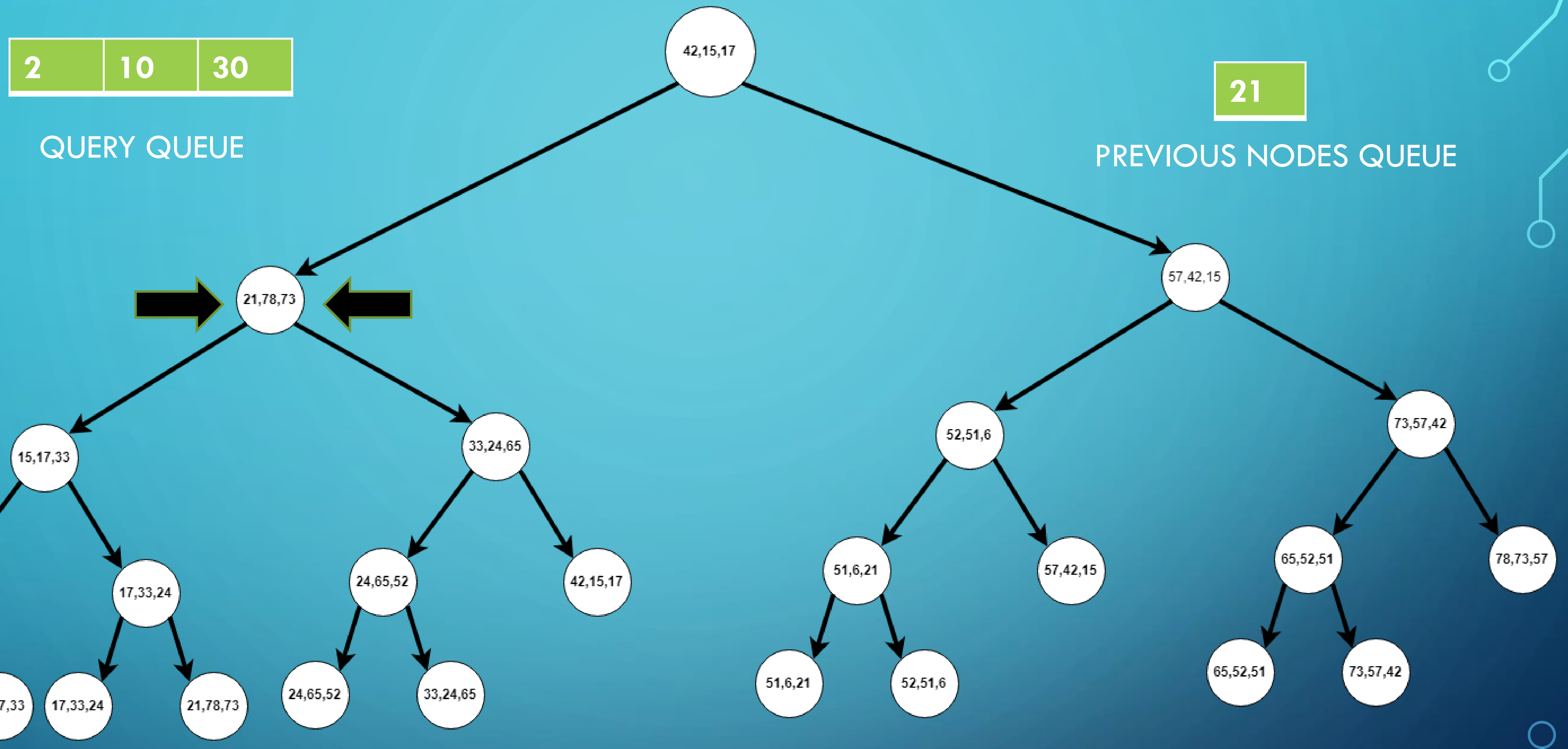
QUERY QUEUE

21

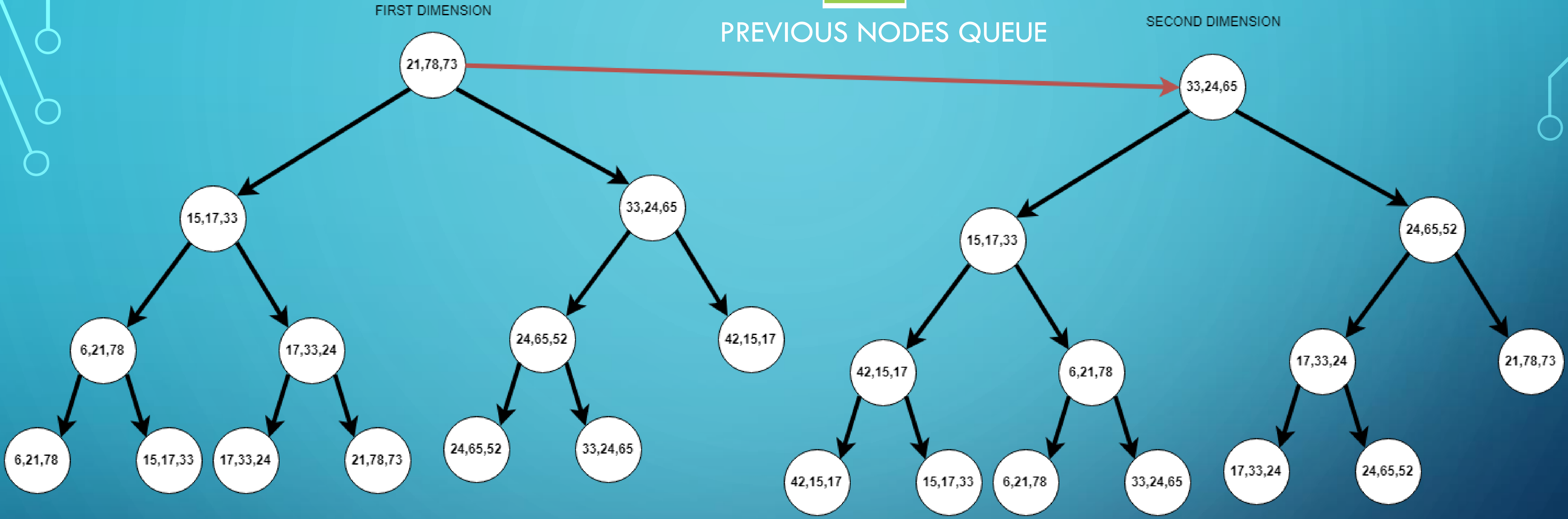
PREVIOUS NODES QUEUE



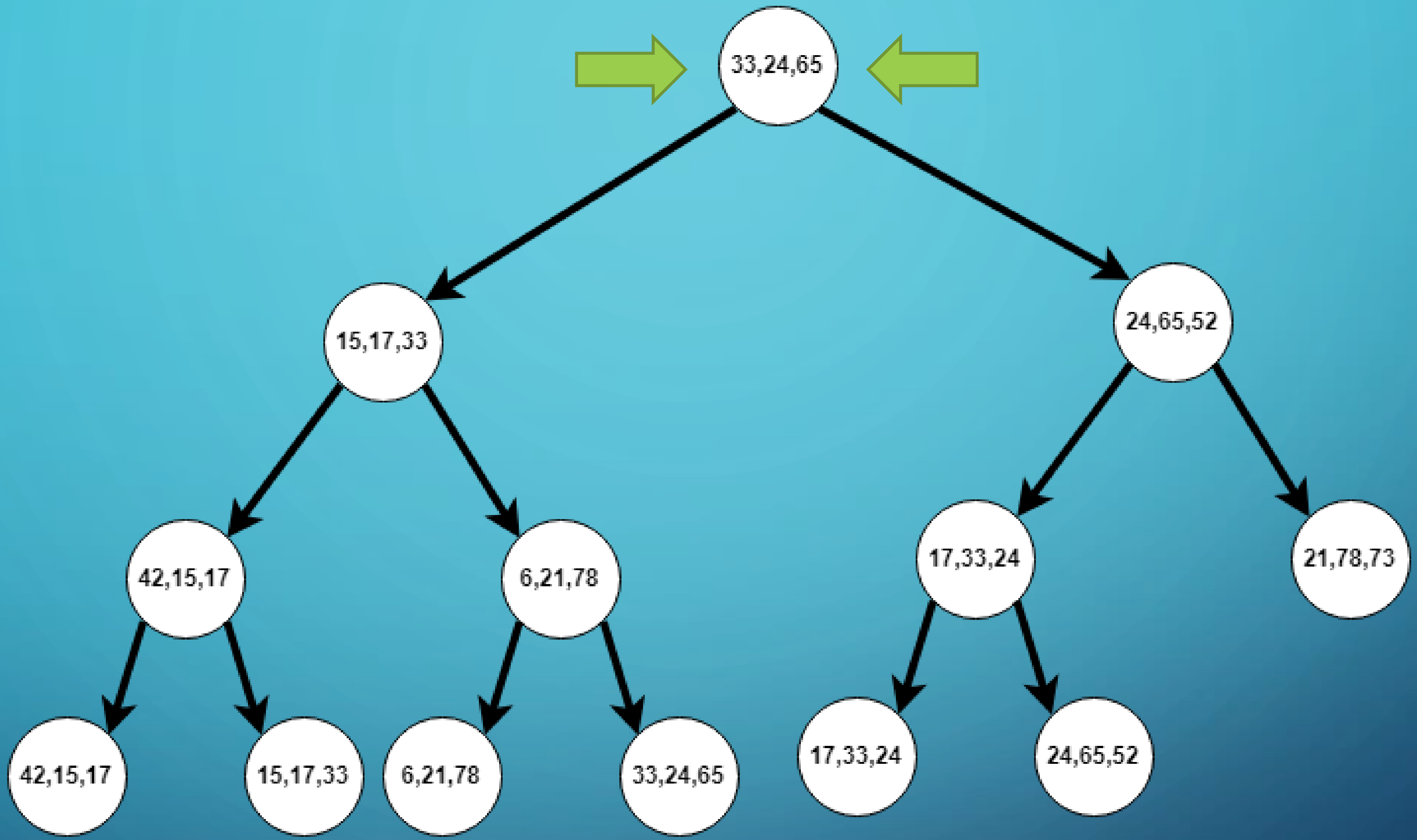
QUERY (6-42,15-33,10-30)



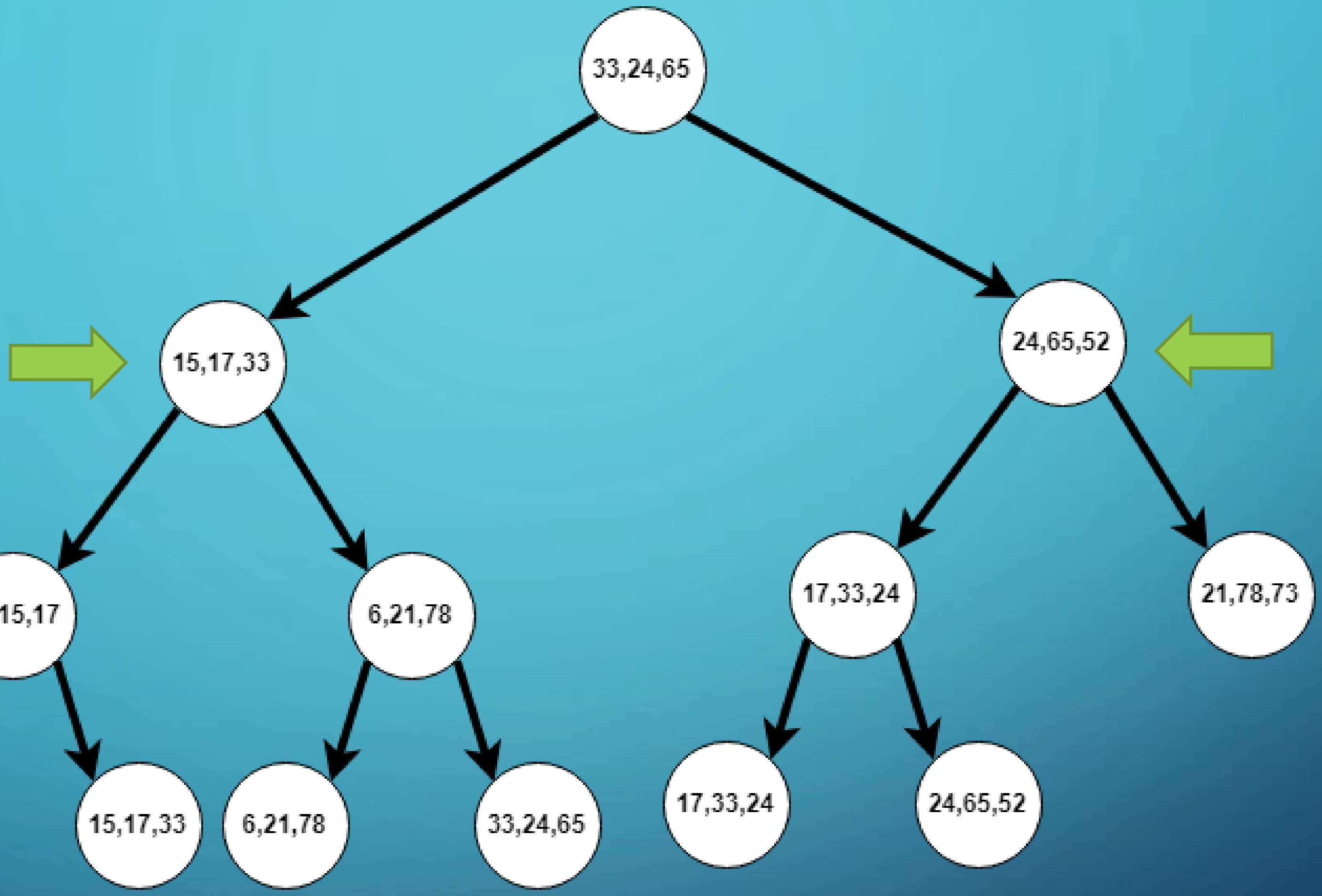
QUERY (6-42,15-33,10-30)



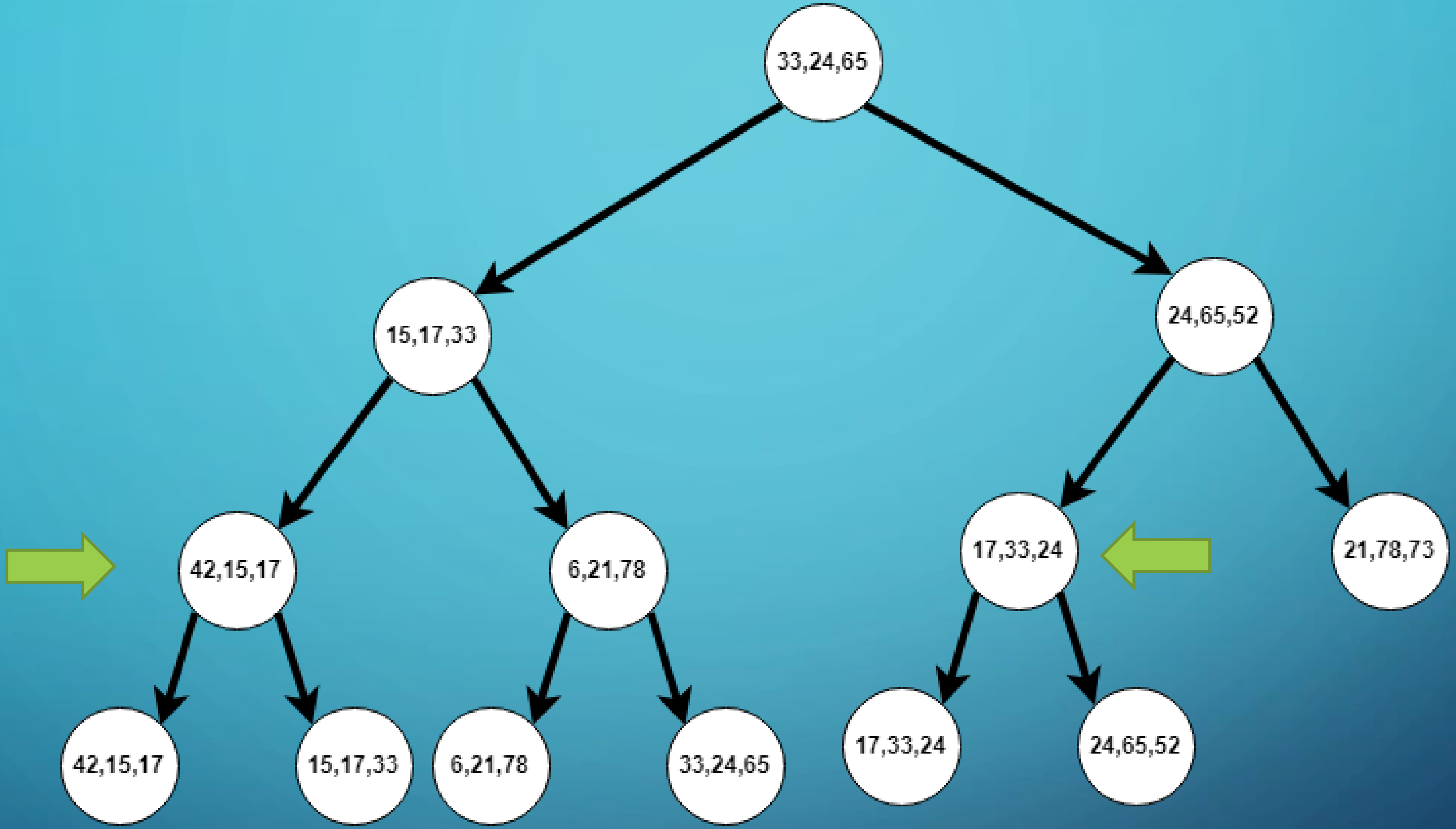
QUERY (6-42,15-33,10-30)



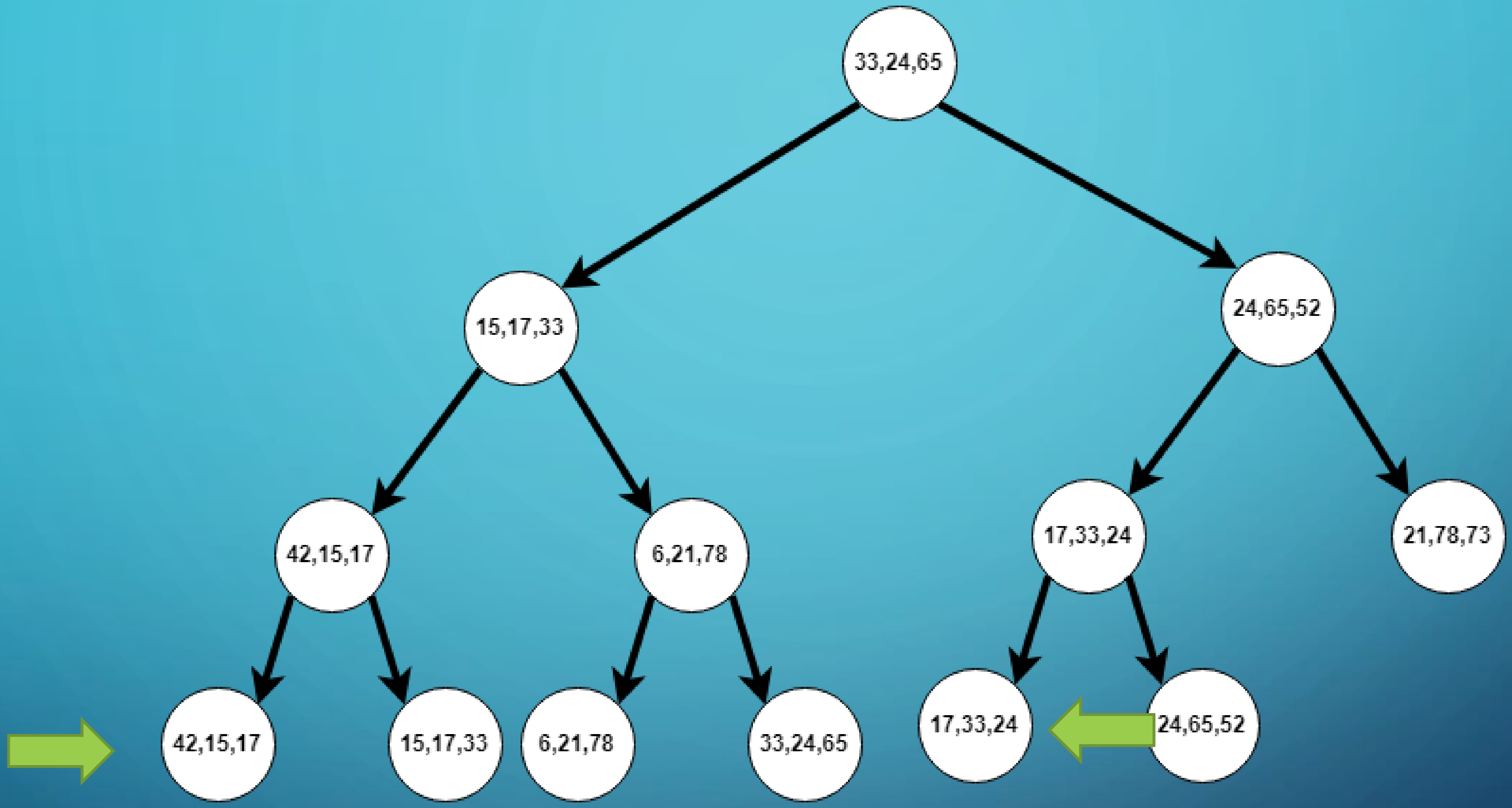
QUERY (6-42,15-33,10-30)



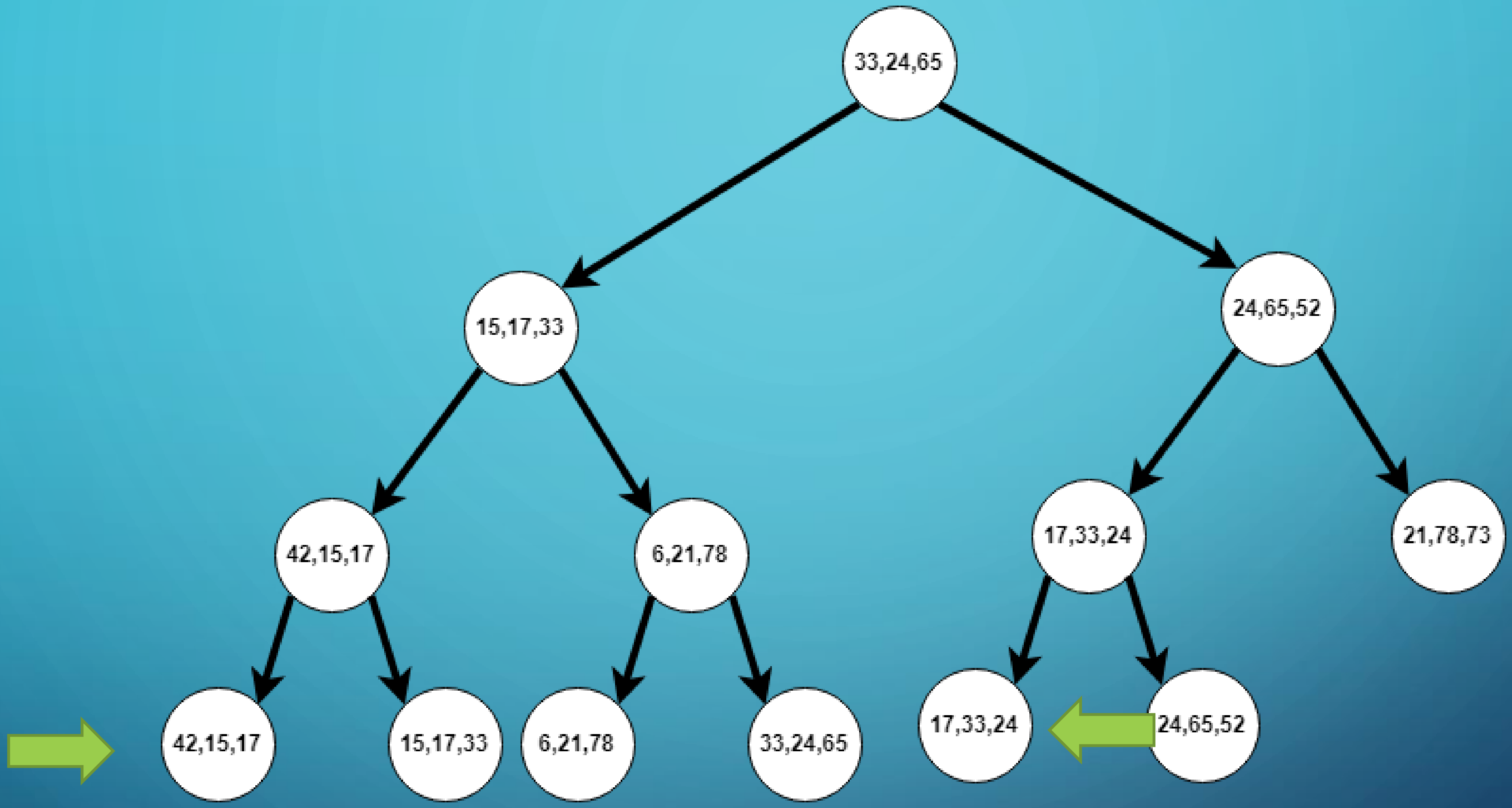
QUERY (6-42,15-33,10-30)



QUERY (6-42,15-33,10-30)



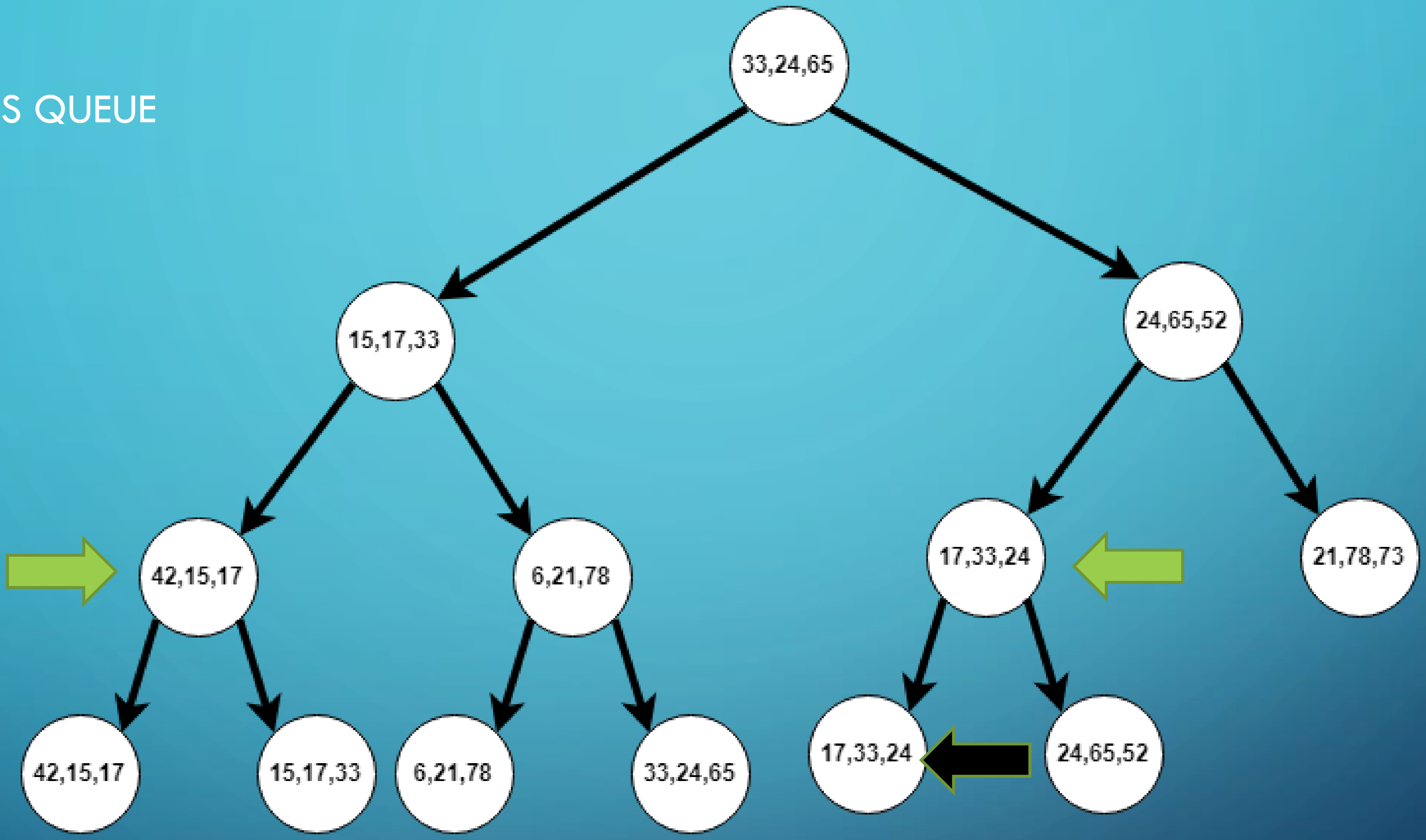
QUERY (6-42,15-33,10-30)



QUERY (6-42,15-33,10-30)

33

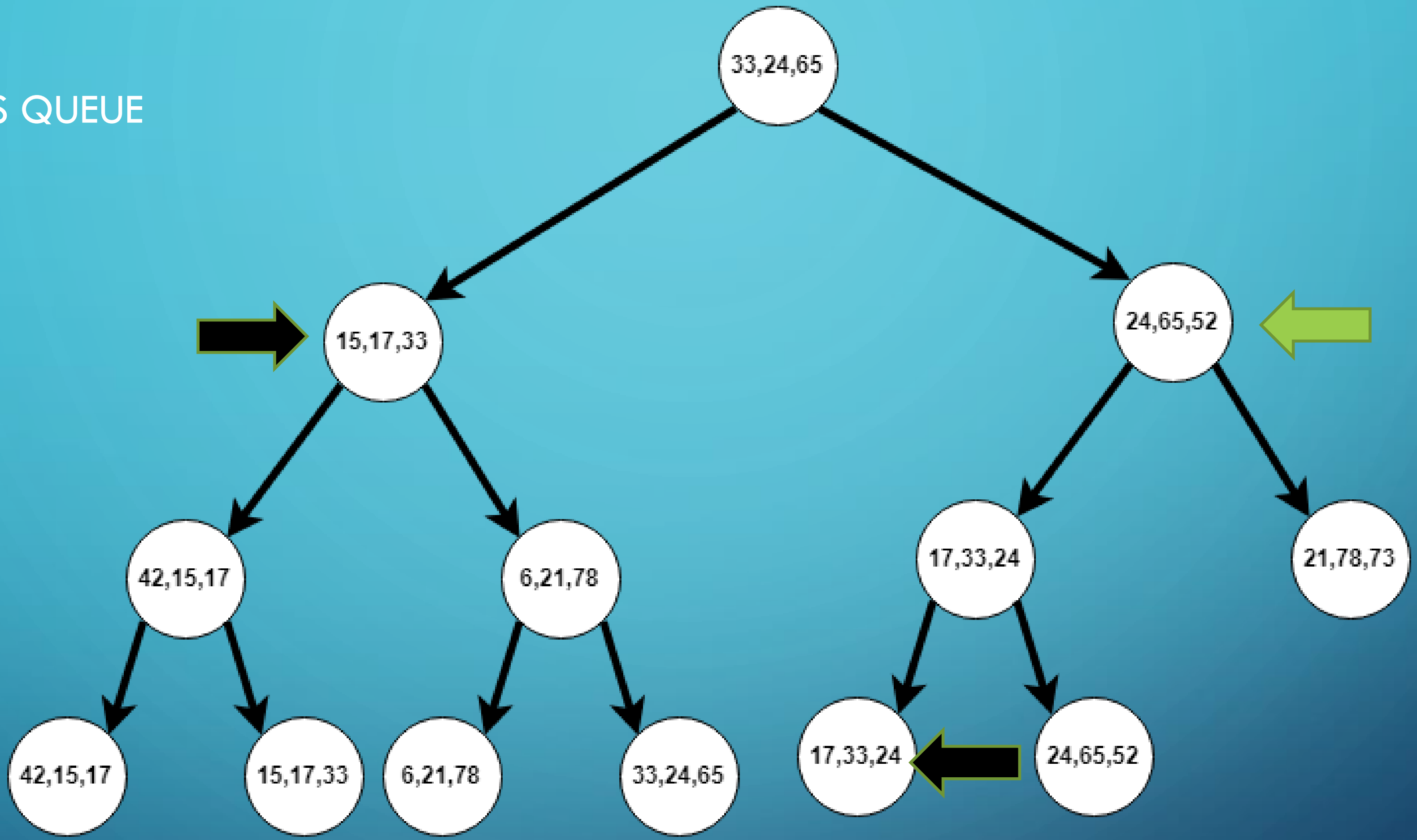
MARKED NODES QUEUE



QUERY (6-42,15-33,10-30)

33	17

MARKED NODES QUEUE



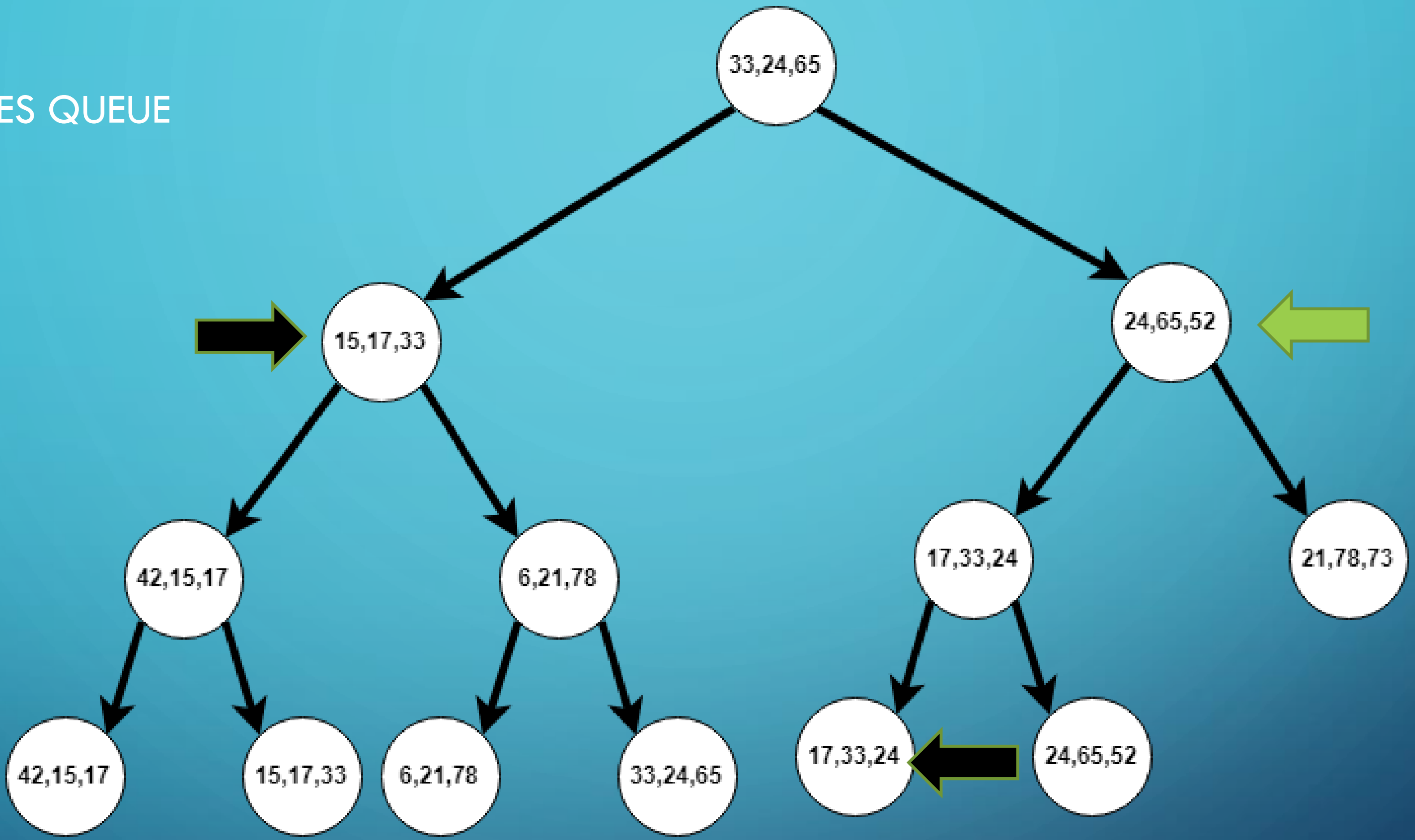
QUERY (6-42,**15-33**,10-30)

33	17

PREVIOUS NODES QUEUE

2	10	30

QUERY QUEUE

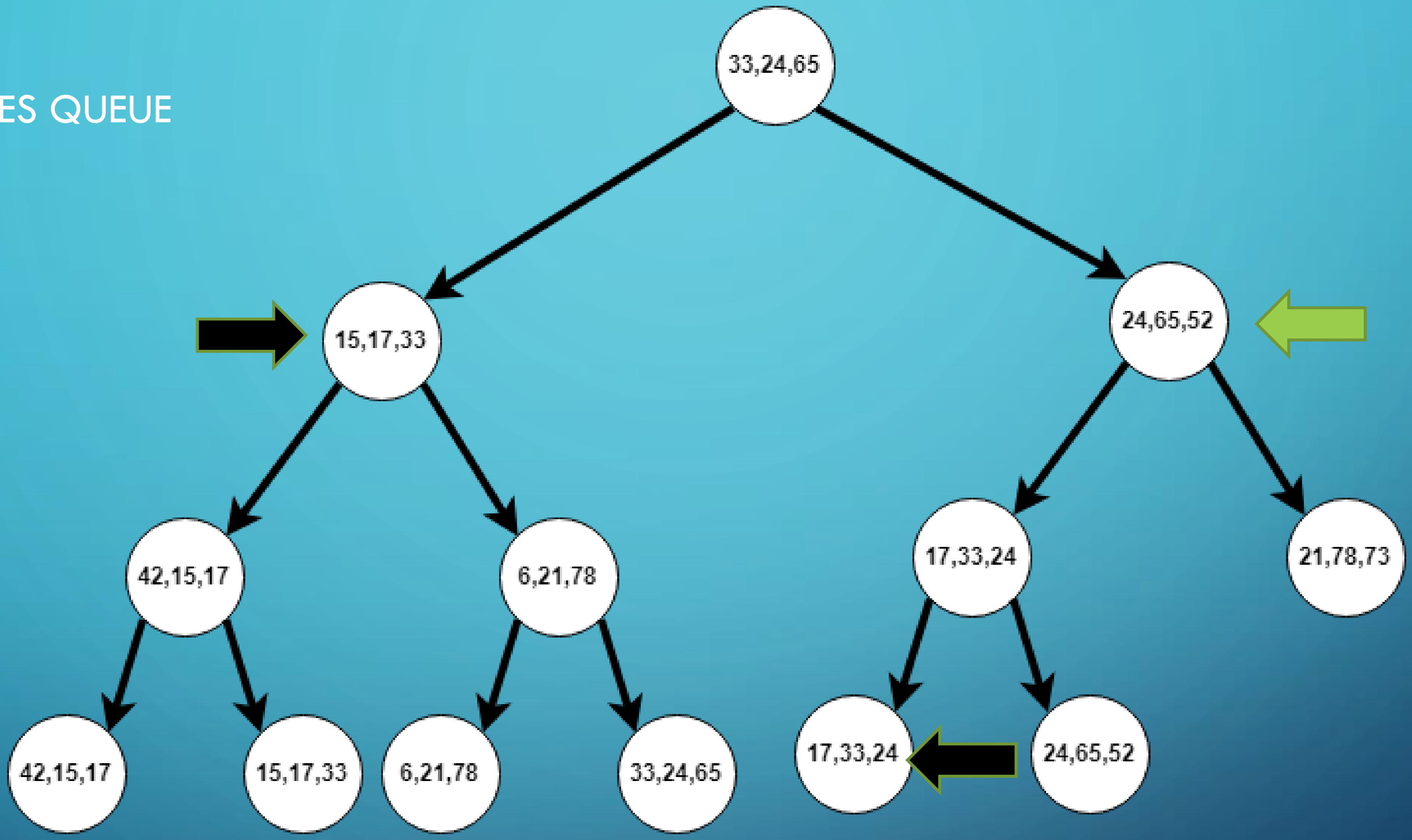


QUERY (6-42,15-33,10-30)

33	17

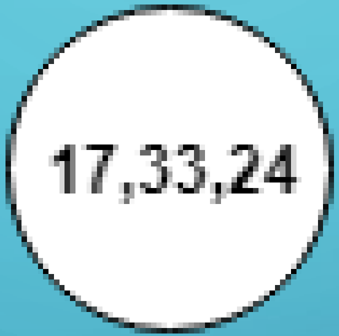
PREVIOUS NODES QUEUE

QUERY QUEUE



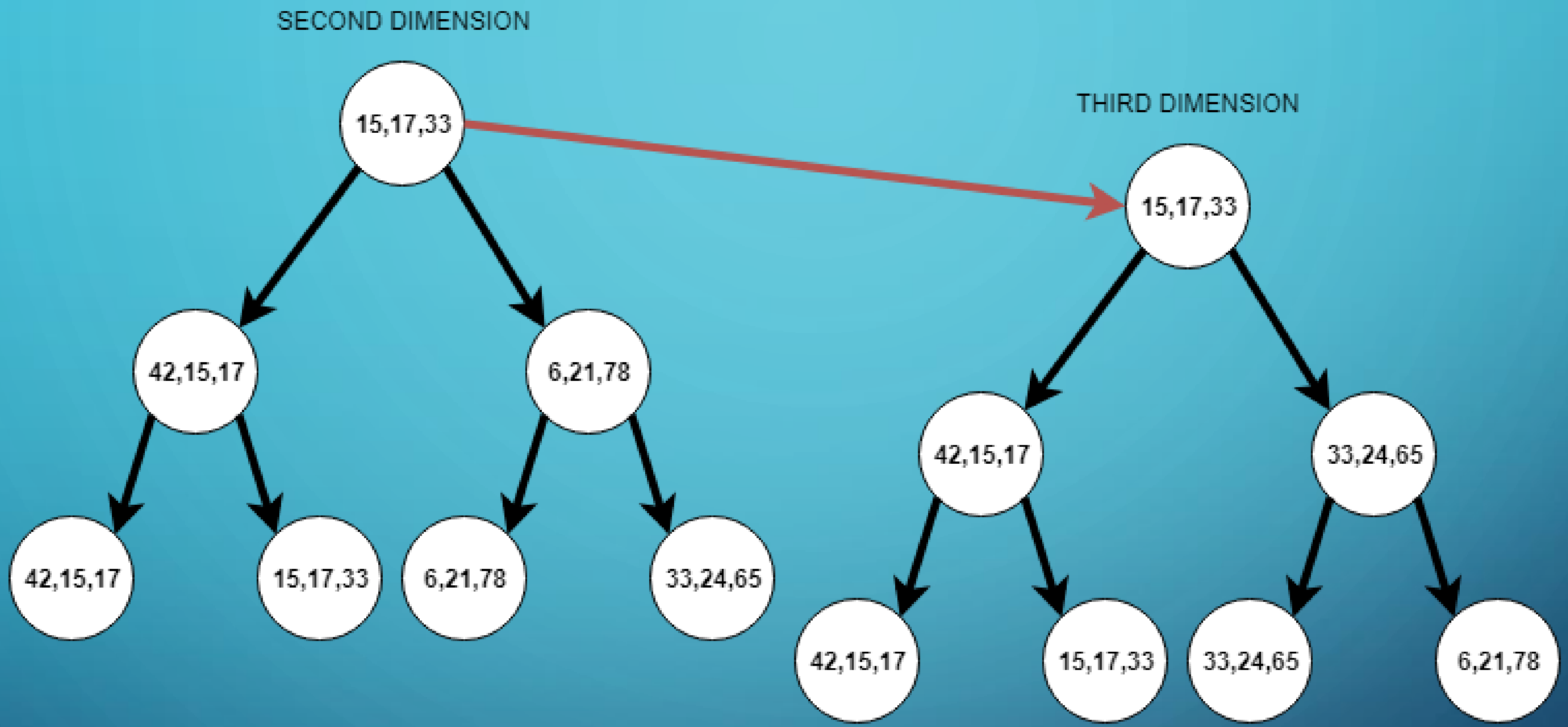
QUERY (6-42,15-33,10-30)

Check leaf with left right10-30 for next dimension

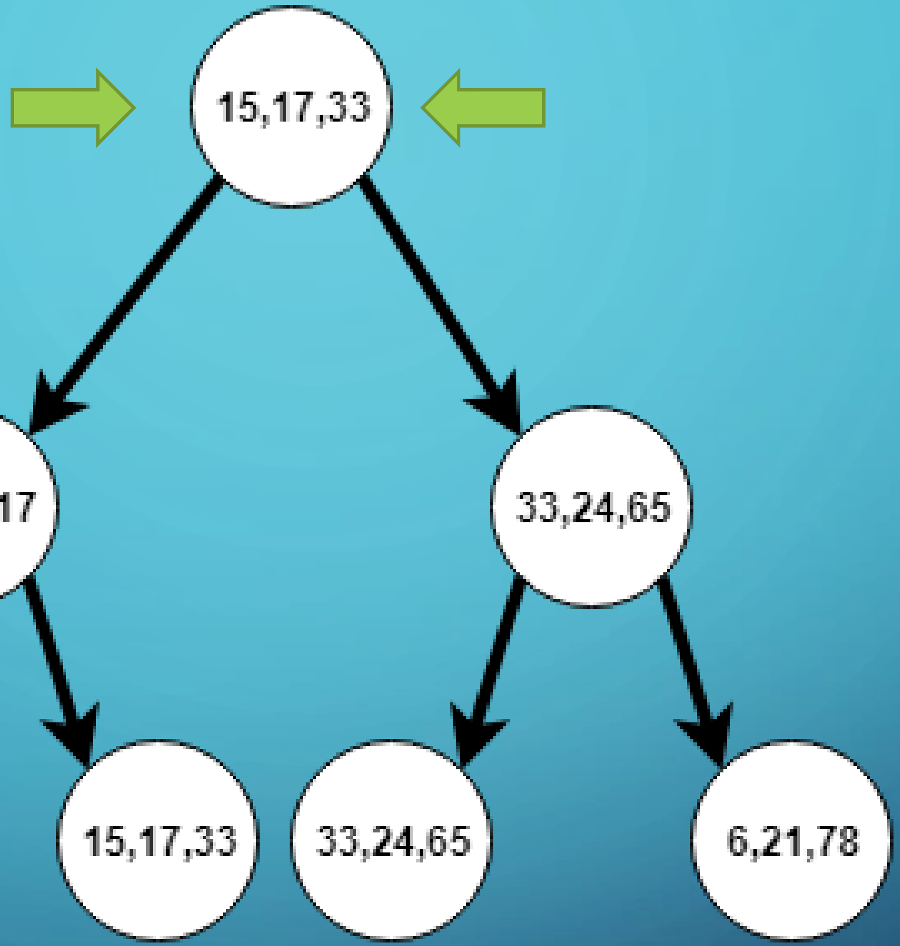


MARKED NODES QUEUE

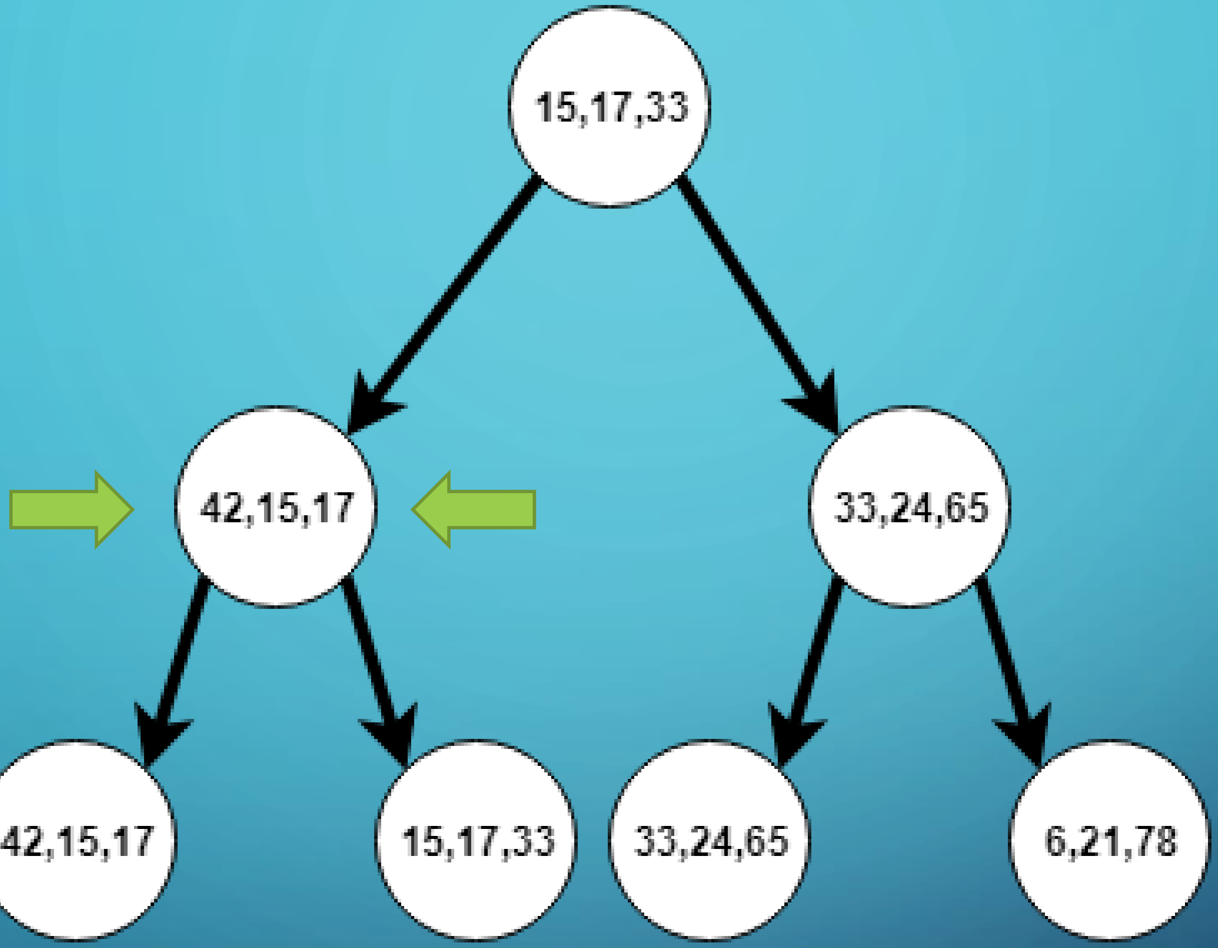
QUERY (6-42,15-33,10-30)



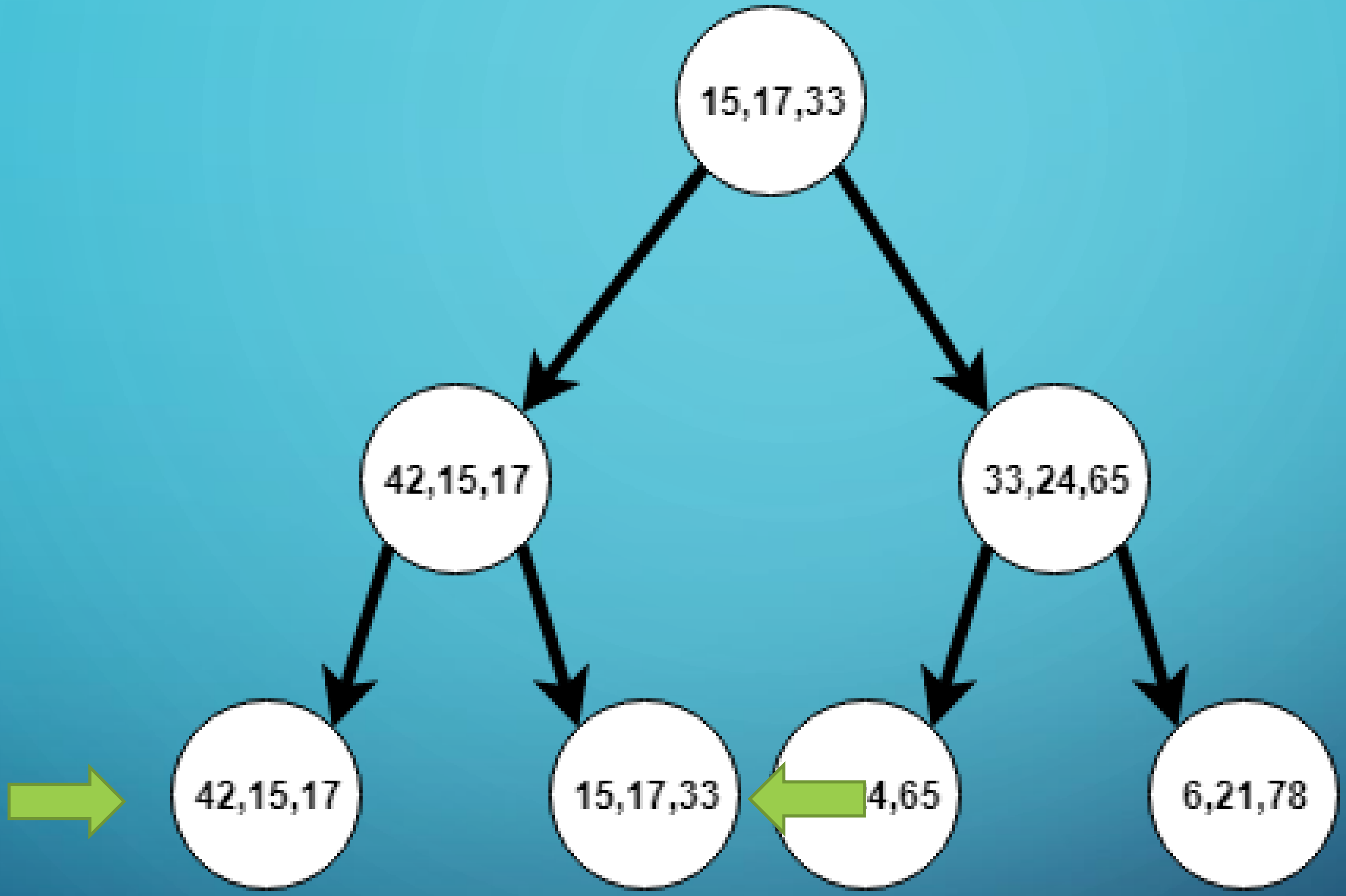
QUERY (6-42,15-33,10-30)



QUERY (6-42,15-33,**10-30**)



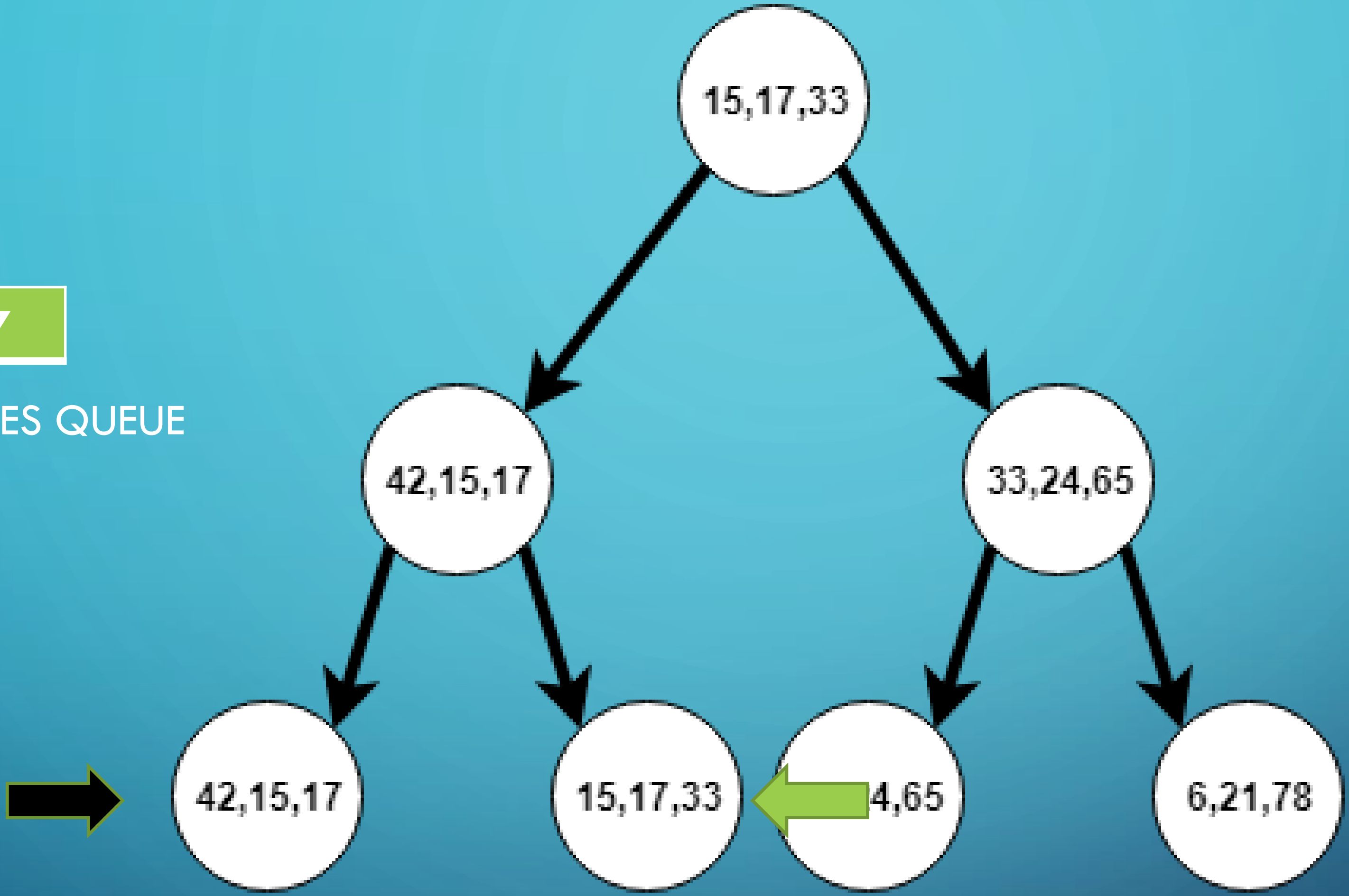
QUERY (6-42,15-33,10-30)



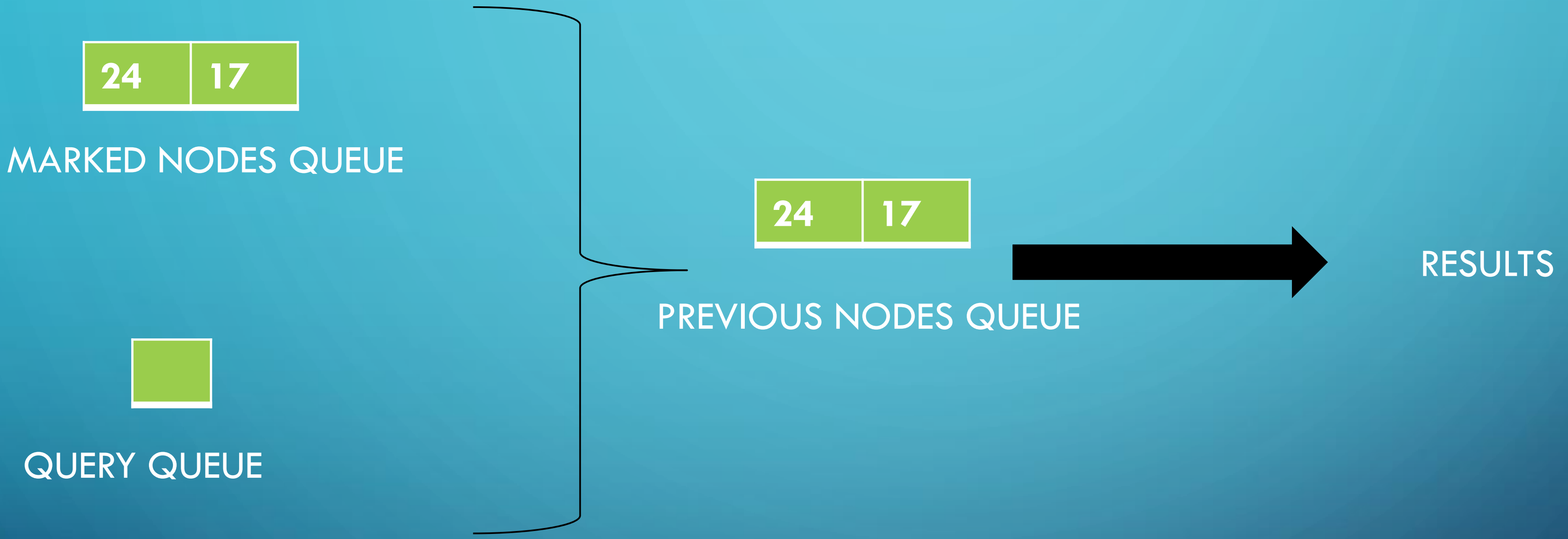
QUERY (6-42,15-33,10-30)

24	17

MARKED NODES QUEUE

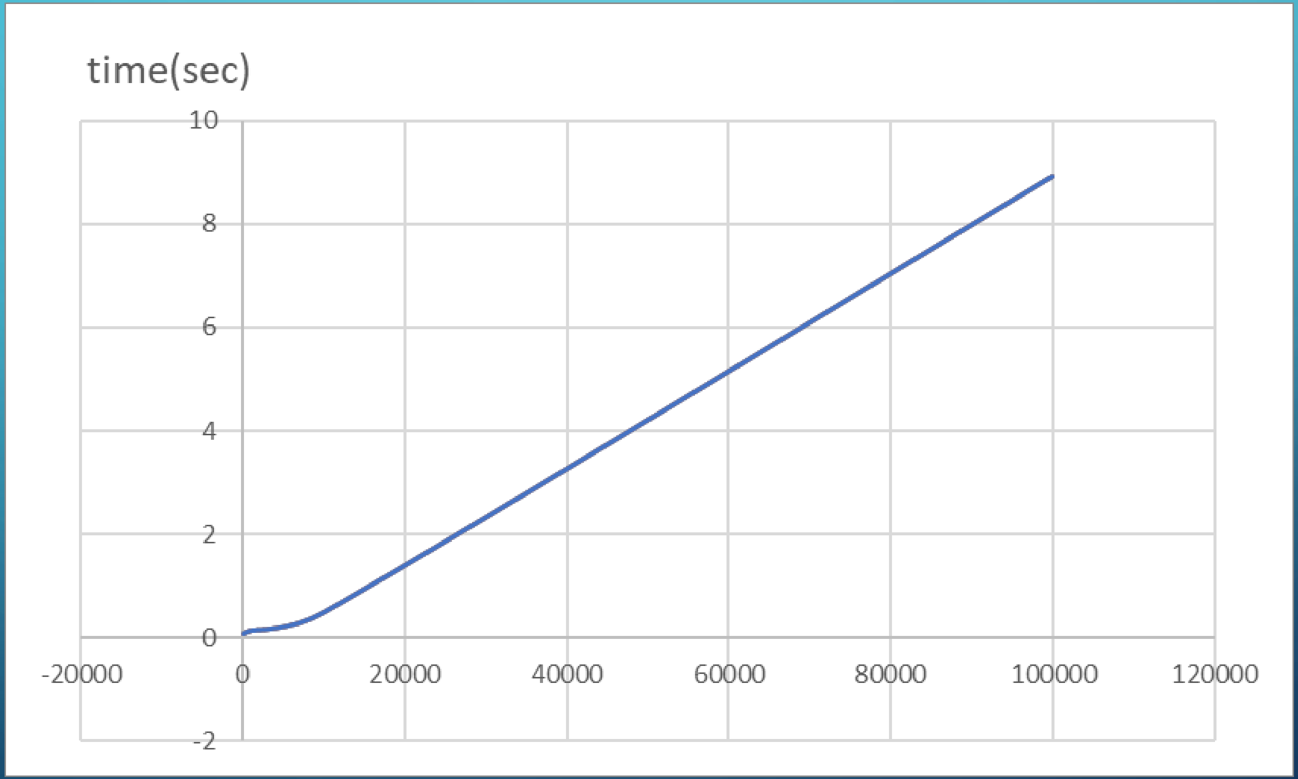


QUERY (6-42,15-33,10-30)



QUERY (6-42,15-33,10-30)

TIME – ITEMS DIAGRAM (3 DIMENSIONS)


time(sec)