

# Balanced Binary Search Trees

# elements

1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

elements

first

last

1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

buildBalancedTree(elements, first, last)

elements

first

m

last

1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

```
buildBalancedTree(elements, first, last)
```

elements

first

m-1

1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

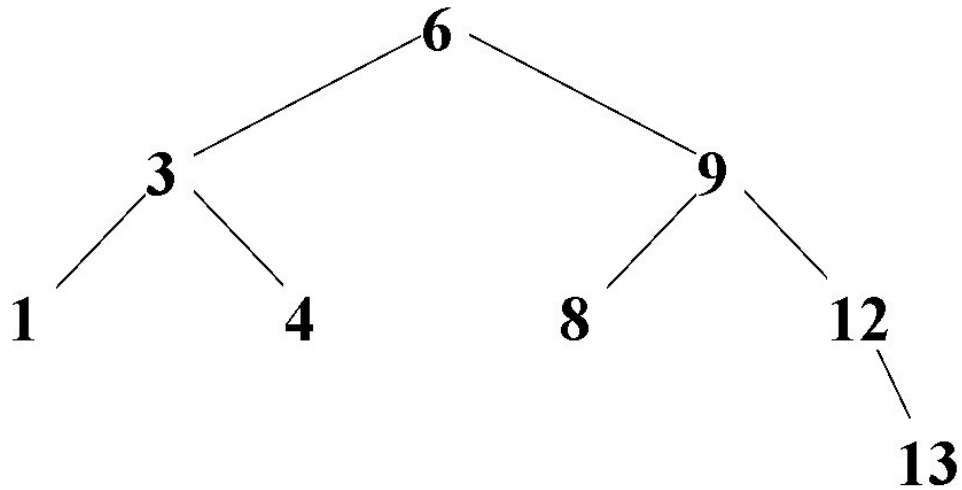
left = buildBalancedTree(elements, first, m-1)

elements

first

m-1

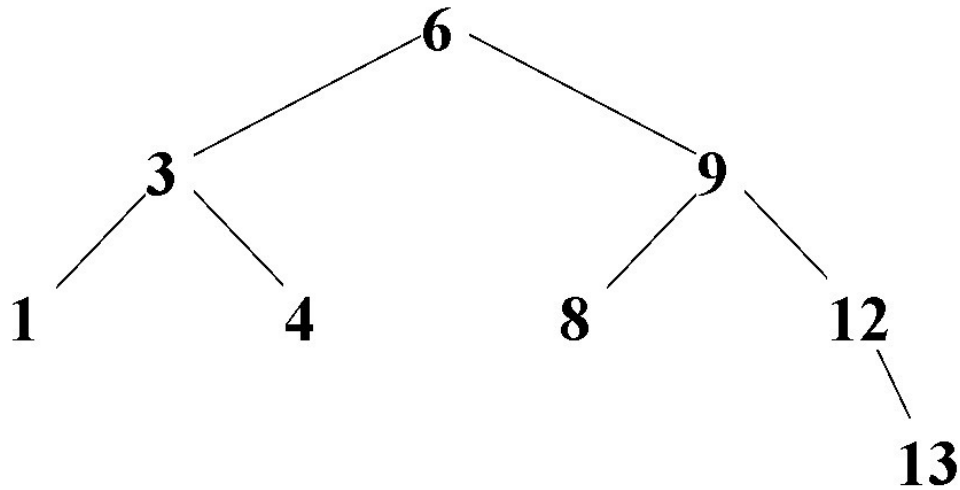
1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



left = buildBalancedTree(elements, first, m-1)

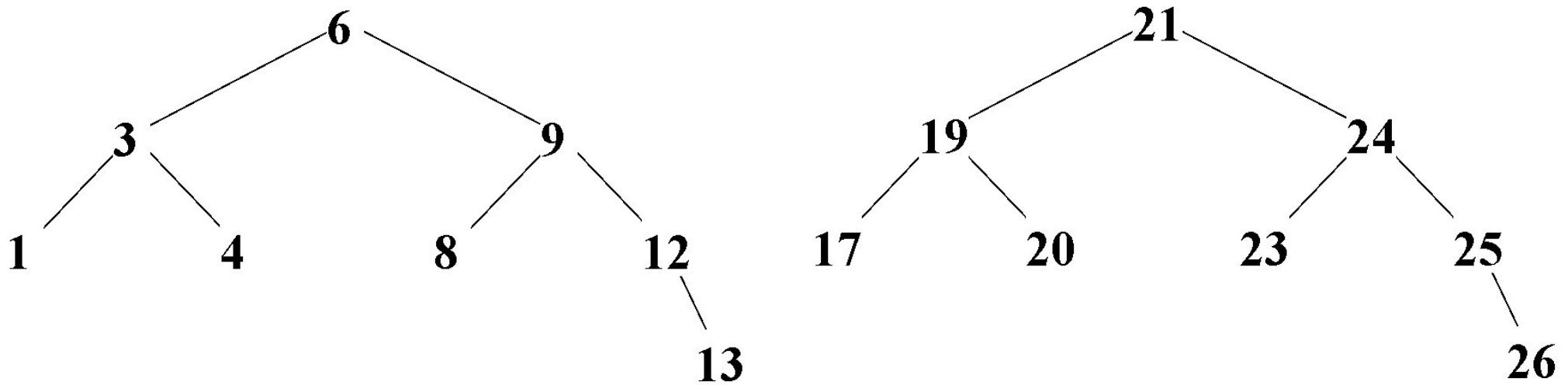
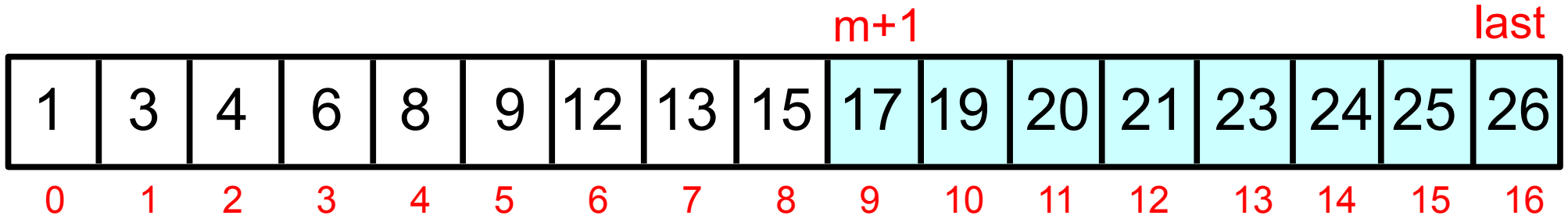
elements

1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



right = buildBalancedTree(elements, m+1, last)

elements



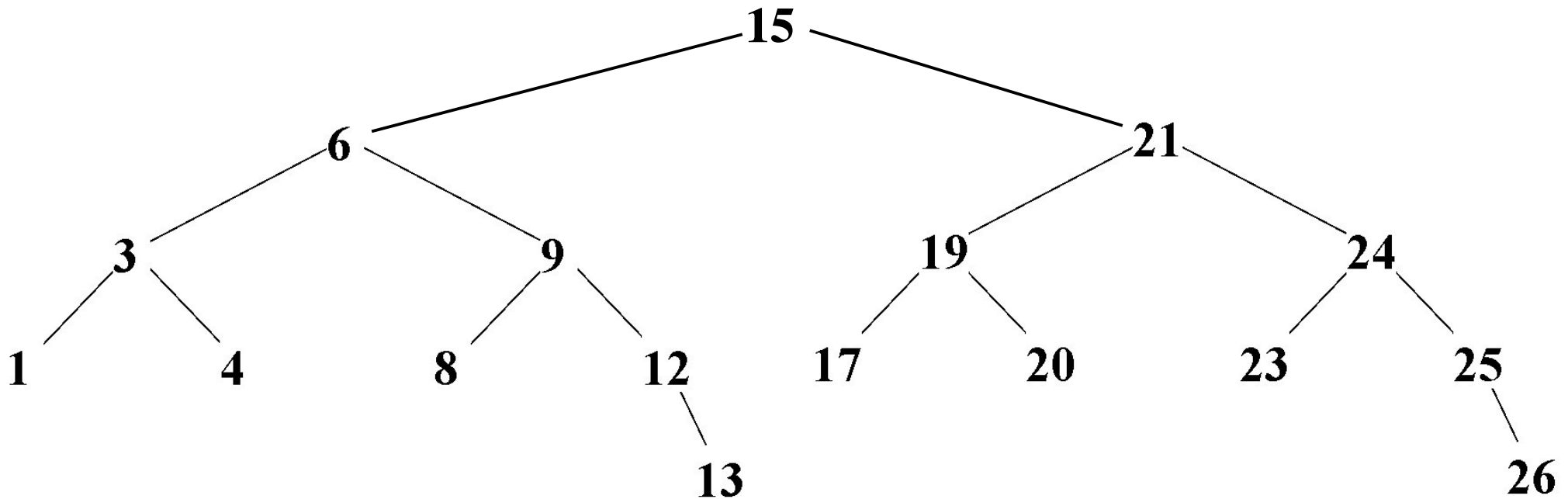
right = buildBalancedTree(elements, m+1, last)



elements

m

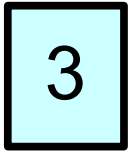
1	3	4	6	8	9	12	13	15	17	19	20	21	23	24	25	26
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16



new BinarySearchTree(**elements[m]**, left, right)

# Base Cases

first  
last

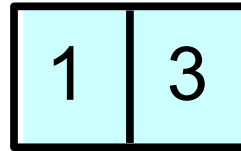


0

3

last == first

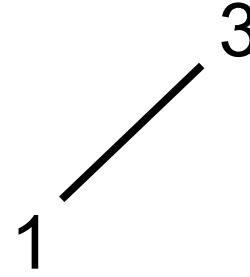
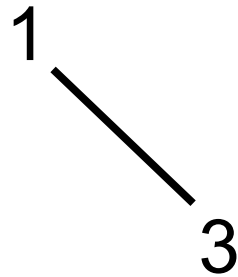
first last



0

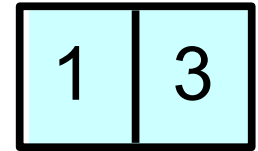
1

or



last == first+1

last first



0

1

*null*

last < first