**Insertion**

Insert at the beginning:

Even

1 2 3 4 5 6

**^**^

0 1 2 3 4 5 6

**^** Offset delta 0 (-0.5)

Odd

1 2 3 4 5 6 7

**^**

0 1 2 3 4 5 6 7

**^**^Offset delta -1 (-0.5)

Insert at the end:

Even

1 2 3 4 5 6

**^**^

1 2 3 4 5 6 7

**^** Offset delta 1 (0.5)

Odd

1 2 3 4 5 6 7

**^**

1 2 3 4 5 6 7 8

**^**^Offset delta 0 (0.5)

Insert median 1:

Even

1 2 3 4 5 6

**^**^

1 2 3 3 4 5 6

**^** Offset delta 1 (0.5 from left, -0.5 from right)

Odd

1 2 3 4 5 6 7

**^**

1 2 3 4 4 5 6 7

**^**^Offset delta 0 (0.5 from left, -0.5 from right)

Insert median 2:

Even

1 2 2 3 3 4 5 6

**^**^

1 2 2 3 3 3 4 5 6

**^** Offset delta 1 (0.5 from left, -0.5 from right)

Odd

0 1 2 2 3 3 4 5 6

**^**

0 1 2 2 3 3 3 4 5 6

**^**^Offset delta 0 (0.5 from left, -0.5 from right)

Insert median 3:

Even

1 2 3 3 4 5 6 7

**^**^

1 2 3 3 3 4 5 6 7

**^** Offset delta 1 (0.5 from left, -0.5 from right)

Odd

0 1 2 3 3 4 5 6 7

**^**

0 1 2 3 3 3 4 5 6 7

**^**^Offset delta 0 (0.5 from left, -0.5 from right)

Looks like we have a hard and fast rule. If inserting a value < median, an even set will see no change, while an odd will see a change of -1. If inserting a value that *will be* >= median, an even set will see a change of 1, while an odd set will see a change of zero. Thus we get the same formula as before:

If the number of elements being inserted (n) is odd, and the original number of elements is even, the offset is -n/2 if the element being inserted is < the current median, and n/2 + 1 if the element being inserted is >= the current median. If the original number of elements is odd, the offset is –n/2 – 1 if the element being inserted is < the current median, and n/2 if the element being inserted is >= the current median.

If the number of elements being inserted is even, or if we’re counting by halves, the offsets are -n/2 and n/2.

**Deletion**

Delete from the beginning:

Even

1 2 3 4 5 6

**^**^

2 3 4 5 6

**^** Offset delta 1 (0.5)

Odd

1 2 3 4 5 6 7

**^**

2 3 4 5 6 7

**^**^Offset delta 0 (0.5)

Delete from the end:

Even

1 2 3 4 5 6

**^**^

1 2 3 4 5

**^** Offset delta 0 (-0.5)

Odd

1 2 3 4 5 6 7

**^**

1 2 3 4 5 6

**^**^Offset delta -1 (-0.5)

Delete median 1:

Even

1 2 3 4 5 6

**^**^

1 2 4 5 6

**^** Offset delta 1 (0.5 – if done before deletion)

We’re not really deleting the median here, we’re deleting *before* the median.

Odd

0 1 2 3 4 5 6

**^**

0 1 2 4 5 6

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete median 2:

Even

1 2 2 3 3 4 5 6

**^**^

1 2 2 3 4 5 6

**^** Offset delta 0 (-0.5 – if done before deletion)

Odd

0 1 2 2 3 3 4 5 6

**^**

0 1 2 2 3 4 5 6

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete median 3:

Even

1 2 3 3 4 5 6 7

**^**^

1 2 3 4 5 6 7

**^** Offset delta 1 (0.5 – if done before deletion)

Odd

0 1 2 3 3 4 5 6 7

**^**

0 1 2 3 4 5 6 7

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete median 4:

Even

1 3 3 3 4 5 6 7

**^**^

1 3 3 4 5 6 7

**^** Offset delta 1 (0.5 – if done before deletion)

Odd

0 1 3 3 3 4 5 6 7

**^**

0 1 3 3 4 5 6 7

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete median 6:

Even

1 2 3 3 3 4 5 6

**^**^

1 2 3 3 4 5 6

**^** Offset delta 0 (-0.5 – if done before deletion)

Odd

0 1 2 3 3 3 4 5 6

**^**

0 1 2 3 3 4 5 6

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete median 7:

Even

1 2 2 3 3 3 4 5

**^**^

1 2 2 3 3 4 5

**^** Offset delta 0 (-0.5 – if done before deletion)

Odd

0 1 2 2 3 3 3 4 5

**^**

0 1 2 2 3 3 4 5

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete median 8:

Even

2 3 3 3 3 4 5 6

**^**^

2 3 3 3 4 5 6

**^** Offset delta 0 (-0.5 – if done before deletion)

Odd

1 2 3 3 3 3 4 5 6

**^**

1 2 3 3 3 4 5 6

**^**^Offset delta -1 (-0.5 – if done before deletion)

Delete second median 1:

Even

1 2 3 4 5 6

**^**

1 2 3 5 6

**^** Offset delta 0

Delete second median 2:

Even

1 2 3 4 4 6

**^**

1 2 3 4 6

**^** Offset delta 0

Insertion algorithm for **one** value (v) using nodes and offsets:

If medianNode is the median node, medianNode.value the median value, and medianNode.valueCount is the count of that value, and medianOffset is the median offset, then, AFTER the value has been inserted, if totalCount is the new total number of values:

if totalCount % 2 == 0 {

if val < medianNode.value {

medianOffset -= 1

if medianOffset < 0 {

medianNode = medianNode.prev

medianOffset = medianNode.valueCount – 1

}

}

} else {

if val >= medianNode.value {

medianOffset += 1

if medianOffset >= medianNode.count {

medianNode = medianNode.next

medianOffset = 0

}

}

}

Deletion algorithm for **one** value (v) using nodes and offsets:

BEFORE the value is inserted, if totalCount is the existing total number of values:

if totalCount % 2 == 0 {

if val < medianNode.value ||

val == medianNode.value && medianOffset == medianNode.valueCount – 1 {

medianOffset += 1

if medianOffset >= medianNode.valueCount {

medianNode = medianNode.next

medianOffset = 0

}

}

} else {

if val >= medianNode.value {

medianOffset -= 1

if medianOffset < 0 {

medianNode = medianNode.prev

medianOffset = medianNode.valueCount – 1

}

}

}