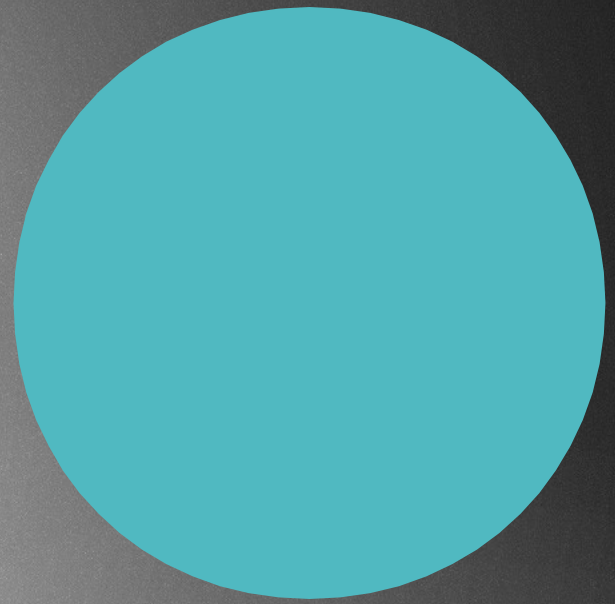
SORTING ALGORITHMS



SHELL SORT

Shell sort

- Another quadratic running time sorting algorithm
- It is the generalization of the insertion sort
- Insertion sort → sometimes we have to make lots of shifts / swaps
- This feature is not so good → thats why shell sort came to be as an enhanced insertion sort
- The method starts by sorting pairs of elements far apart from each other
- Then progressively reducing the gap between elements to be compared
- Starting with far apart elements can move some out-of-place elements into position faster than a simple nearest neighbor exchange

Shell sort

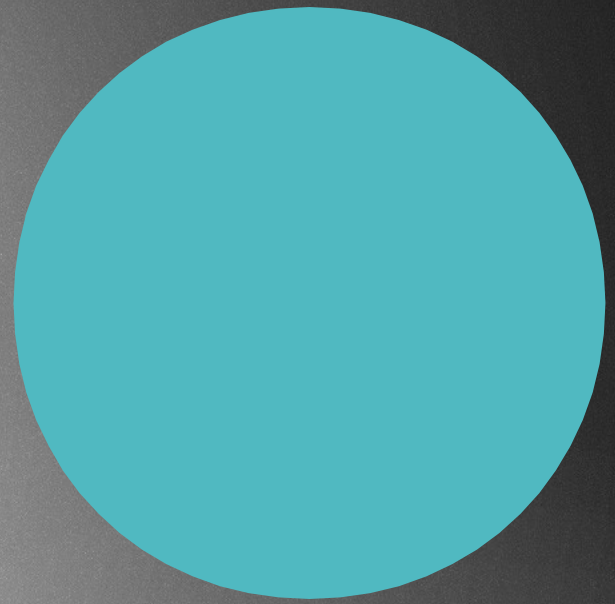
- Shell sort is heavily dependent on the gap sequence it uses
- Consider every **h**-th element in the array
- Such a subarray is said to be **h**-sorted
- We use insertion sort as a subprocedure → the only difference is that we start sorting items far away from each other
- This rearrangement allows elements to move long distances in the original list → reducing large amounts of disorder quickly

Shell sort

- Unstable → it changes the relative order of elements with equal value
- Because it relies heavily on insertion sort → it is also an adaptive algorithm so runs faster on partially sorted input
- Not so popular algorithm nowadays !!!

Shell sort

```
shellSort(array)

    for every gap in gaps

        for i=gap to n
            temp = array[i]

            for j=i and j >= gap and array[j-gap] > temp
                array[j] = array[j-gap]
                j = j – gap
            end

            a[j] = temp
        end
    end
```



Shell sort

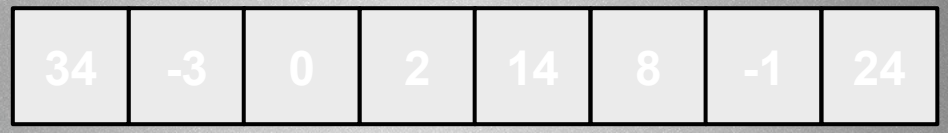
```
shellSort(array)

    for every gap in gaps

        for i=gap to n
            temp = array[i]

            for j=i and j >= gap and array[j-gap] > temp
                array[j] = array[j-gap]
                j = j – gap
            end

            a[j] = temp
        end
    end
```

We have to define some formula for choosing the gaps: for example at the beginning we consider every **3rd** item … after that we consider every **2nd** item

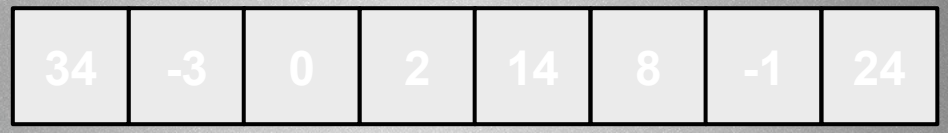
Shell sort

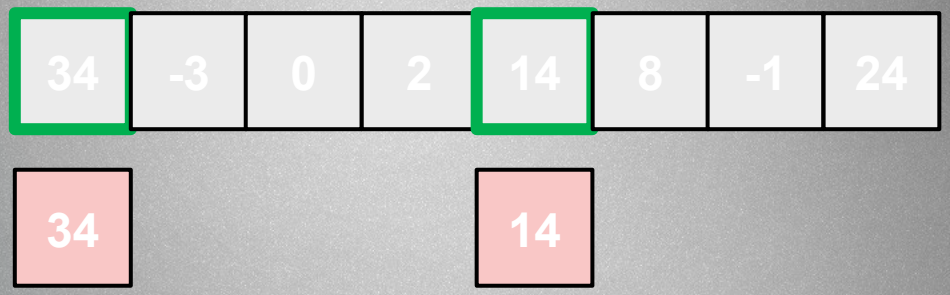
```
shellSort(array)

    for every gap in gaps

        for i=gap to n
            temp = array[i]

            for j=i and j >= gap and array[j-gap] > temp
                array[j] = array[j-gap]
                j = j – gap
            end

            a[j] = temp
        end
    end
```

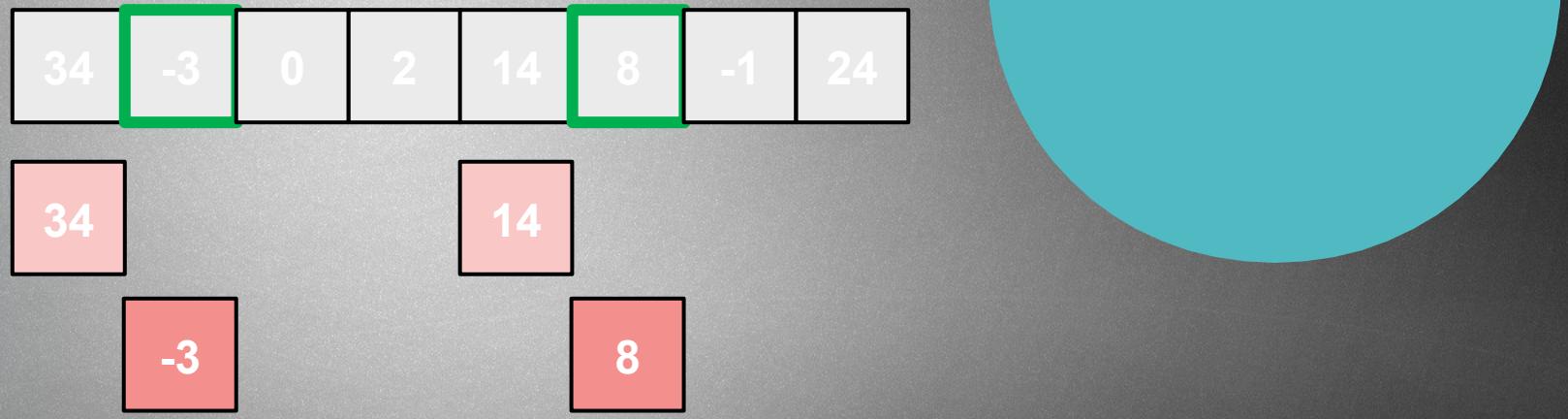
Basically we do a simple insertion sort on the subarray

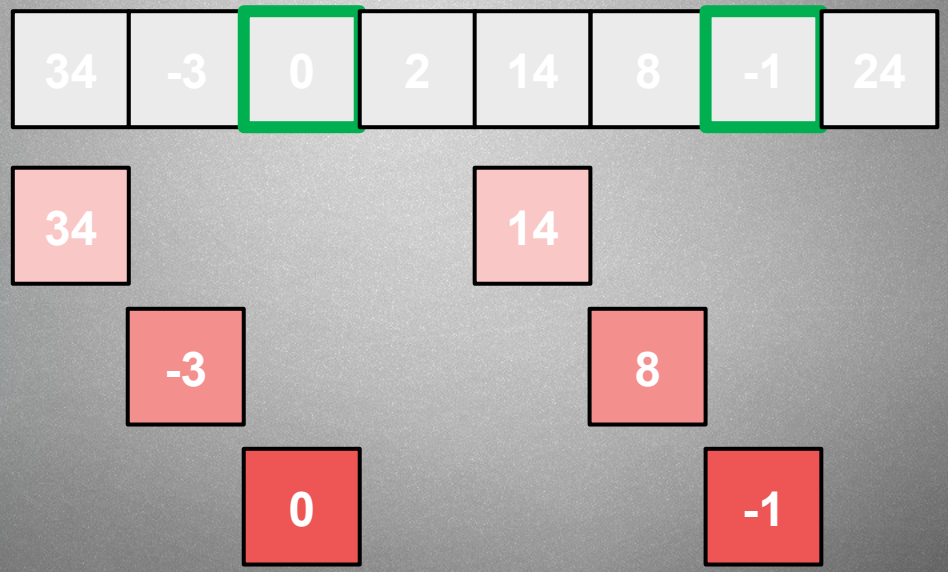

34
-3
0
2
14
8
-1
24

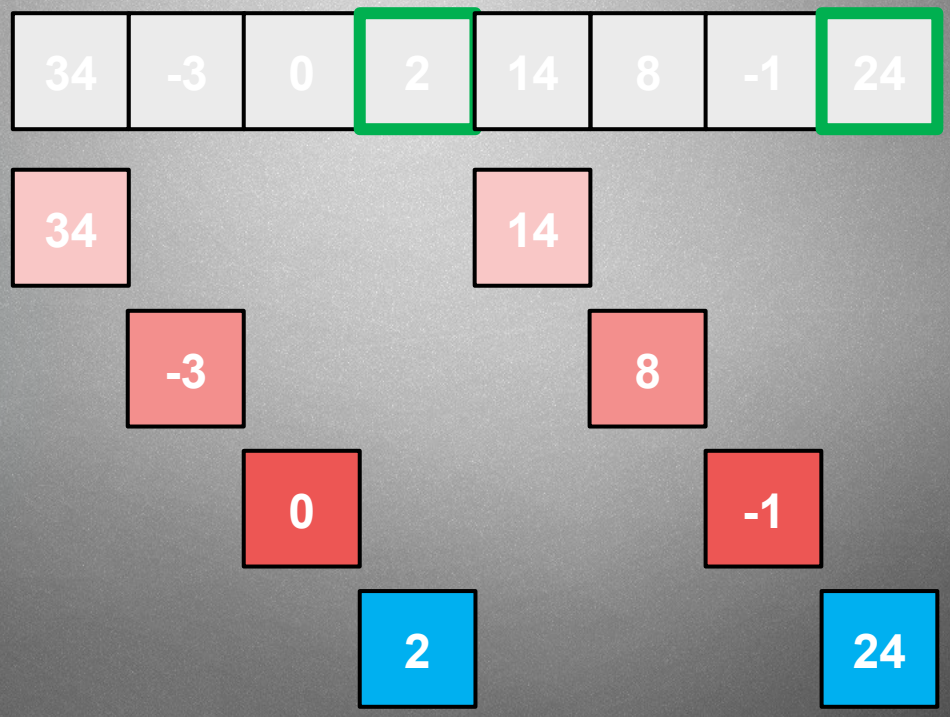
There are a lots of articles and models on how to choose the gaps
to be the most efficient !!!
~ we start at **4** → it means we consider every **4th** item and sort them




34
-3
0
2
14
8
-1
24
34
14


34
-3
0
2
14
8
-1
24
34
14
-3
8

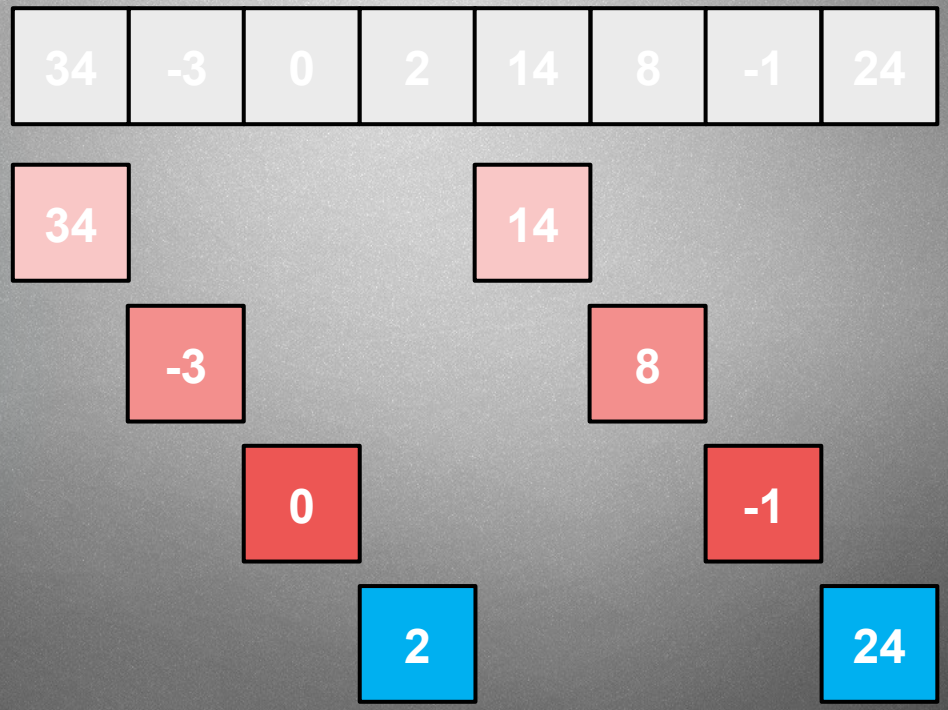

34
-3
0
2
14
8
-1
24
34
14
-3
8
0
-1


34
-3
0
2
14
8
-1
24
34
14
-3
8
0
-1
2
24

So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

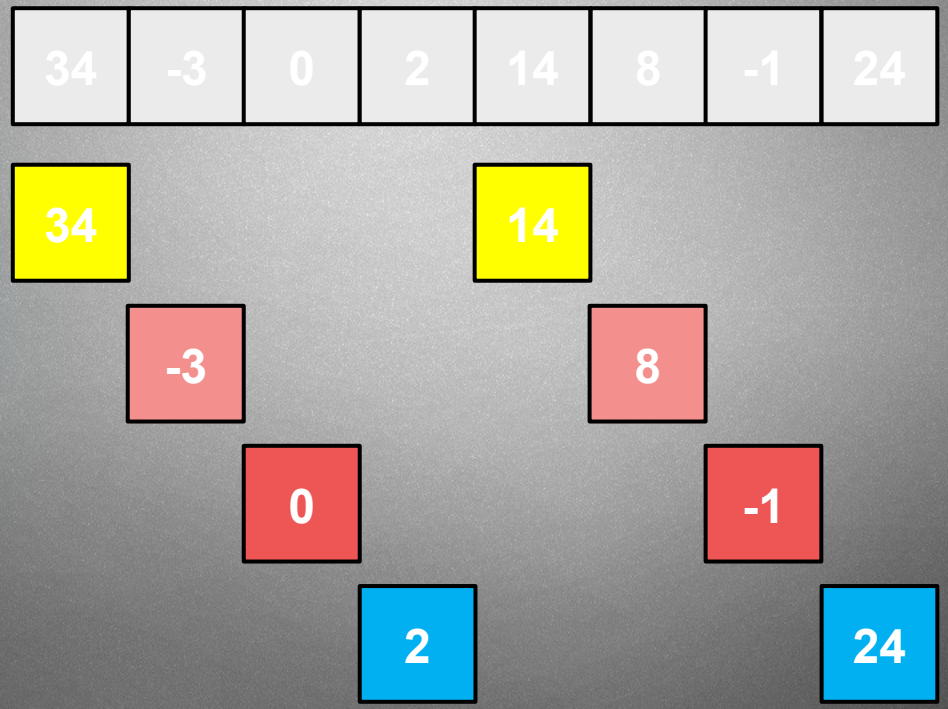
+ insert them back into the original array



So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

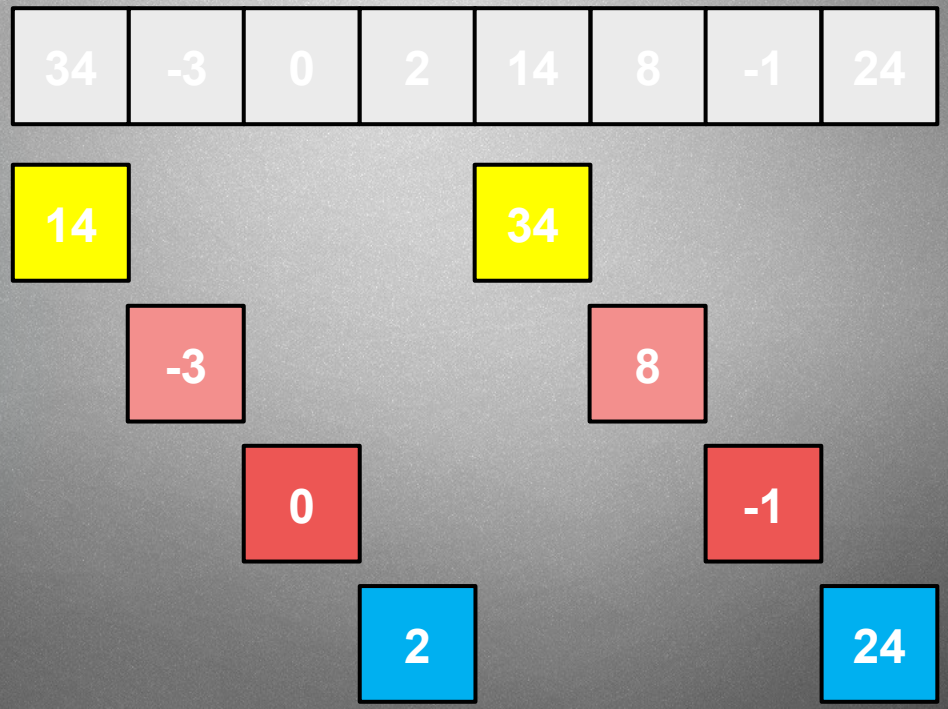
+ insert them back into the original array



So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

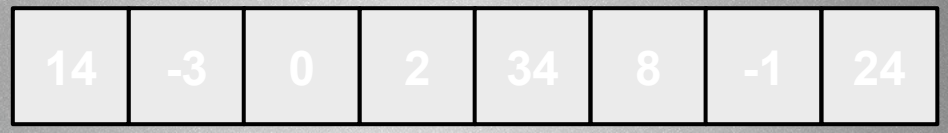
+ insert them back into the original array

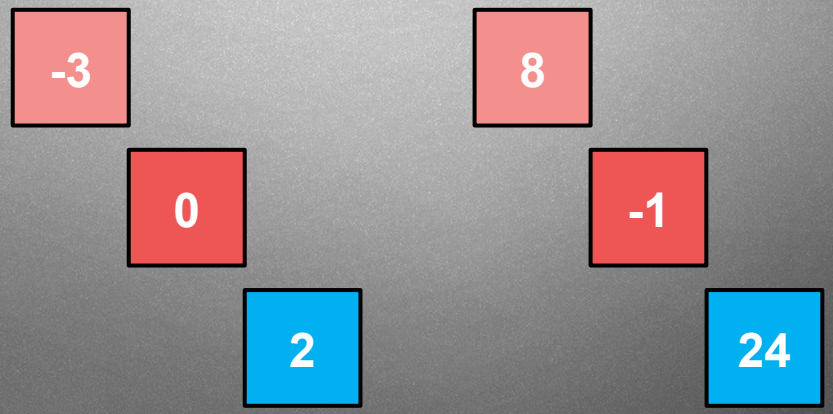


So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

+ insert them back into the original array

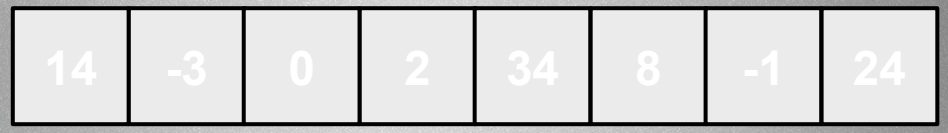


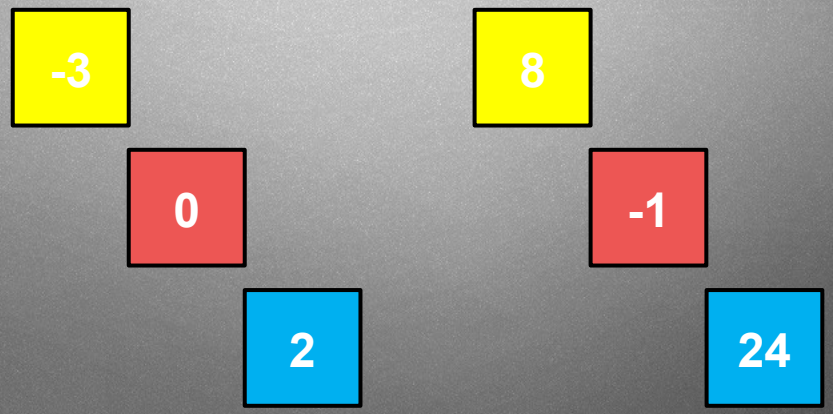


So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

+ insert them back into the original array

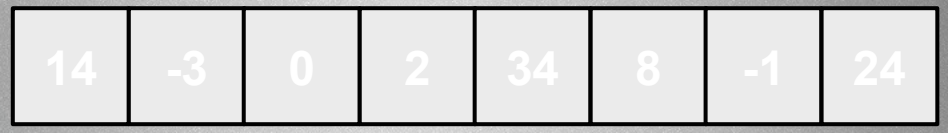


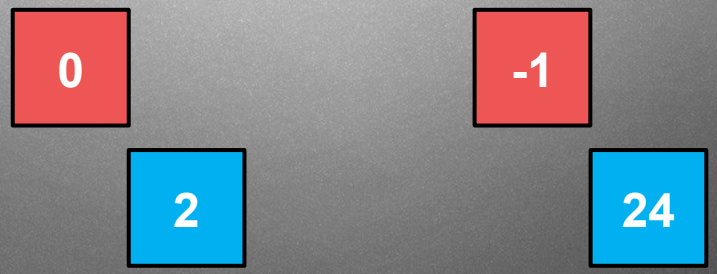


So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

+ insert them back into the original array

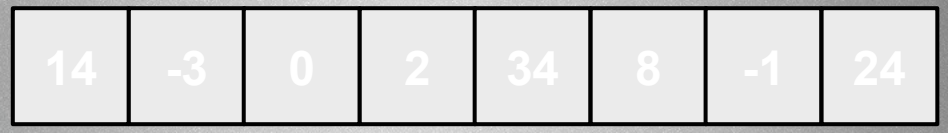


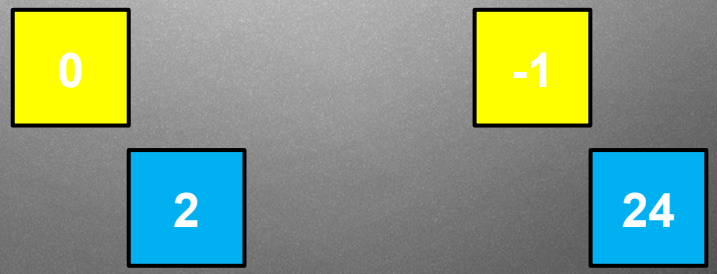


So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

+ insert them back into the original array

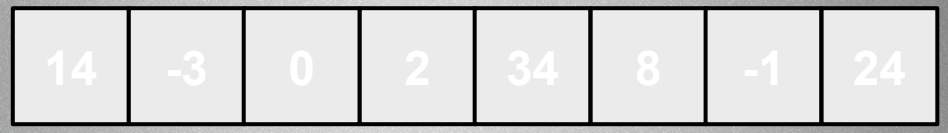


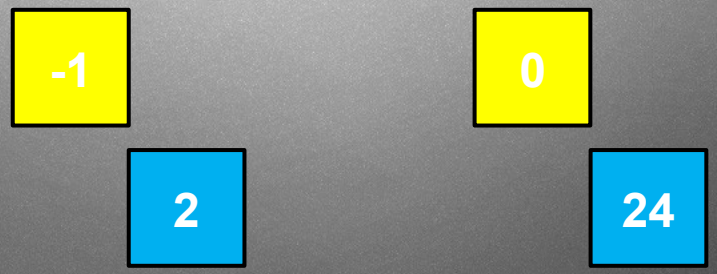


So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

+ insert them back into the original array

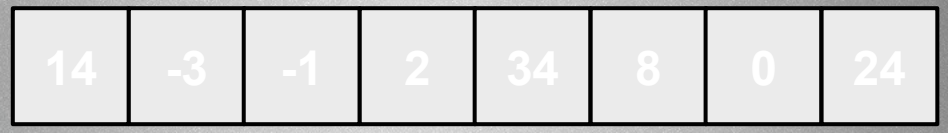




So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

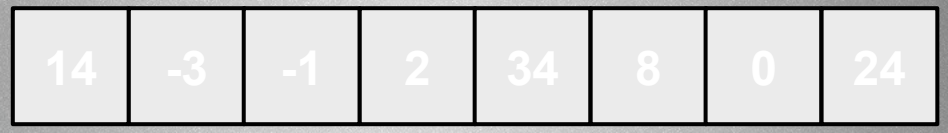
+ insert them back into the original array



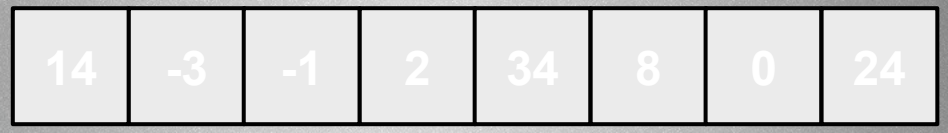
So we have managed to found all the subarrays with gap **4**, meaning all the subarrays where the items are **4** units away from each other

~ we have to sort these sublists independently

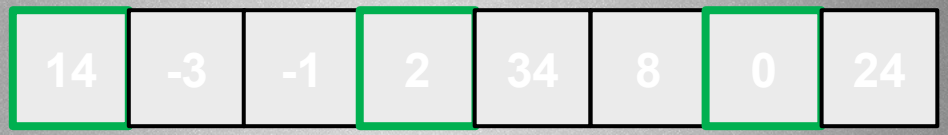
+ insert them back into the original array



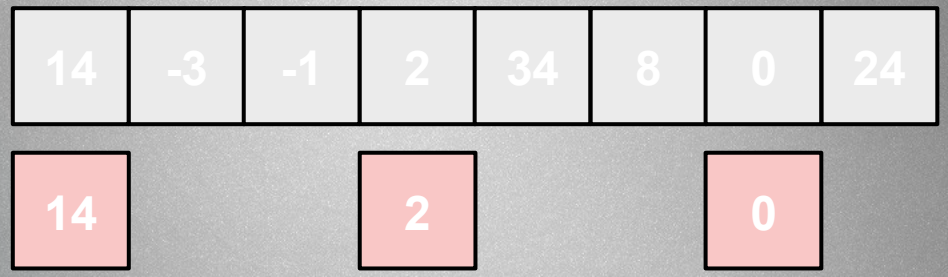
The gap so far was **4** units → lets decrease it to be **3** units



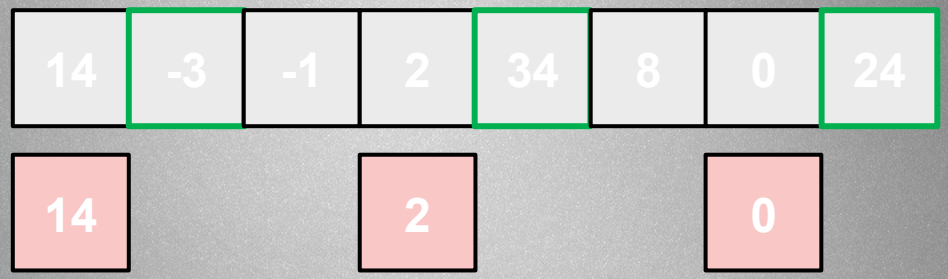
The gap so far was **4** units → lets decrease it to be **3** units



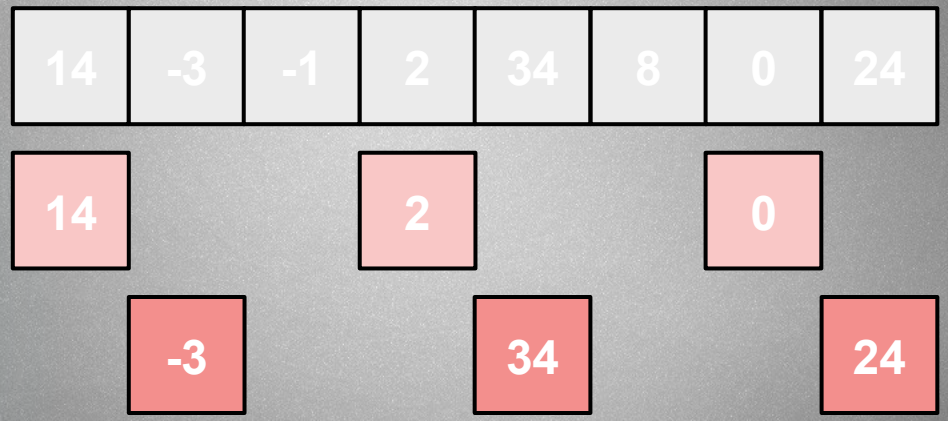
The gap so far was **4** units → lets decrease it to be **3** units



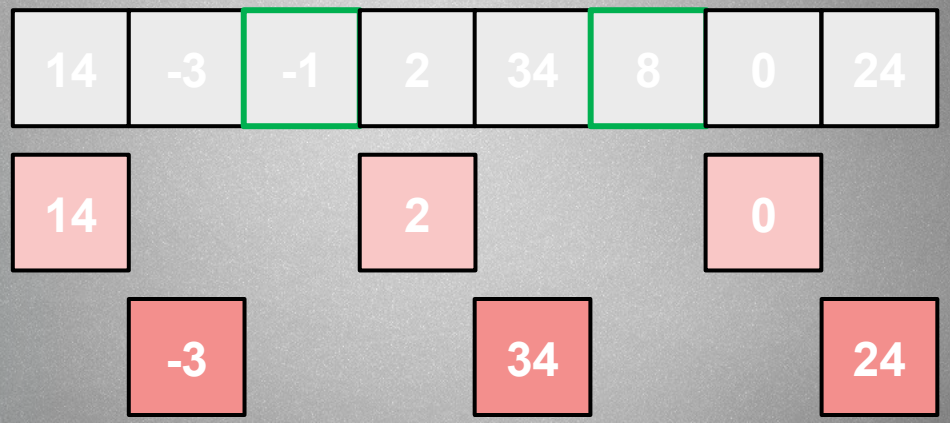
The gap so far was **4** units → lets decrease it to be **3** units



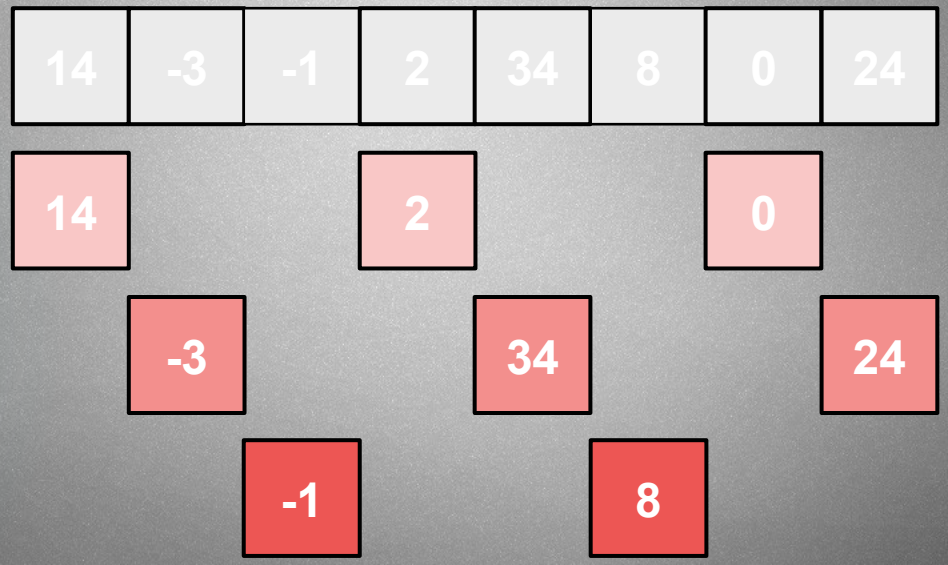
The gap so far was **4** units → lets decrease it to be **3** units



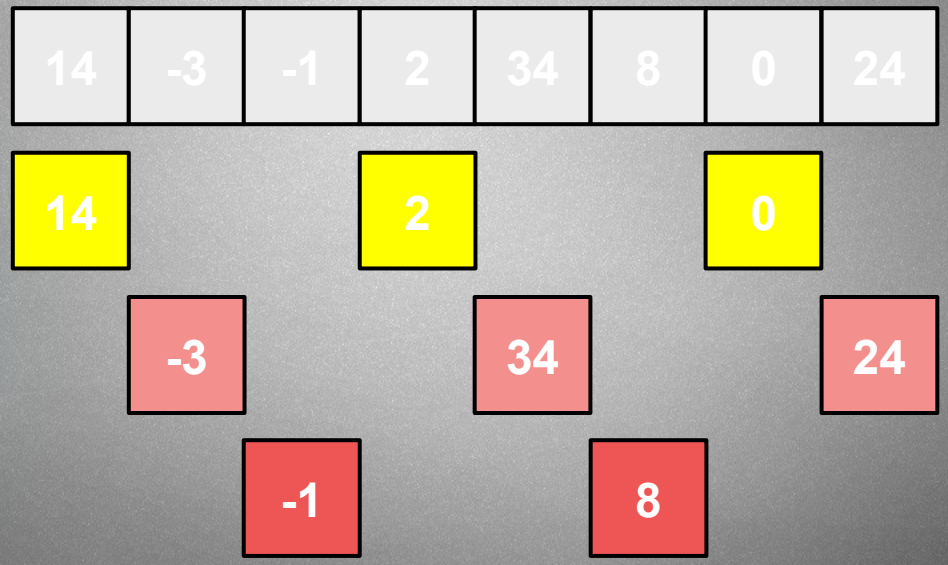
The gap so far was **4** units → lets decrease it to be **3** units



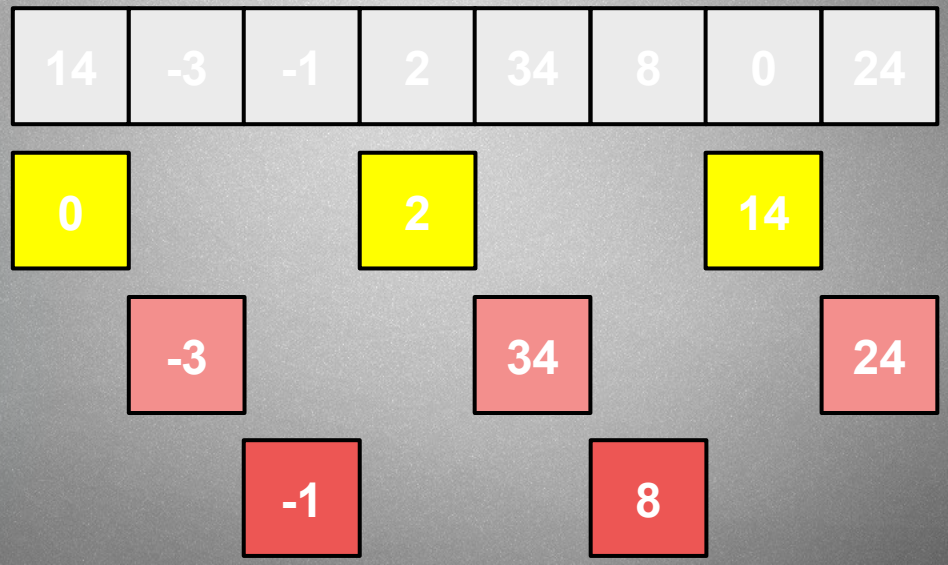
The gap so far was **4** units → lets decrease it to be **3** units



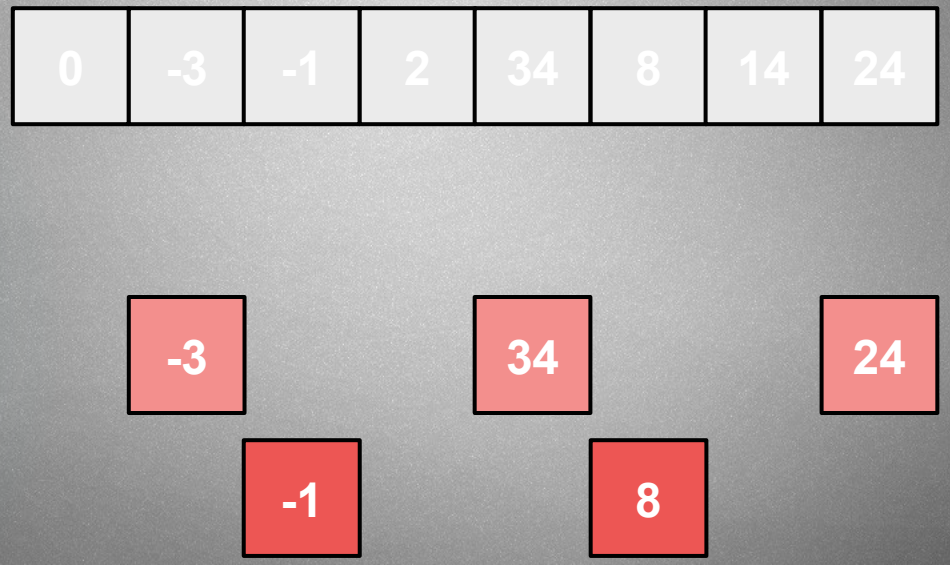
The gap so far was **4** units → lets decrease it to be **3** units



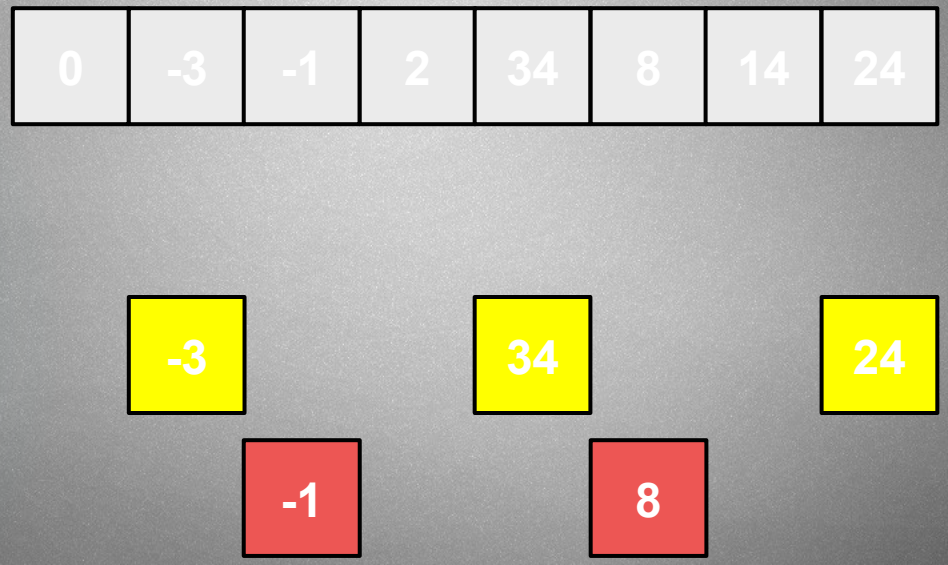
The gap so far was **4** units → lets decrease it to be **3** units



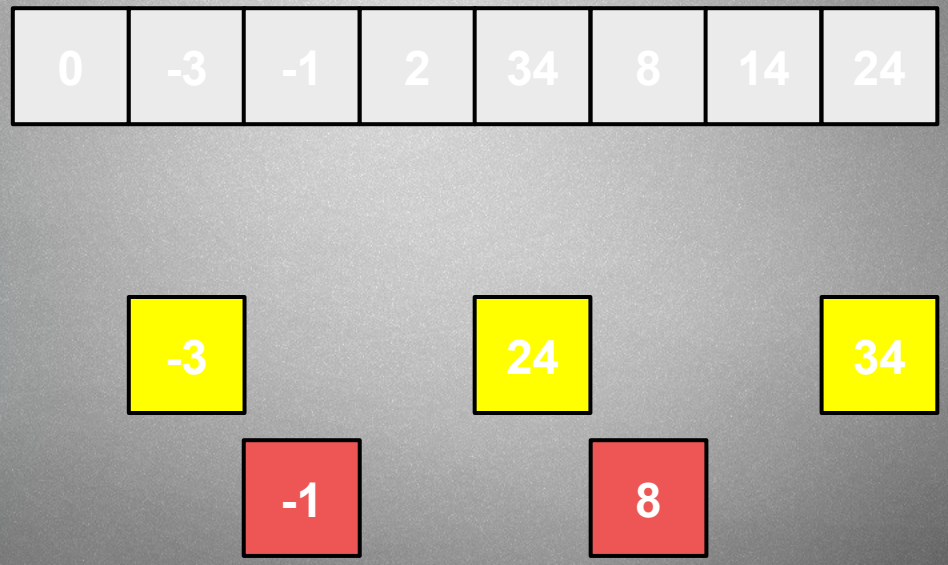
The gap so far was **4** units → lets decrease it to be **3** units



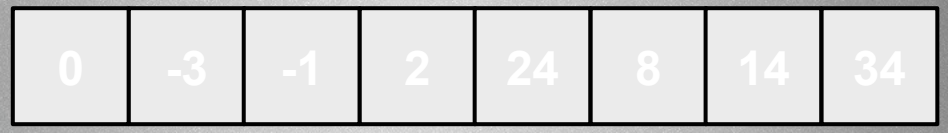
The gap so far was **4** units → lets decrease it to be **3** units



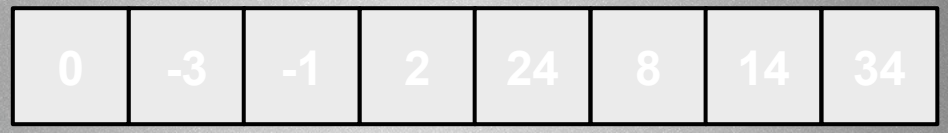
The gap so far was **4** units → lets decrease it to be **3** units



The gap so far was **4** units → lets decrease it to be **3** units

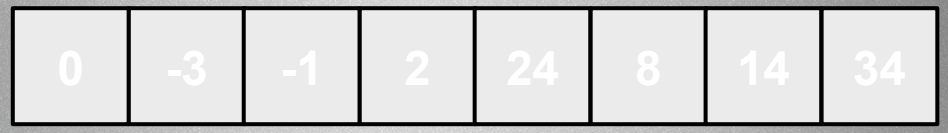


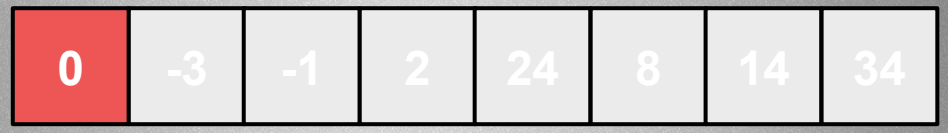
The gap so far was **4** units → lets decrease it to be **3** units

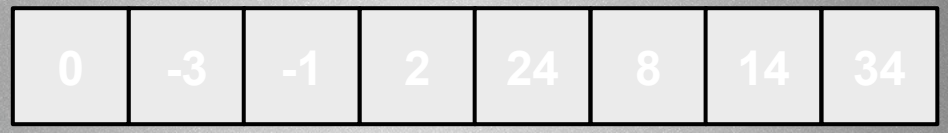


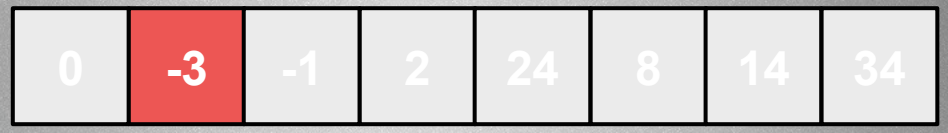
The gap so far was **3** units → lets decrease it to be **1** unit

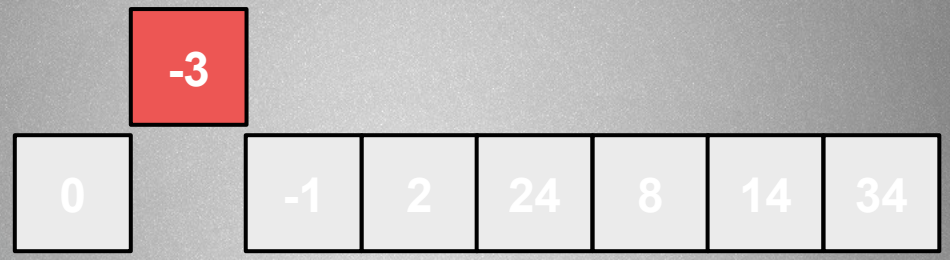
What does it mean **1** unit? It is basically the insertion sort !!!

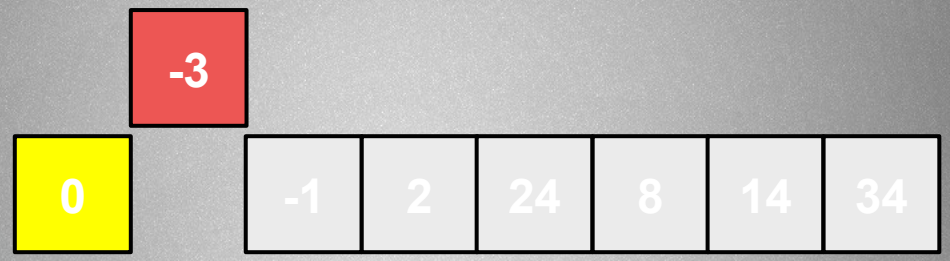



0
-3
-1
2
24
8
14
34

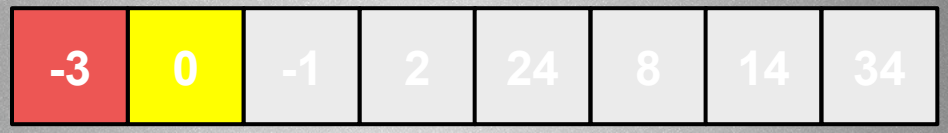

0
-3
-1
2
24
8
14
34

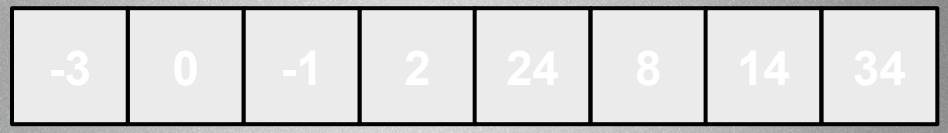

0
-3
-1
2
24
8
14
34

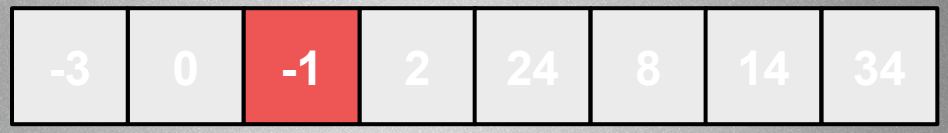

-3
0
-1
2
24
8
14
34

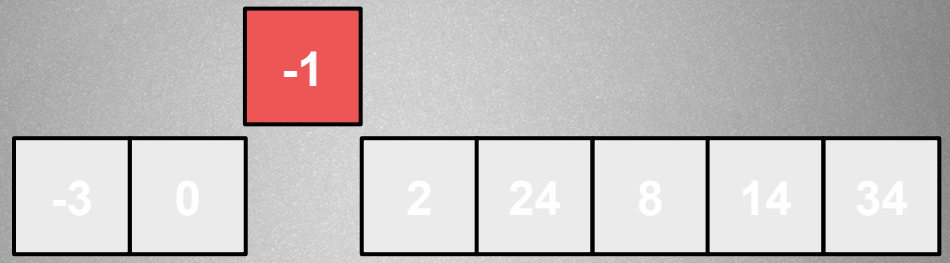

-3
0
-1
2
24
8
14
34

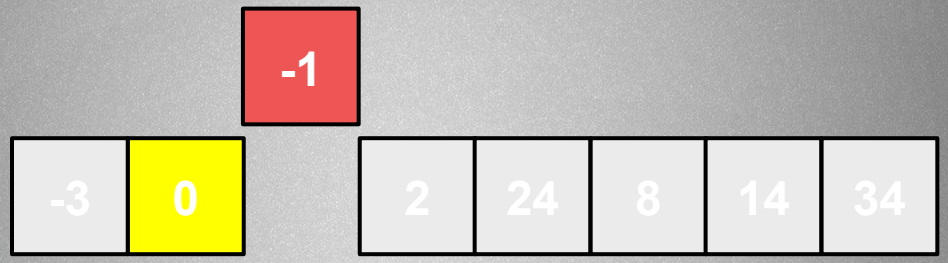

-3
0
-1
2
24
8
14
34

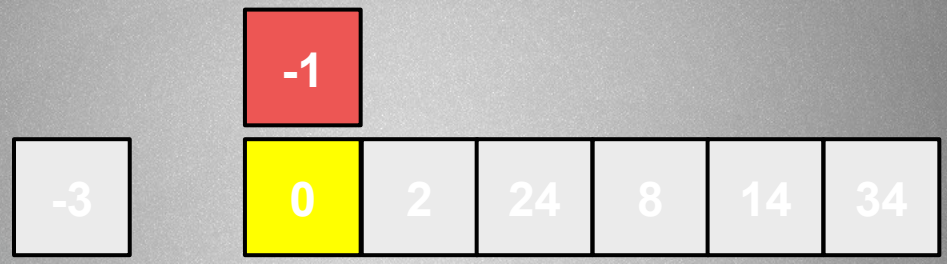

-3
0
-1
2
24
8
14
34

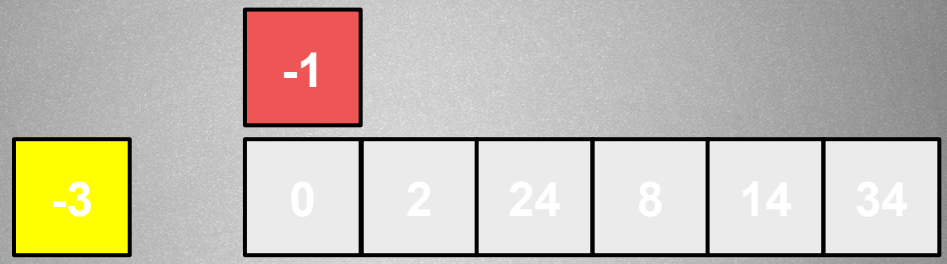

-3
0
-1
2
24
8
14
34

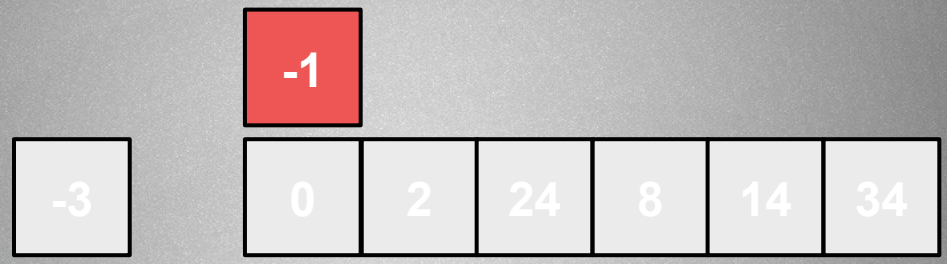

-3
0
-1
2
24
8
14
34

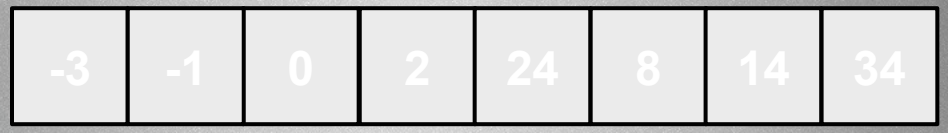

-1
-3
0
2
24
8
14
34

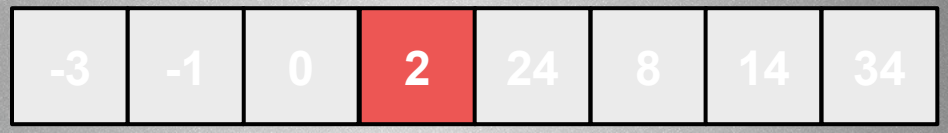

-1
-3
0
2
24
8
14
34

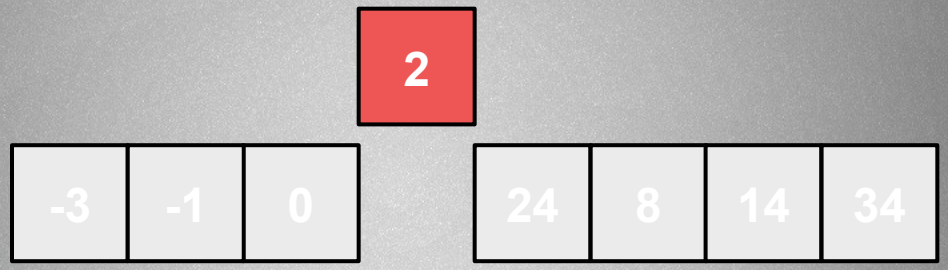

-1
-3
0
2
24
8
14
34

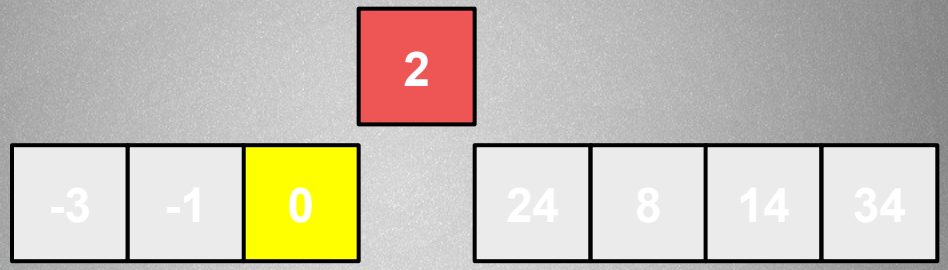

-1
-3
0
2
24
8
14
34

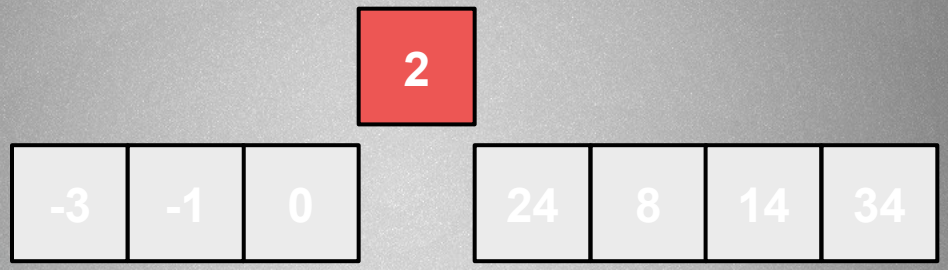

-1
-3
0
2
24
8
14
34

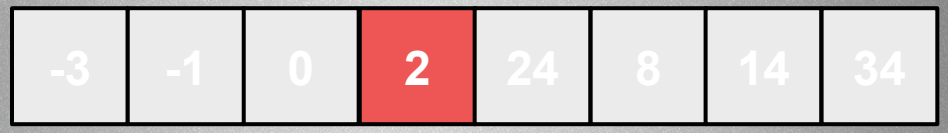

-3
-1
0
2
24
8
14
34

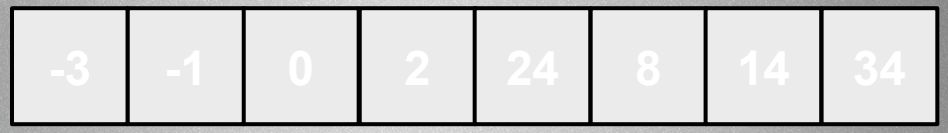

-3
-1
0
2
24
8
14
34

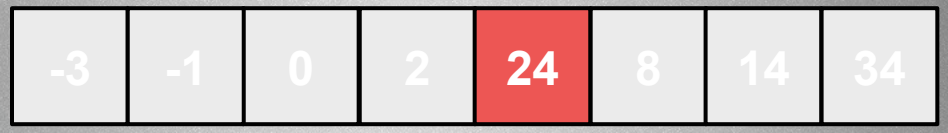

2
-3
-1
0
24
8
14
34

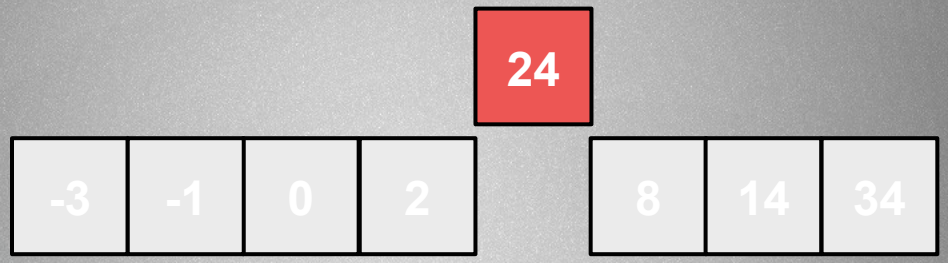

2
-3
-1
0
24
8
14
34

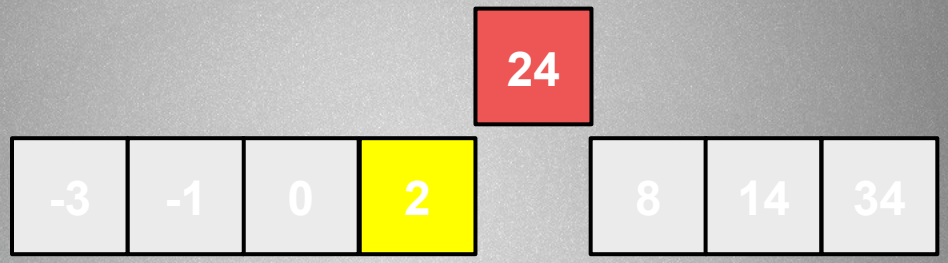

2
-3
-1
0
24
8
14
34

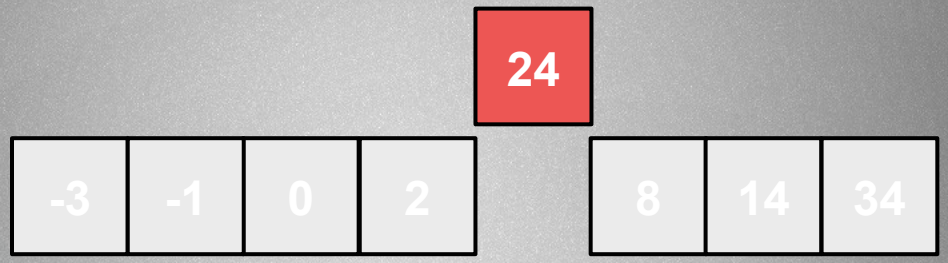

-3
-1
0
2
24
8
14
34

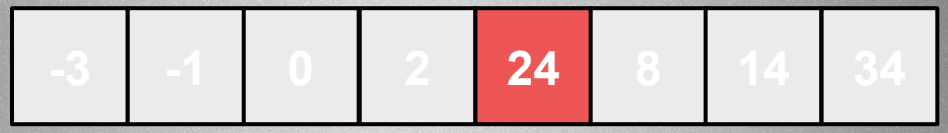

-3
-1
0
2
24
8
14
34

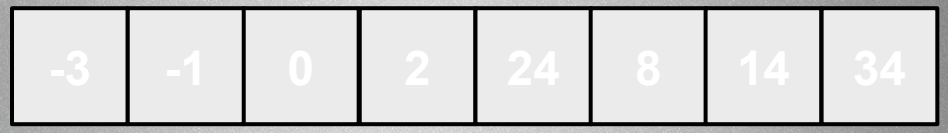

-3
-1
0
2
24
8
14
34

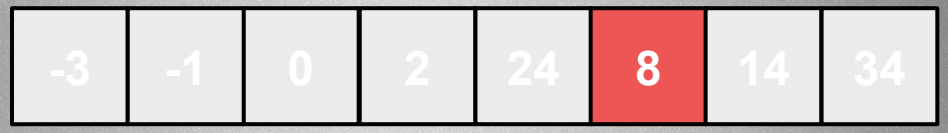

24
-3
-1
0
2
8
14
34

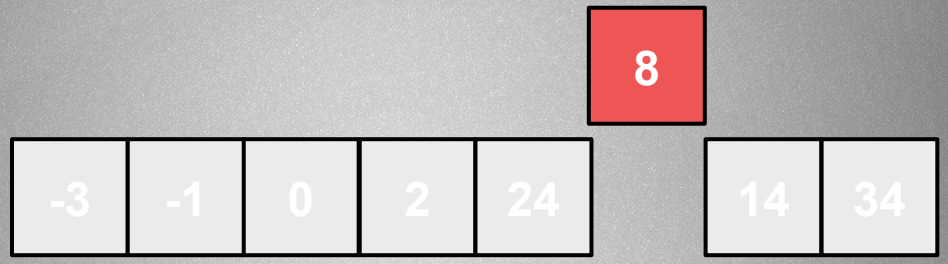

24
-3
-1
0
2
8
14
34

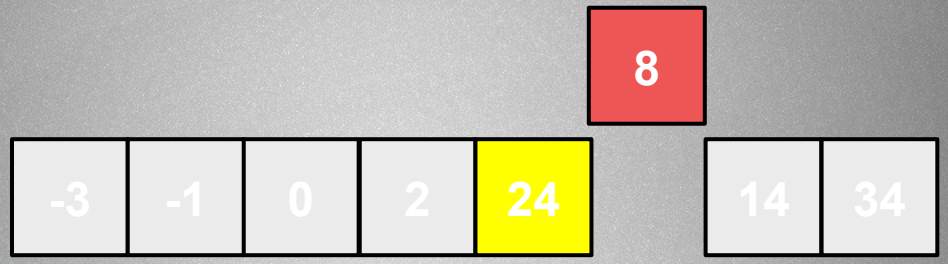

24
-3
-1
0
2
8
14
34

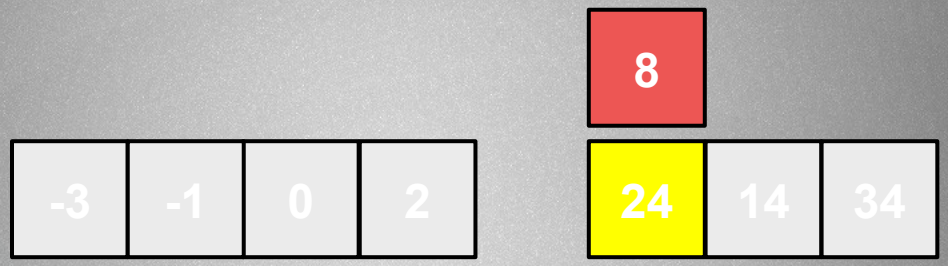

-3
-1
0
2
24
8
14
34

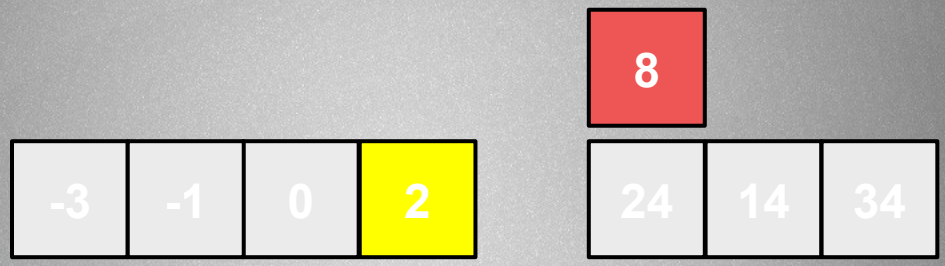

-3
-1
0
2
24
8
14
34

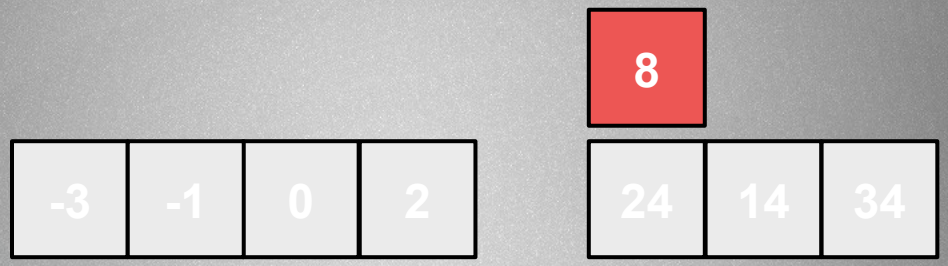

-3
-1
0
2
24
8
14
34

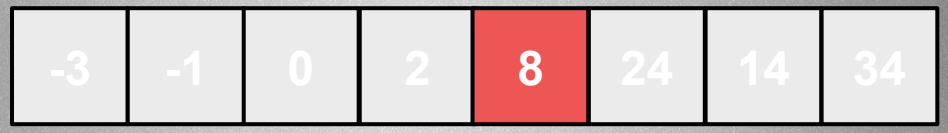

8
-3
-1
0
2
24
14
34

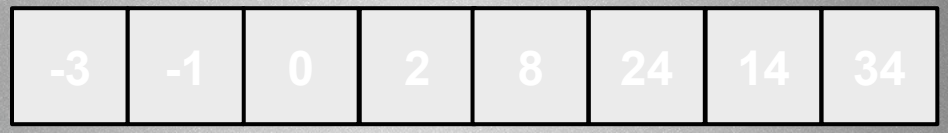

8
-3
-1
0
2
24
14
34

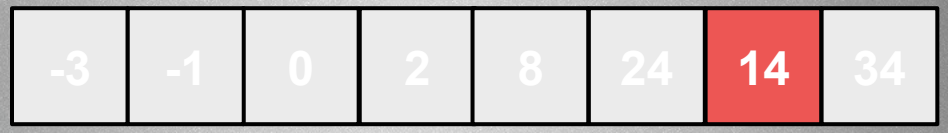

8
-3
-1
0
2
24
14
34

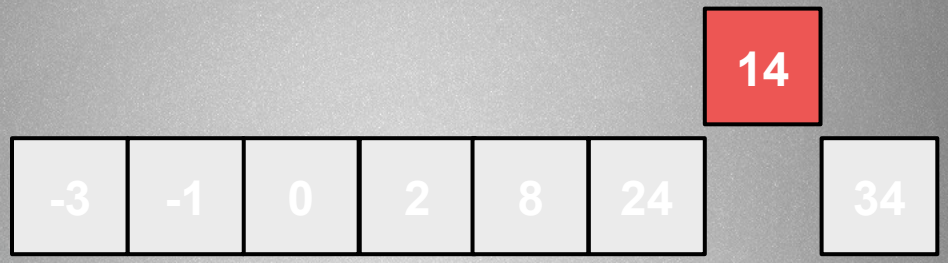

8
-3
-1
0
2
24
14
34

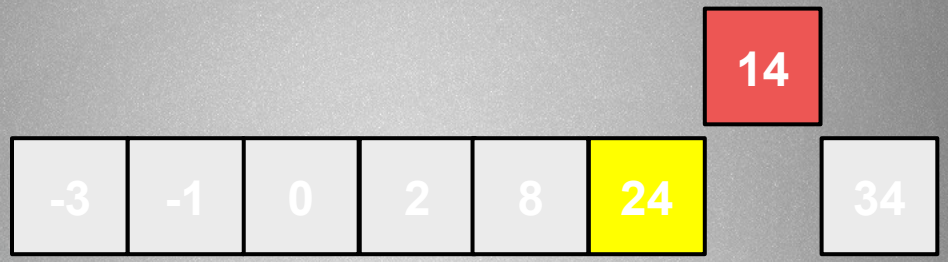

8
-3
-1
0
2
24
14
34

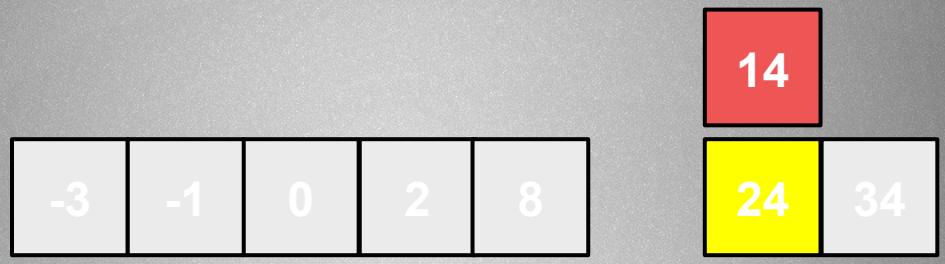

-3
-1
0
2
8
24
14
34

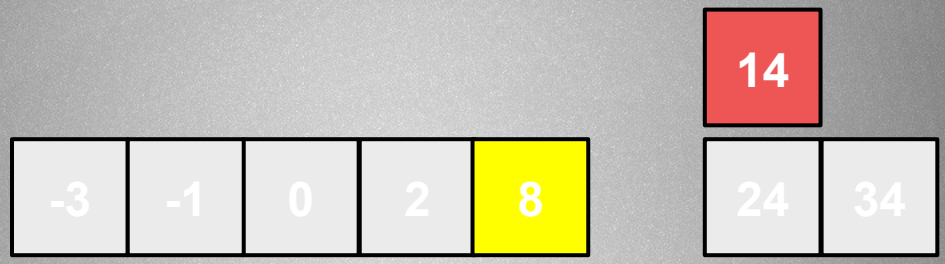

-3
-1
0
2
8
24
14
34

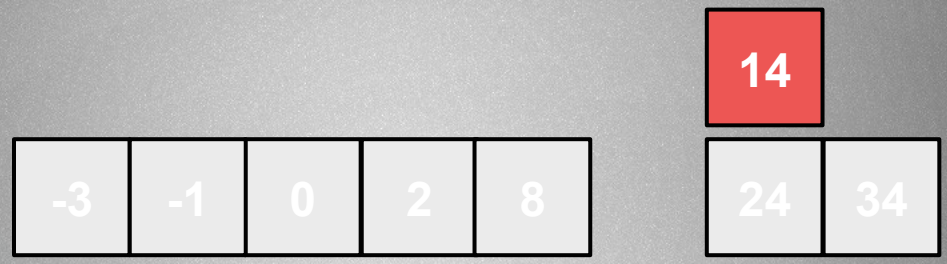

-3
-1
0
2
8
24
14
34

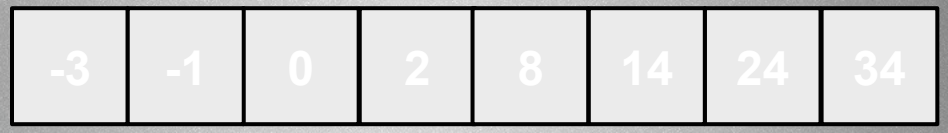

14
-3
-1
0
2
8
24
34

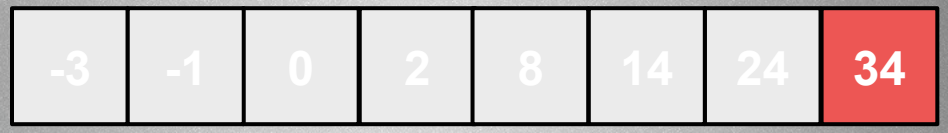

14
-3
-1
0
2
8
24
34

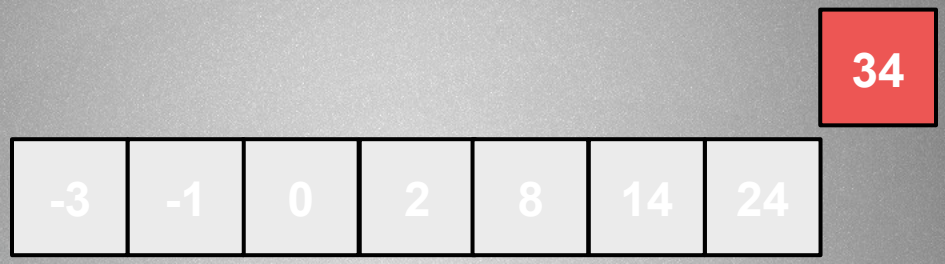

14
-3
-1
0
2
8
24
34

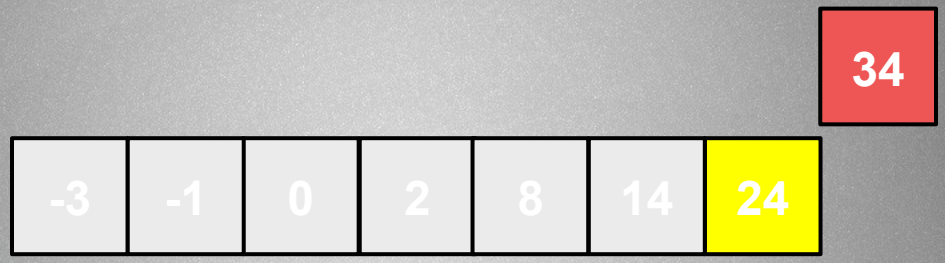

14
-3
-1
0
2
8
24
34

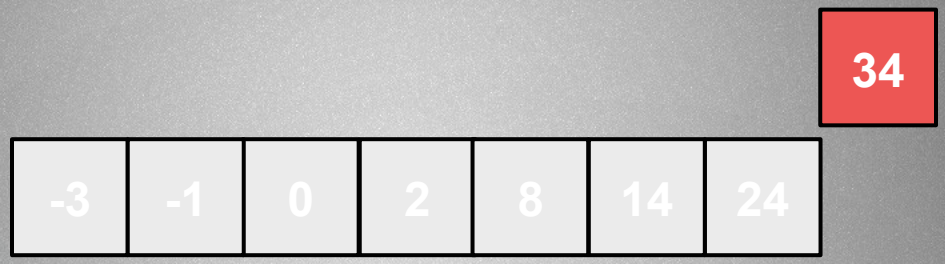

14
-3
-1
0
2
8
24
34

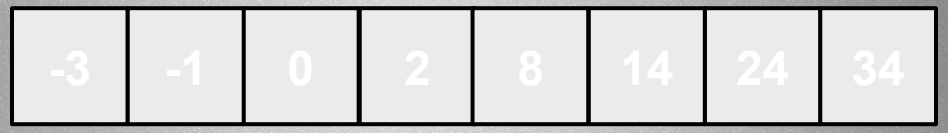

-3
-1
0
2
8
14
24
34

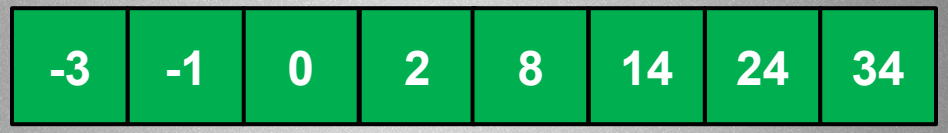

-3
-1
0
2
8
14
24
34


34
-3
-1
0
2
8
14
24


34
-3
-1
0
2
8
14
24


34
-3
-1
0
2
8
14
24


-3
-1
0
2
8
14
24
34


-3
-1
0
2
8
14
24
34