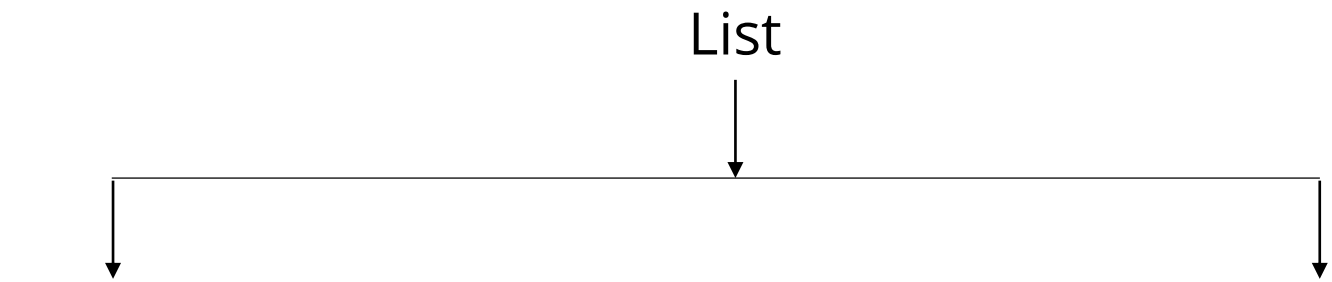
Classification of List

“A deeper look”

Prerequisite: ArrayList

Md. Saidul Hoque Anik
onix.hoque.mist@gmail.com

Classification



Ordered List

Sequence is important

Sorted List

Sequence is important
+ Sorted

- SortedArrayList
- SortedLinkedList

0	1	2	3	4
5	2	11	9	4

0	1	2	3	4
2	4	5	9	11

Additional Operations of SortedList

1. FindPosition (val) - find the position of an object in the sorted list

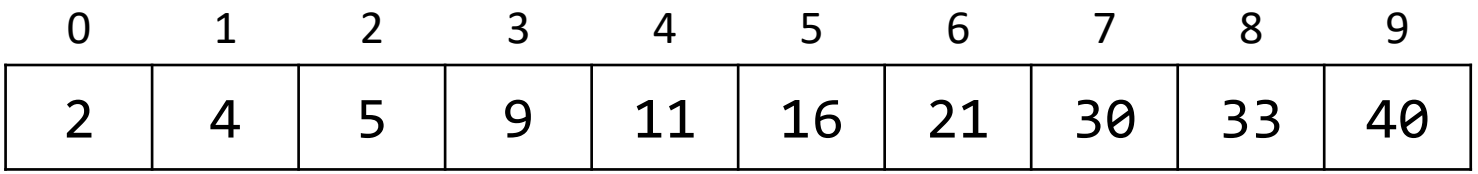
2. getAt(index) - access the object at a given position in the sorted list

3. Remove(index) - remove the object at a given position from the sorted list.

Comparison

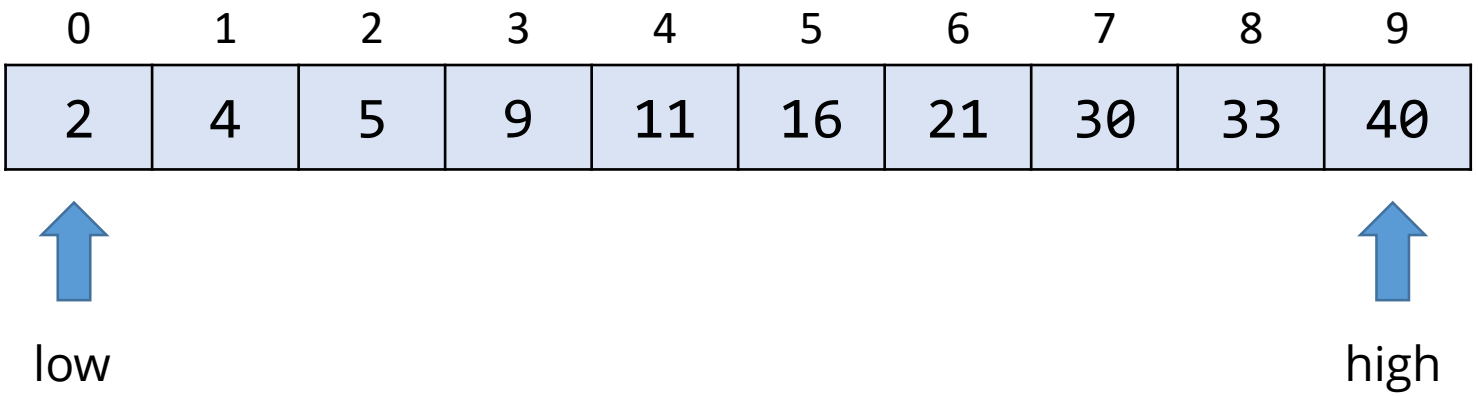
	Sorted list implementation	
Function	SortedListAsArray	SortedListAsLinkedList
Insert	*O*(*n*)	*O*(*n*)
IsMember	*O*(*n*)	*O*(*n*)
Find	O(logn)	*O*(*n*)
FindPosition	O(logn)	*O*(*n*)
Withdraw	*O*(*n*)	*O*(*n*)

Binary Search in Sorted List



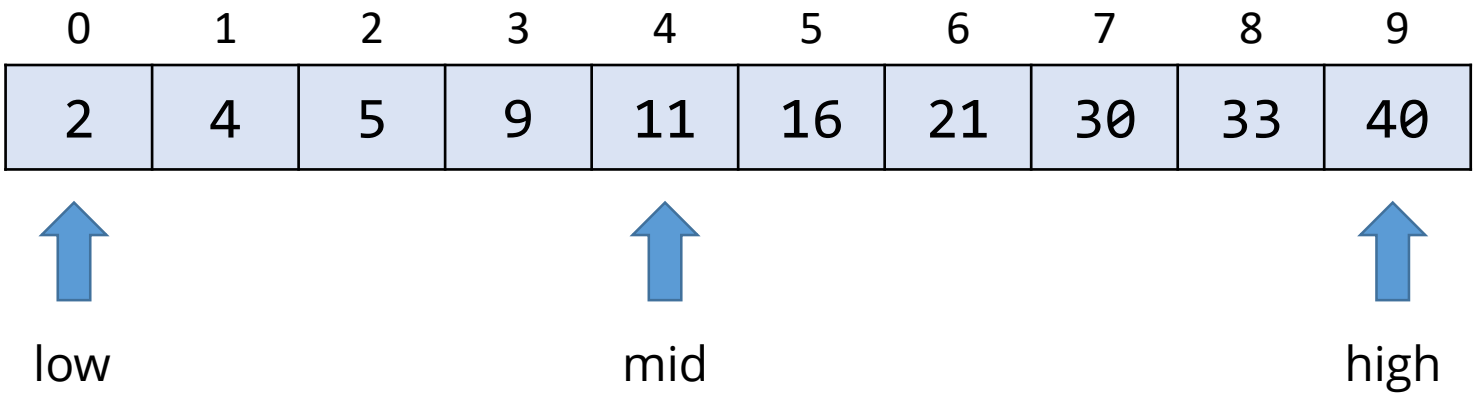
Binary Search in Sorted List

Look for 9



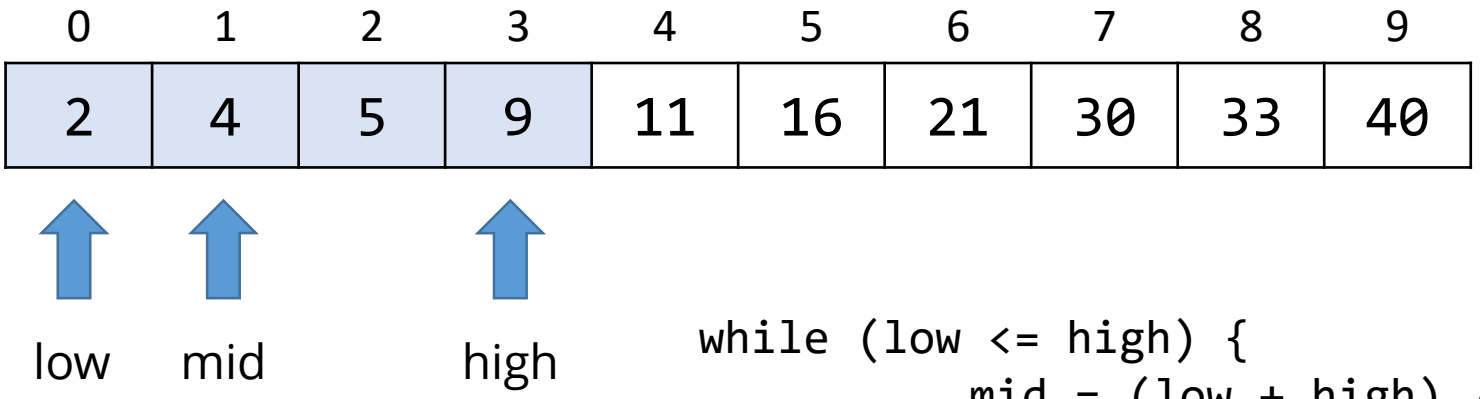
Binary Search in Sorted List

Look for 9



Binary Search in Sorted List

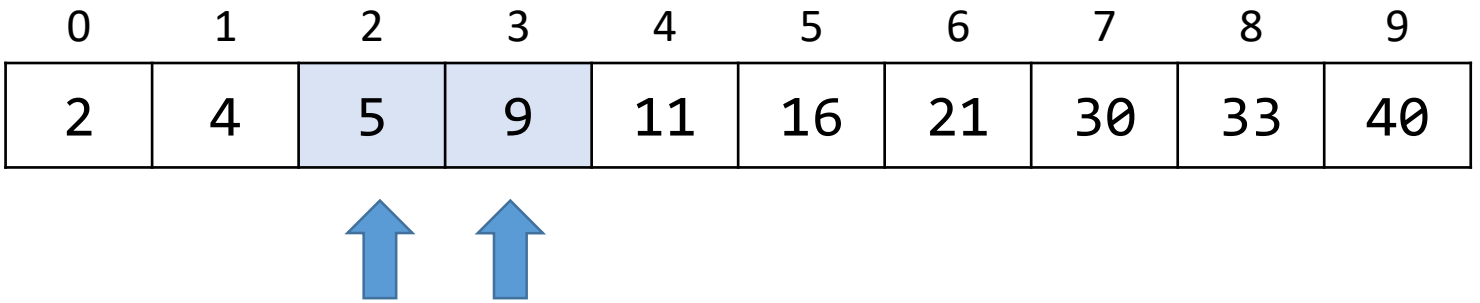
Look for 9




```
while (low <= high) {
        mid = (low + high) / 2
        if (A[mid] > value)
            high = mid - 1
        else if (A[mid] < value)
            low = mid + 1
        else
            return mid
    }
```

Binary Search in Sorted List

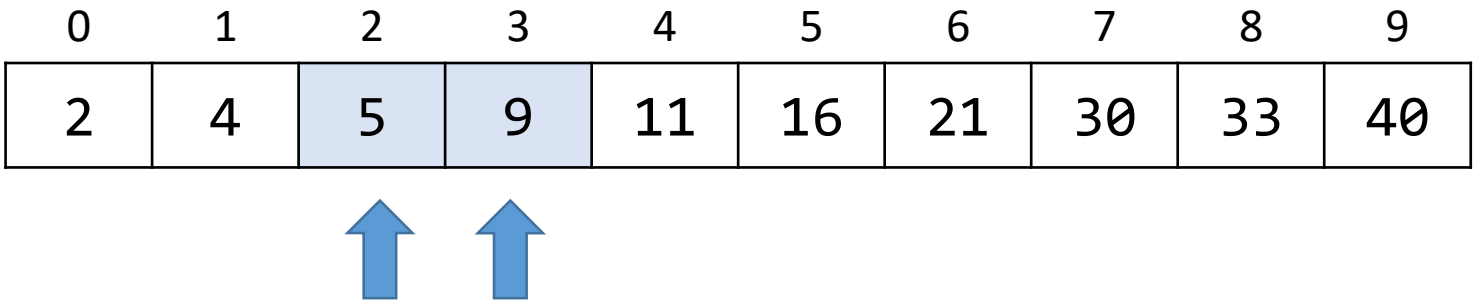
Look for 9




```
while (low <= high) {
        mid = (low + high) / 2
        if (A[mid] > value)
            high = mid - 1
        else if (A[mid] < value)
            low = mid + 1
        else
            return mid
    }
```

Binary Search in Sorted List

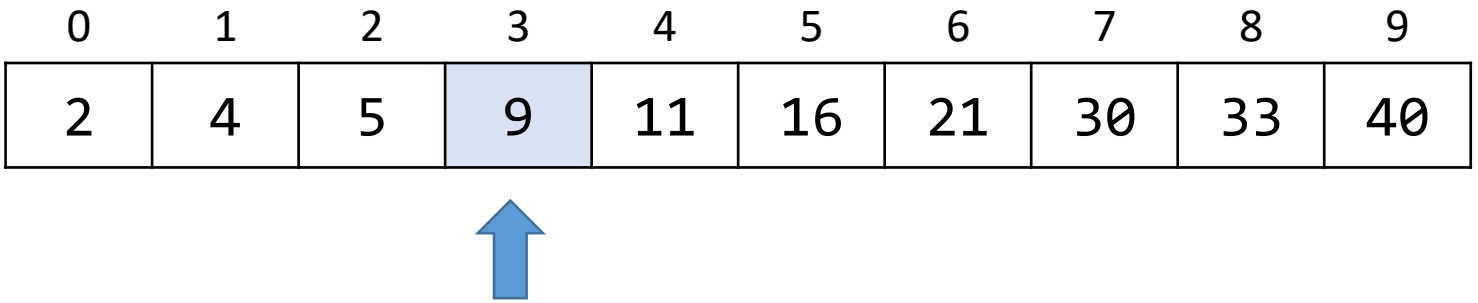
Look for 9




```
while (low <= high) {
        mid = (low + high) / 2
        if (A[mid] > value)
            high = mid - 1
        else if (A[mid] < value)
            low = mid + 1
        else
            return mid
    }
```

Binary Search in Sorted List

Look for 9




```
while (low <= high) {
        mid = (low + high) / 2
        if (A[mid] > value)
            high = mid - 1
        else if (A[mid] < value)
            low = mid + 1
        else
            return mid
}
```

Binary Search in Sorted List

```
BinarySearch(A[0..N-1], value) {
      low = 0
      high = N - 1
      while (low <= high) {
          mid = (low + high) / 2
          if (A[mid] > value)
              high = mid - 1
          else if (A[mid] < value)
              low = mid + 1
          else
              return mid
      }
      return not_found
}
```

Complexity Analysis

```
BinarySearch(A[0..N-1], value) {
        low = 0
        high = N - 1
        while (low <= high) {
            // invariants: value > A[i] for all i < low
            //             value < A[i] for all i > high
            mid = (low + high) / 2
            if (A[mid] > value)
                high = mid - 1
            else if (A[mid] < value)
                low = mid + 1
            else
                return mid
        }
        return not_found
}
```

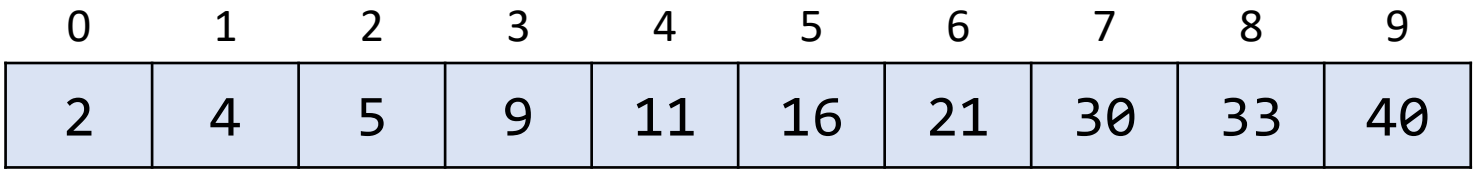
Complexity Analysis

What will be the complexity of the while loop?

```
BinarySearch(A[0..N-1], value) {
        low = 0
        high = N - 1
        while (low <= high) {
            // invariants: value > A[i] for all i < low
            //             value < A[i] for all i > high
            mid = (low + high) / 2
            if (A[mid] > value)
                high = mid - 1
            else if (A[mid] < value)
                low = mid + 1
            else
                return mid
        }
        return not_found
}
```

Complexity Analysis

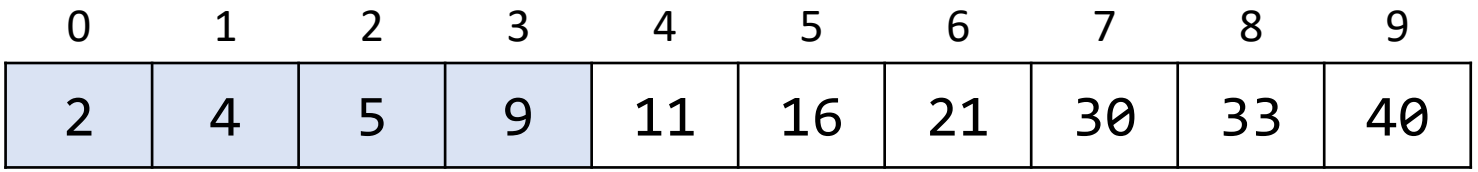
What will be the complexity of the while loop?



Total Searchable item : 10

Complexity Analysis

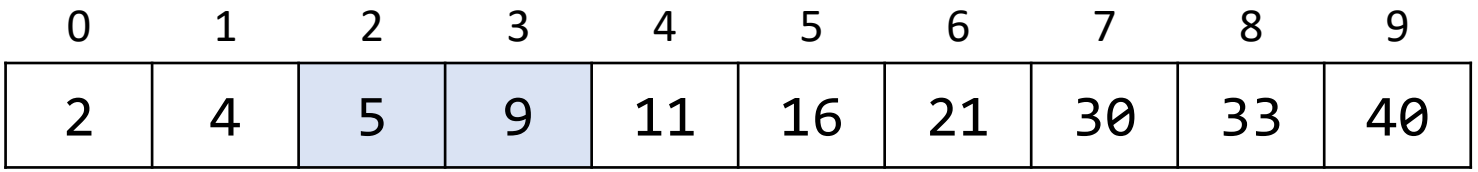
What will be the complexity of the while loop?



Total Searchable item : 4

Complexity Analysis

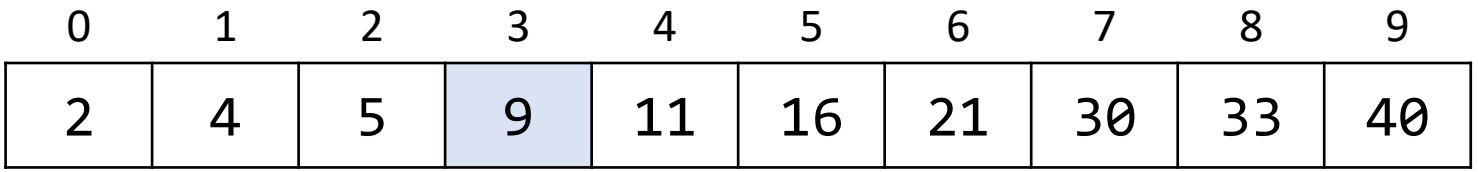
What will be the complexity of the while loop?



Total Searchable item : 2

Complexity Analysis

What will be the complexity of the while loop?



Total Searchable item : 1

Reference

- https://book.huihoo.com/data-structures-and-algorithms-with-object-oriented-design-patterns-in-c++/html/page166.html

Source: Wikipedia