Hence, the worst-case time complexity of the binary search algorithm is *O* **(log n)**

3.3 Analysis of sorting algorithm - Bubble Sort

In a bubble sort algorithm, the first and the second data elements are compared. In the first iteration, if the first data element is greater than the second, then they are swapped. This process is continued till the last data item is reached. The comparison and swapping operations are controlled by the inner 'for' loop. This completes one pass, by the end of which the largest data item occupies the last position.

This step is repeated for all the remaining elements completing a total of n-1 passes. At the end of the last pass, the entire list is sorted. The Algorithm is shown in Figure-11

```
begin
 for i ← 1 to N
  for j ← 1 to N - 1
    if (Arr[j] > Arr[j+1])
     swap (Arr[j] > Arr[j+1])
    end if
  end for
 end for
end
```

FIGURE 3: BUBBLE SORT

Now, let's find the time complexity of a bubble sort algorithm. In the first pass, the number of elements to be sorted is 'n', which requires 'n' comparisons. At the end of the first pass, one element would be in its sorted position.

In the second pass, the number of elements to be sorted is n-1 which requires n-2 comparisons. Continuing these passes till all the elements are sorted, the total number of comparisons are

= (n-1) + (n-2) + (n-3) +..... + 1

This translates to the sum of the first (n-1) natural numbers = $\frac{(n-1)n}{2} = \frac{n^2-n}{2} = O(n^2)$

‹ qsort() Library Function Demo (/?q=MULNCourseBook/listDetailedCBLearningContents/178099/cidb/full/view/153784/431550/431535/39731)