|  |
| --- |
| 2.6 |
| Write an algorithm that searches a sorted list of n items by dividing it into three sublists of almost n=3 items. This algorithm finds the sublist that might contain the given item and divides it into three smaller sublists of almost equal size. The algorithm repeats this process until it finds the item or concludes that the item is not in the list. Analyze your algorithm and give the results using order notation. |

Divide the array of size into three parts by taking third1 and third 2 with the formulas:

*third1 = l + (r-l)/3   
third2 = r – (r-l)/3*

For first iteration l=0 and r=n-1

Sort the array if it’s not sorted.

Search for key X**:**

1. Compare X with the element at position third1. If equal return third1.
2. Compare X with the element at position third2. If equal return third2.
3. Check if X is less than the element at third1. If yes, then recursion to first third.
4. Check if X is greater than the element at third2. If yes, then recursion to third part.
5. If neither, then recursion to the second third.

**Complexity**

Time: log3n

Space: n

|  |
| --- |
| 2.7 |
| Use the divide and conquer approach to write an algorithm that finds the largest item in a list of n items. Analyze your algorithm and show the results in order notation. |

First the l=0 and h=n-1.

Recursive formula:

1. Base case just an array of size 1, return the value
2. Having l=0 and h=n-1 at first.
3. Find the value at the middle=(l+h)/2
4. Max at Left=Recursion with sub array from l to middle
5. Max at Right= Recursion with sub array from middle to h
6. Return the larger between Left and Right.

**Complexity**

Time: log n

Space: n

|  |
| --- |
| 2.13 |
| Write an algorithm that sorts a list of n items by dividing it into three sublists of about n=3 items sorting each sublist recursively and merging the three sorted sublists. Analyze your algorithm and give the results under order notation. |

Divide the array of size into three parts by taking third1 and third 2 with the formulas:

*third1 = l + (r-l)/3   
third2 = r – (r-l)/3*

The Base case is a list of size one, which is by definition sorted

Merge sort dividing the list into equal thirds with until it can no more be divided. By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

1. Having l=0 and h=n-1 at first.
2. Find the value at the middle=(l+h)/2
3. Recursively divide the list into three parts
4. Return the merge of the smaller lists

**Complexity**

Time: log3n

Space: n