Design and Analysis of the Algorithm: -

So, in this problem each input is an array of integers which can be any of the four distributions that is mentioned in the problem statement.

We have used divide and conquer approach which helped us ease the solution.

***Note***: - Interpretation of diagram can be taken from leaf nodes and go till the root node. So, that the state change will be clear. Basically, the diagram portrays from bottom-up.

Let’s take an example: -

1. 3 5 7 9 11 13 15 17 19

![A screenshot of a cell phone

Description automatically generated]()

The above figure showing a case of strictly increasing. So, our approach goes dividing them and during merge we compare the values and based on the state and after each small merge we save the state and for next comparison it uses the previous state and also compares the last element of first chunk and first element of the second chunk. In that way we monitor the changes in the data for unit chunks and on whole we get the distribution.

Let’s also consider a case of maxima.

1. 3 5 7 9 10 9 8 7 6 5

![A screenshot of a cell phone

Description automatically generated]()

Here, also the same approach is taken, and the logic considers even if the previous state is different from the current state while merging the chunks. For instance, in this case when the left chunk is 3 5 7 9 which is an increasing distribution and the right chunk is 10 9 8 7 6 which is a decreasing distribution, the code considers these cases and understands there is a dip in the distribution which is maxima and every time we compare the arr[m] and arr[m+1]. Since it is maxima, the dip point and hence it is 10. The code works similarly for minima and other case i.e. decreasing distribution.

Analysis of the algorithm: -

Suppose there are N elements in the input array we run a recursive code that splits them into smaller nodes and then we merge them back. Merging is just doing comparison of middle and middle + 1 element.

So, O(logN) will be for recursively splitting them.

For merging it is just doing *comparison* which takes constant time i.e. O(1).

So, the overall complexity of the algorithm is O(logN).

Space is taken for the constant enum of distributions and the input array of elements of size N.