|  |
| --- |
| Read Input File  Convert each line to list of integers  For each list of integers  #find distribution type  #Calculate Maxima and minima using divide and conquer in a recursive function  Function findMinAndMax(list,left,right,min,max)  #base condition for list having odd no.of elements  if list has only 1 element  if left=right  if min > left => min = left  if max < left => max = left  return min,max  #base condition for list having even no.of elements  if list has only 2 elements  if left > right  if min > right => min = right  if max < left => max = left  return min,max  # find mid element  mid = (left + right) // 2  # recur for left sub list  min, max = self.findMinAndMax(data, left, mid, min, max)  # recur for right sub list  minimum, maximum = self.findMinAndMax(data, mid + 1, right, min, max)  return min, max  #Predict distribution type based on  1. Comparison 1st two elements of the list : telling if we have to give min or max  2. Position of Maxima/Minima : telling if its increasing, decreasing , maxima or minima    #Write the result in output file |

**Design of the Program :**

How Does the Algorithm work?

|  |
| --- |
| A **divide-and-conquer algorithm** works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.   * Divide: In this approach, the array is divided into two halves. * Conquer: Using recursive approach maximum and minimum numbers in each halves are found. * Combine the results: Return the maximum of two maxima of each half and the minimum of two minima of each half. |

**Advantages over Brute force Approach**

|  |
| --- |
| * Reduces the degree of difficulty since it divides the problem into sub problems that are easily solvable * Usually runs faster than other algorithms would * Uses memory caches effectively. The reason for this is the fact that when the sub problems become simple enough, they can be solved within a cache, with-out having to access the slower main memory |

Time Complexity Analysis of the findMinMax() function

|  |
| --- |
| T(n)= T(n/2)+ c = O **θ (nlog2n)** |