|  |
| --- |
| 1. Read Input File 2. Convert each line to list of integers 3. For each list of integers – we do the following two steps  * Find distribution type * Calculate maxima and minima using the divide and conquer strategy          1. We then predict distribution type based on following two criteria  * Comparison 1st two elements of the list which helps us eliminate two possibilities out of four * Position of Maxima/Minima found by divide and conquer strategy shown above which will help us pinpoint the exact type of the graph out of reduced possibilities  1. Write the result to the 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 half 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 program**

|  |
| --- |
| Below the divide and conquer technique is shown in the form of tree structure where at every step, we are dividing our problem at hand into half. Hence, recurrence relation can be written which will give us the time complexity of our program.  Therefore, recurrence relation will be T(n)= 2T(n/2) + k where k is the time required for combining results. Now solving this recurrence relation by substitution or tree method, we get that time complexity is **O(n).** |