Walchand College Of Engineering

***Department of Computer Science***

**Batch: T3**

**Assignment No. 3 (Part 1) Divide and conquer strategy**

**Name: Piyush Dawkhare**

**PRN: 23510056**

**Problem Statement 1**:

Implement an algorithm to find the maximum element in an array that is first strictly increasing and then strictly decreasing, with time complexity O(log n).

# Algorithms Used

* Binary Search (Divide and Conquer)

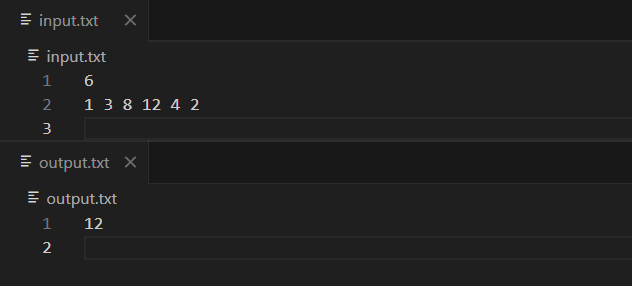
# Related Theory

* A bitonic array increases first, then decreases.
* The maximum element is the turning point.
* Binary Search can be modified to check middle element and decide whether to go left or right.
* Time Complexity: O(log n)

# Algorithm / Procedure

* Set low=0, high=n-1.
* While low ≤ high:
  + Compute mid.
  + If arr[mid] is greater than neighbors → return it.
  + If arr[mid] < arr[mid+1] → search right.
  + Else → search left.

# Output

****

**Problem Statement 2:**

Implement algorithm for Tiling problem: Given an n×n board where n is of form 2k where k

>= 1 (Basically n is a power of 2 with minimum value as 2). The board has one missing cell (of size 1 x 1). Fill the board using L shaped tiles. An L shaped tile is a 2 x 2 square with one cell of size 1×1 missing

# Algorithms Used

* Divide and Conquer Recursive Tiling

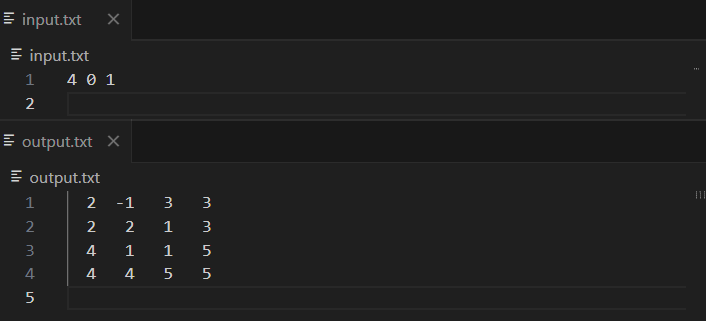
# Related Theory

* Recursive division into 4 quadrants.
* Place one L-shaped tile at the center covering the 3 quadrants that do not contain the missing square.
* Recurse on each quadrant.
* **Time Complexity:** O(n^2) (fills every square).

# Algorithm / Procedure

* Divide board into 4 quadrants.
* Place an L-tile in the center.
* Ensure each quadrant has exactly one missing cell.
* Recurse on each quadrant until size = 1.

# Output

****

**Problem Statement 3:**

Implement algorithm for The Skyline Problem: Given n rectangular buildings in a 2

dimensional city, computes the skyline of these buildings, eliminating hidden lines. The main task is to view buildings from a side and remove all sections that are not visible.

# Algorithms Used

* Divide and Conquer (Skyline Merge Algorithm)

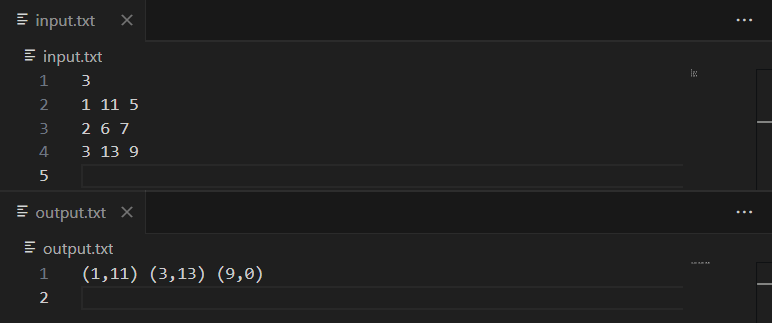
# Related Theory

* The skyline is the upper contour of overlapping rectangles.
* Similar to merging intervals but with heights considered.
* Divide buildings into halves, compute skylines recursively, then merge.
* Time Complexity: O(n log n)

# Algorithm / Procedure

* If one building, skyline = (left, ht) and (right, 0).
* Otherwise, split buildings into halves.
* Recursively compute skyline for left and right halves.
* Merge the two skylines by comparing heights at each x-coordinate.

**Output**

****