DAA Tutorial-3

Note: For each problem, you are required to implement the solution in C++, present the complete code, and demonstrate its execution to the evaluator.

Problem 1

Problem Definition: Given an array $A[1..n]$, count the number of inversions, i.e., pairs (i, j) such that $i < j$ and $A[i] > A[j]$.

Example: $A = [2, 4, 1, 3, 5]$ Inversions: $(2, 1), (4, 1), (4, 3)$ Total = 3.

Problem 2:

Problem Definition: Given n points in the plane, find the convex hull — the smallest convex polygon that contains all points.

Example: Points = $\{(0, 0), (1, 1), (2, 2), (0, 2), (2, 0)\}$ Convex Hull = polygon with vertices $(0, 0) \rightarrow (2, 0) \rightarrow (2, 2) \rightarrow (0, 2)$.

Problem 3:

Problem Definition: Given an unsorted array $A[1..n]$ and an integer k, find the k-th smallest element.

Example: $A = [7, 10, 4, 3, 20, 15]$, $k = 3$ Sorted = $[3, 4, 7, 10, 15, 20]$ Answer = 7.

Problem 4:

Problem Definition: Given an array A of n elements

$$A = [A[0], A[1], \ldots, A[n-1]],$$

find an index i $(0 \leq i < n)$ such that

$$A[i] \geq A[i-1] \quad \text{and} \quad A[i] \geq A[i+1],$$

where

- for the first element $(i = 0)$, we only require $A[0] \geq A[1]$,
- for the last element $(i = n - 1)$, we only require $A[n-1] \geq A[n-2]$.

Such an element $A[i]$ is called a **peak element**, and the index i is called a **peak index**.
Your task is to find and return any one peak index that satisfies the above condition.
Example: $A = [1, 3, 20, 4, 1, 0]$ Peak = index 2 (value 20).

Problem 5:

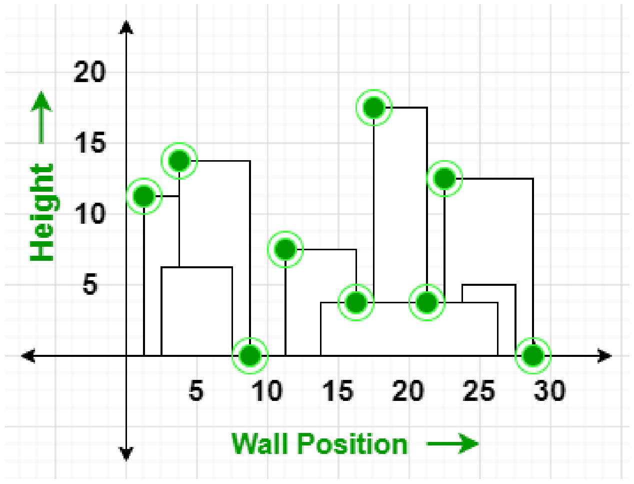
Problem Definition: Sort an array using Dual Pivot QuickSort. First, choose two pivots p and q such that $p < q$. The array is then partitioned into three segments: elements less than p, elements between p and q, and elements greater than q. Finally, each segment is recursively sorted to obtain the fully sorted array.

Example: $A = [4, 7, 2, 9, 1, 5]$ Pivots $= (2, 7)$ Partitions: $[1] \mid [4, 5] \mid [9]$ Sorted $= [1, 2, 4, 5, 7, 9]$.

Problem 6:

Problem Definition: Given n buildings represented by (L_i, R_i, H_i) where L_i and R_i are the left and right x-coordinates, and H_i is the height, output the skyline formed by these buildings.

Example: Input: build[][] = [[1, 5, 11], [2, 7, 6], [3, 9, 13], [12, 16, 7], [14, 25, 3], [19, 22, 18], [23, 29, 13], [24, 28, 4]]
Output: [[1, 11], [3, 13], [9, 0], [12, 7], [16, 3], [19, 18], [22, 3], [23, 13], [29, 0]]
Explanation: The skyline is formed based on the key-points (representing by “green” dots) eliminating hidden walls of the buildings. Note that no end point of the second building (2, 7, 6) is considered because it completely overlaps. Also, the points where the y coordinate changes are considered (Note that the top right of the third building (3, 9, 13) is not considered because it has same y coordinate.


Height
Wall Position