3.11 FINDING THE K-th SMALLEST ELEMENT USING MEDIAN OF MEDIANS

Question:

To Implement the Median of Medians algorithm ensures that you handle the worst-case time complexity efficiently while finding the k-th smallest element in an unsorted array.

AIM

To implement the Median of Medians algorithm for selecting the k-th smallest element in an unsorted array with guaranteed worst-case linear time complexity.

ALGORITHM

1. Divide the array into groups of 5 elements.
2. Sort each group and find the median.
3. Recursively apply the algorithm to find the median of medians.
4. Use this median as a pivot to partition the array into:
    - Elements less than pivot
    - Elements equal to pivot
    - Elements greater than pivot
5. Recurse into the appropriate partition based on the value of k.

PROGRAM

```
def partition(arr, pivot):
    lows = [el for el in arr if el < pivot]
    highs = [el for el in arr if el > pivot]
    pivots = [el for el in arr if el == pivot]
    return lows, pivots, highs

def select(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k - 1]

    medians = [sorted(arr[i:i+5])[len(arr[i:i+5])//2] for i in range(0, len(arr), 5)]
    pivot = select(medians, len(medians)//2 + 1)
    lows, pivots, highs = partition(arr, pivot)

    if k <= len(lows):
        return select(lows, k)
    elif k <= len(lows) + len(pivots):
        return pivot
    else:
        return select(highs, k - len(lows) - len(pivots))

def run_median_of_medians():
    arr = list(map(int, input("Enter array: ").split()))
    k = int(input("Enter k: "))
    print("K-th smallest element:", select(arr, k))

run_median_of_medians()
```

Input:

6,1,24,94,73,15,64 ,key=4

Output:

```
Enter array: 6 1 24 94 73 15 64
Enter k: 4
K-th smallest element: 24
>>>
```

RESULT:

Thus finding convex hull of a set of 2d points using the brute force approach is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

· Time Complexity: O(n)

· Space Complexity: O(n)