166) 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.

arr = [12, 3, 5, 7, 19] k = 2 Expected Output:5
arr = [12, 3, 5, 7, 4, 19, 26] k = 3 Expected Output:5
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] k = 6 Expected Output:6

aim: The worst-case time complexity efficiently while finding the k-th smallest element in an unsorted array

program:

```
def partition(arr, low, high, pivot):
    pivot_value = arr[pivot]
    arr[pivot], arr[high] = arr[high], arr[pivot]
    store_index = low
    for i in range(low, high):
        if arr[i] < pivot_value:
            arr[store_index], arr[i] = arr[i], arr[store_index]
            store_index += 1
    arr[store_index], arr[high] = arr[high], arr[store_index]
    return store_index


def select(arr, low, high, k):
    while True:
        if low == high:
            return arr[low]
        pivot_index = median_of_medians(arr, low, high)
        pivot_index = partition(arr, low, high, pivot_index)
        if k == pivot_index:
            return arr[k]
        elif k < pivot_index:
            high = pivot_index - 1
        else:
            low = pivot_index + 1
```

```
def median_of_medians(arr, low, high):
    n = high - low + 1
    if n < 10:
        return partition5(arr, low, high)

    for i in range((n + 4) // 5):
        sub_left = low + i * 5
        sub_right = min(sub_left + 4, high)
        median = partition5(arr, sub_left, sub_right)
        arr[low + i], arr[median] = arr[median], arr[low + i]

    mid = (n // 10) + low + 1
    return select(arr, low, low + (n // 5) - 1, mid)

def partition5(arr, low, high):
    sub_list = arr[low:high + 1]
    sub_list.sort()
    median_index = (high - low) // 2 + low
    arr[median_index] = sub_list[(high - low) // 2]
    return median_index

def kth_smallest(arr, k):
    return select(arr, 0, len(arr) - 1, k - 1)

arr1 = [12, 3, 5, 7, 19]
k1 = 2
print(f"Expected Output: 5, Actual Output: {kth_smallest(arr1, k1)}")
```

output:

```
Expected Output: 5, Actual Output: 7
```

TIMECOMPLEXITY: O(N)