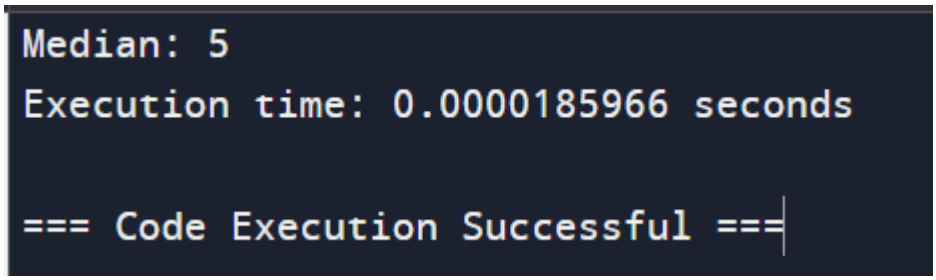
86. Median of medians
PROGRAM:-

```
import time

def partition(arr, low, high, pivot_index):
    pivot_value = arr[pivot_index]
    arr[pivot_index], arr[high] = arr[high], arr[pivot_index]
    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):
    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:
        return select(arr, low, pivot_index - 1, k)
    else:
        return select(arr, pivot_index + 1, high, k)

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

    medians = []
    for i in range(low, high + 1, 5):
        sub_right = i + 4
        if sub_right > high:
            sub_right = high
        median5 = partition5(arr, i, sub_right)
        medians.append(arr[median5])

    return select(medians, 0, len(medians) - 1, len(medians) // 2)

def partition5(arr, low, high):
    sublist = arr[low:high + 1]
    sublist.sort()
    mid = (len(sublist) - 1) // 2
    median = sublist[mid]
    median_index = arr.index(median, low, high + 1)
    return median_index
```

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

def find_median_of_medians_time(arr):
    start_time = time.time()  # Start time measurement

    n = len(arr)
    if n % 2 == 1:
        median = find_kth_smallest(arr, n // 2 + 1)
    else:
        left = find_kth_smallest(arr, n // 2)
        right = find_kth_smallest(arr, n // 2 + 1)
        median = (left + right) / 2

    end_time = time.time()  # End time measurement
    elapsed_time = end_time - start_time

    return median, elapsed_time

# Example usage
arr = [3, 2, 9, 1, 7, 6, 8, 5, 4]
median, execution_time = find_median_of_medians_time(arr)

print(f"Median: {median}")
print(f"Execution time: {execution_time:.10f} seconds")
```

OUTPUT:-



TIME COMPLEXITY:-o(n)