ASSIGNMENT-8(25-06-24)

1.**Median of Medians Algorithm**

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

2.median of medians:

```
def medianofmedians(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k]

    sublists = [arr[i:i+5] for i in range(0, len(arr), 5)]
    medians = [sorted(sublist)[len(sublist)//2] for sublist in sublists]

    pivot = medianofmedians(medians, len(medians)//2)

    pivotindex = arr.index(pivot)
    pivotindex = partition(arr, 0, len(arr) - 1, pivotindex)

    if k == pivotindex:
        return arr[k]
    elif k < pivotindex:
        return medianofmedians(arr[:pivotindex], k)
    else:
```

```
        return medianofmedians(arr[pivotindex + 1:], k - pivotindex - 1)


print(median_of_medians([12, 3, 5, 7, 19], 2))
```

Output: 5

2.**Closest Pair of Points (Divide and Conquer)**

```
import heapq
def kclosest(points, k):
    def distance(point):
        return point[0] ** 2 + point[1] ** 2


    return heapq.nsmallest(k, points, key=distance)


print(kclosest([[1, 3], [-2, 2], [5, 8], [0, 1]], 2))
```

Output: [[-2, 2], [0, 1]]

3.foursumcount

```
def foursumcount(A, B, C, D):
    ABsum = {}
    for a in A:
        for b in B:
            ABsum[a + b] = ABsum.get(a + b, 0) + 1


    count = 0
    for c in C:
        for d in D:
            count += ABsum.get(-(c + d), 0)


    return count


print(foursumcount([1, 2], [-2, -1], [-1, 2], [0, 2]))
```

Output: 2