Name: S.NedaAnjum
Reg no: 19231107 1
Subcode: CSA0672

(1) To implement the median algorithm ensures that ensures ~~that~~ you handle the worst-case time complexity efficiently while loop ~~be~~ finding the K^{th} smallest element in an unsorted array

(i) arr = [12,3,5,7,19], K = 2.

Given
arr = [12,3,5,7,19], K=2

Arrange the array in ascending order = $[\overset{0}{3},\overset{1}{5},\overset{2}{7},\overset{3}{12},\overset{4}{19}]$

$$\text{median} = \frac{\text{low} + \text{high}}{2} = \frac{0+4}{2} = 2$$

Median = 7

As given K=2, the value of (K=2) = 5

(ii) arr = [12,3,5,7,4,19,26], K=3

Given arr = [12,3,5,7,4,19,26] K=3

Arrange the array in ascending order = $[\overset{0}{12},\overset{1}{3},\overset{2}{5},\overset{3}{7},\overset{4}{4},\overset{5}{19},$

$$\text{median} = \frac{\text{low} + \text{high}}{2} = \frac{0+6}{2} = 3$$

Median = 7

As given K=2,

The value of (K=3) = 5

(iii) arr = [1,2,3,4,5,6,7,8,9,10] K=6

Given
arr = [1,2,3,4,5,6,7,8,9,10], K=6

Arrange the array in ascending order, it is already in arranged = $[\overset{0}{1},\overset{1}{2},\overset{2}{3},\overset{3}{4},\overset{4}{5},\overset{5}{6},\overset{6}{7},\overset{7}{8},\overset{8}{9},\overset{9}{10}]$

$$\text{median} = \frac{\text{low} + \text{high}}{2} = \frac{0+9}{2} = 4.5 \simeq 5.$$

median = 6

As, given $k=6$, The value $q(k=6) = 6$

2) To implement a function median of median (arr, k) that takes unsorted array arr and an integer K, and returns the k^{th} smallest element in the array.

arr = [1,2,3,4,5,6,7,8,9,10], $k=6$

Given

arr = [1,2,3,4,5,6,7,8,9,10], $k=6$

Arrange it in ascending order, but it is already sorted

0 1 2 3 4 5 6 7 8 9

[1,2,3,4,5,6,7,8,9,10]

$$\text{median} = \frac{0+9}{2} = 4.5 \approx 5$$

Median = 6

As given $k=6$, the value of $(k=6) = 6$.

(ii) arr = [23,17,31,44,55,21,20,18,19,27] $k=5$

Arrange the order in ascending order.

0 1 2 3 4 5 6 7 8 9

= [17,18,19,20,21,23,27,31,44,55]

$$\text{Median} = \frac{\text{low+high}}{2} = \frac{0+9}{2} = 4.5 \approx 5$$

$4.5 \approx 4$

As given $k=5$, the given value of $(k=5) = 21$.

3 Closest pair of points

1) Given an array of points where points $[i] = [x_i, y_i]$ represents point an the x-y plane and an integer k, return the k-closest pair to the origin (0,0).

As the arrangement of points should be done in a such a way that are close to origin

$\therefore k = 2$

$[3,3], [-2,4]$.

4) Given four lists A, B, C, D of integer values, write a program to compute how many tuples (i, j, k, l) There are such that A[i] + B[j] + C[k] + D[l] is zero.

A = [1,2], B = [-2,-1], C = [-1,2), D = [0,2]

```
from collections import default dict

def four lists (A,B,C,D):
    AB-four lists = default dict (int)
    for a in A:
      for b in B:
    AB-sum- counts [a+b]+=1
    count= 0
      for c in C:
        for d in D:
          complement= -(c+d)
    if complement in AB- sum- counts= count+=
                    AB_sum-counts [complement]

return count
```

A = [1,2]

B = [-3,-1]

C = [-1,2]

D = [0,2]

print (four-sum-count (A,B,C,D)).

(i) points = [[1,3], [-2,2], [5,8], [0,1]], k=2

Given

points = [[1,3], [-2,2], [5,8], [0,1]

distance = x^2+y^2

$[1,3] = 1^2+3^2 = 10$

$[-2,2] = (-2)^2+(2)^2 = 8$

$[5,8] = [5^2+8^2]$

$= 24+64$

$= 89$

$[0,1] = 0^2+1^2$

$= 1$

Distance = [8, 89, 10, 1]

Now arrange the points in that order, close to origin

[[0,1], [-2,2], [1,3], [5,8]]

At the value k=2

consider first 2 points, so the closest pair = [[0,1], [-2,2]]

(ii) points = [[1,3], [-2,2]], k=1

$[1,3] = [1]^2+[3]^2 = 10$

$[-2,2] = (-2)^2+(2)^2 = 4+4$

$= 8$

Distance = [10, 8]

Distance = x^2+y^2

Arrange the points in such orders that are close to the origin by considering distance = [[-2,2], [1,3]]

k=1

[-2,2]

(iii) points [[3,3], [5,-1], [-3,4]], k=2

Distance x^2+y^2

$[3,3] = [9+9] = 18$

$[5,1] = [25+1] = 26$

$[-2,4] = [4,16] = 20.$

ii) A=[0], B=[0], C=[0], D=[0]

```
for collections import default dict
def four-sum-count(A,B,C,D):
    AB-sum-counts = default dict(int)
        for a in A:
            for b in B:
            AB-sum-counts [a+b]+ =1
    count=0
for c in C:
    for d in D:
    complement =(c+d)
if complement in AB-sum-counts [complement]
return count
```