Y.Harika
192325023
CSA0669,,

ASSIGNMENT-5

1) To implement the Median of Medians algorithm ensures that you handle the worst-case time complexity efficiently while finding the kth smallest element in an unsorted array.

arr = [12, 3, 5, 7, 19] k = 2

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

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

i) arr = [12, 3, 5, 7, 19] = $[3, 5, \boxed{7}, 12, 19]$

Median = 7

Arrange the values less than 7 in leftside of '7' and greater than 7 in rightside.

∴ [12, 3, 5, 7, 19] = $[3, 5, \boxed{7}, 12, 19]$

∴ 2nd Smallest element = 5,,

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

Divide into sub arrays. $[\overset{0}{3}, \overset{1}{4}, \overset{2}{5}, \overset{3}{\boxed{7}}, \overset{4}{12}, \overset{5}{19}, \overset{6}{26}]$

A_1 = [12, 3, 5, 7, 4] and A_2 = [19, 26]

From A_1 Median is = 5

From A_2 Median is = 19

∴ Medians [5, 19] = 5,,

Partition around 5, Arrange the values less than 5 in leftside of '5' and greater than 5 in rightside,,

[3, 4, <u>5</u>, 7, 12, 19, 26]

= 5 is the 3rd smallest element

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

Divide into subarrays

A_1 = [1, 2, 3, 4, 5] ; A_2 = [6, 7, 8, 9, 10]

Median = 3 ; Median = 8.

Median of Medians = [3, 8]

∴ = 3

∴ Partition around 3, Arrange the elements than 3 are in leftside and greater ele right side.

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

∴ 6th smallest ele = 6

2) To implement a function median_of_medians(arr, k) that takes an unsorted array arr and an integer k and returns the kth smallest element an array.

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

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

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

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

0 1 2 3 4 5 6 7 8 9

Median = 6.

∴ k = 6 (6th smallest element).

2) arr = [23, 17, 31, 44, 55, 21, 20, 18, 19, 27] ; k = 5

0 1 2 3 4 5 6 7 8 9

∴ Mid = $\frac{0+9}{2}$ = 4.5 ≈ 5

∴ Median = 21

sorted

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

0 1 2 3 4 5 9

5 . Median = 23.

∴ k^{th} smallest element is = 21,,) (6th = k)

closest pair of points:

3) Given an array of points where points[T] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k-closest pair to the origin.

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

Given

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

Cistance = x^2+y^2

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

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

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

$= 25+64 = 89$

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

$= 1.$

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]]

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

Distance = x^2+y^2

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

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

Distance = [10, 8]

Arrange the points in such order 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

[-3,4] = [21, 16] = 20.

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

∴ 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 fourlists (A,B,C,D):
        AB - fourlists = default dict(int)
        for a in A:
            for b in B:
        AB - som - 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))
```

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

```
for collections import default dict.
    def four_sum_cant (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:
                count+ = AB_sum-counts [complement]
        return count.
```

= 5 is the 3rd smallest element.