DAY-6 PROGRAMS

1. To implement the median of medians algorithm ensure that you handle the worst case time complexity efficiency while finding the k-th smallest element in an unsorted array.

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

Expected output:5

Code:

import random

def median\_of\_medians(arr, k):

if len(arr) == 1:

return arr[0]

subarrays = [arr[i:i+5] for i in range(0, len(arr), 5)]

medians = [sorted(subarray)[len(subarray)//2] for subarray in subarrays]

if len(medians) <= 5:

pivot = sorted(medians)[len(medians)//2]

else:

pivot = median\_of\_medians(medians, len(medians)//2)

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

if k <= len(left):

return median\_of\_medians(left, k)

elif k <= len(left) + len(middle):

return middle[0]

else:

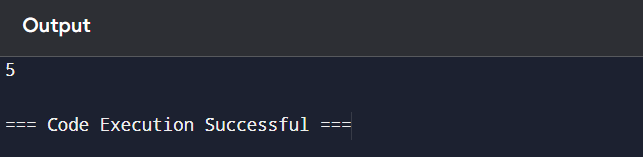
return median\_of\_medians(right, k - len(left) - len(middle))

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

k = 2

print(median\_of\_medians(arr, k))

output:



2. To implement a function median of medians (arr,k)that takes an unsorted array arr and an integer,and return the k-th smallest element in the array.

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

Output:An intrger representing the k-th smallest elements in the array.

Code:

import random

def median\_of\_medians(arr, k):

if len(arr) == 1:

return arr[0]

subarrays = [arr[i:i+5] for i in range(0, len(arr), 5)]

medians = [sorted(subarray)[len(subarray)//2] for subarray in subarrays]

if len(medians) <= 5:

pivot = sorted(medians)[len(medians)//2]

else:

pivot = median\_of\_medians(medians, len(medians)//2)

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

if k <= len(left):

return median\_of\_medians(left, k)

elif k <= len(left) + len(middle):

return middle[0]

else:

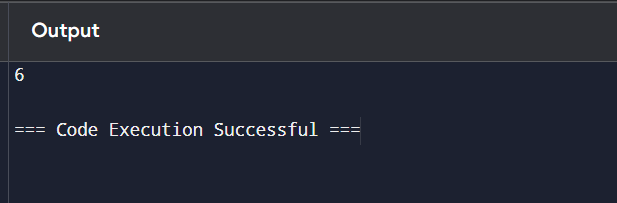
return median\_of\_medians(right, k - len(left) - len(middle))

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

k = 6

print(median\_of\_medians(arr, k))

output:



3. write a program to implement meet in the middle technique.give an array of integers and a target sum , find the subset whose sum is closest to the target. You will use the meet in the middle technique

To eifficiency find this subset.

Set=[45,34,4,12,5,2]

Target sum=42

Code:

from itertools import combinations

def meet\_in\_the\_middle(arr, target):

n = len(arr)

mid = n // 2

left\_half = arr[:mid]

right\_half = arr[mid:]

def get\_subset\_sums(arr):

subset\_sums = set()

for r in range(len(arr) + 1):

for combo in combinations(arr, r):

subset\_sums.add(sum(combo))

return sorted(subset\_sums)

left\_sums = get\_subset\_sums(left\_half)

right\_sums = get\_subset\_sums(right\_half)

closest\_sum = None

closest\_diff = float('inf')

for left\_sum in left\_sums:

target\_right\_sum = target - left\_sum

low, high = 0, len(right\_sums) - 1

while low <= high:

mid = (low + high) // 2

right\_sum = right\_sums[mid]

current\_sum = left\_sum + right\_sum

current\_diff = abs(current\_sum - target)

if current\_diff < closest\_diff:

closest\_diff = current\_diff

closest\_sum = current\_sum

if current\_sum < target:

low = mid + 1

else:

high = mid - 1

return closest\_sum

arr = [45, 34, 4, 12, 5, 2]

target = 42

closest\_sum = meet\_in\_the\_middle(arr, target)

print(f"The subset sum closest to {target} is {closest\_sum}.")

output:

