DESIGN AND ANALYSIS OF ALGORITHMS

NAME : MALREDDY.P

REGISTER NO: 192372015

DAY-5 PROGRAMMES

1.Write a Program to find both the maximum and minimum values in the array. Implement using any programming language of your choice. Execute your code and provide the maximum and minimum values found. Input : N= 8, a[] = {5,7,3,4,9,12,6,2} Output : Min = 2, Max = 12 Test Cases : Input : N= 9, a[] = {1,3,5,7,9,11,13,15,17} Output : Min = 1, Max = 17 Test Cases : Input : N= 10, a[] = {22,34,35,36,43,67, 12,13,15,17} Output : Min 12, Max 67.

```python
def find_min_max(arr):
    min_val = arr[0]
    max_val = arr[0]

    for num in arr:
        if num < min_val:
            min_val = num
        if num > max_val:
            max_val = num

    return min_val, max_val

test_cases = [
    (8, [5, 7, 3, 4, 9, 12, 6, 2]),
    (9, [1, 3, 5, 7, 9, 11, 13, 15, 17]),
    (10, [22, 34, 35, 36, 43, 67, 12, 13, 15, 17])
]

for N, arr in test_cases:
    min_val, max_val = find_min_max(arr)
    print(f"Input: N={N}, a[]={arr}")
    print(f"Output: Min = {min_val}, Max = {max_val}\n")
```

```
Input: N=8, a[]=[5, 7, 3, 4, 9, 12, 6, 2]
Output: Min = 2, Max = 12

Input: N=9, a[]=[1, 3, 5, 7, 9, 11, 13, 15, 17]
Output: Min = 1, Max = 17

Input: N=10, a[]=[22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
Output: Min = 12, Max = 67


=== Code Execution Successful ===
```

2.Consider an array of integers sorted in ascending order: 2,4,6,8,10,12,14,18. Write a Program to find both the maximum and minimum values in the array. Implement using any programming language of your choice. Execute your code and provide the maximum and minimum values found. Input : N=8, 2,4,6,8,10,12,14,18. Output : Min = 2, Max =18 Test Cases : Input : N= 9, a[] =

{11,13,15,17,19,21,23,35,37} Output : Min = 11, Max = 37 Test Cases : Input : N= 10, a[] = {22,34,35,36,43,67, 12,13,15,17} Output : Min 12, Max 67.

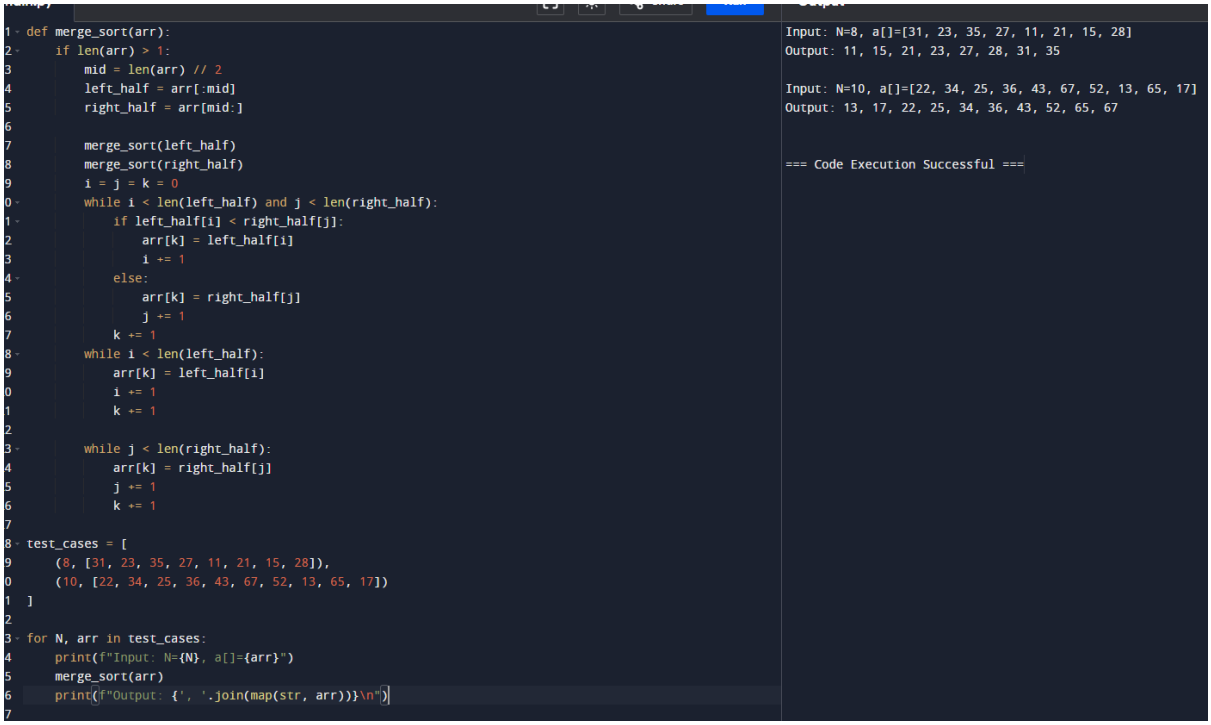
main.py

Share Run Output

```
def find_min_max(arr):
    min_val = arr[0]
    max_val = arr[-1]

    return min_val, max_val

test_cases = [
    (8, [2, 4, 6, 8, 10, 12, 14, 18]),
    (9, [11, 13, 15, 17, 19, 21, 23, 35, 37]),
    (10, [22, 34, 35, 36, 43, 67, 12, 13, 15, 17])
]

for N, arr in test_cases:
    min_val, max_val = find_min_max(arr)
    print(f"Input: N={N}, a[]={arr}")
    print(f"Output: Min = {min_val}, Max = {max_val}\n")
```

```
Input: N=8, a[]=[2, 4, 6, 8, 10, 12, 14, 18]
Output: Min = 2, Max = 18

Input: N=9, a[]=[11, 13, 15, 17, 19, 21, 23, 35, 37]
Output: Min = 11, Max = 37

Input: N=10, a[]=[22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
Output: Min = 22, Max = 17


=== Code Execution Successful ===
```

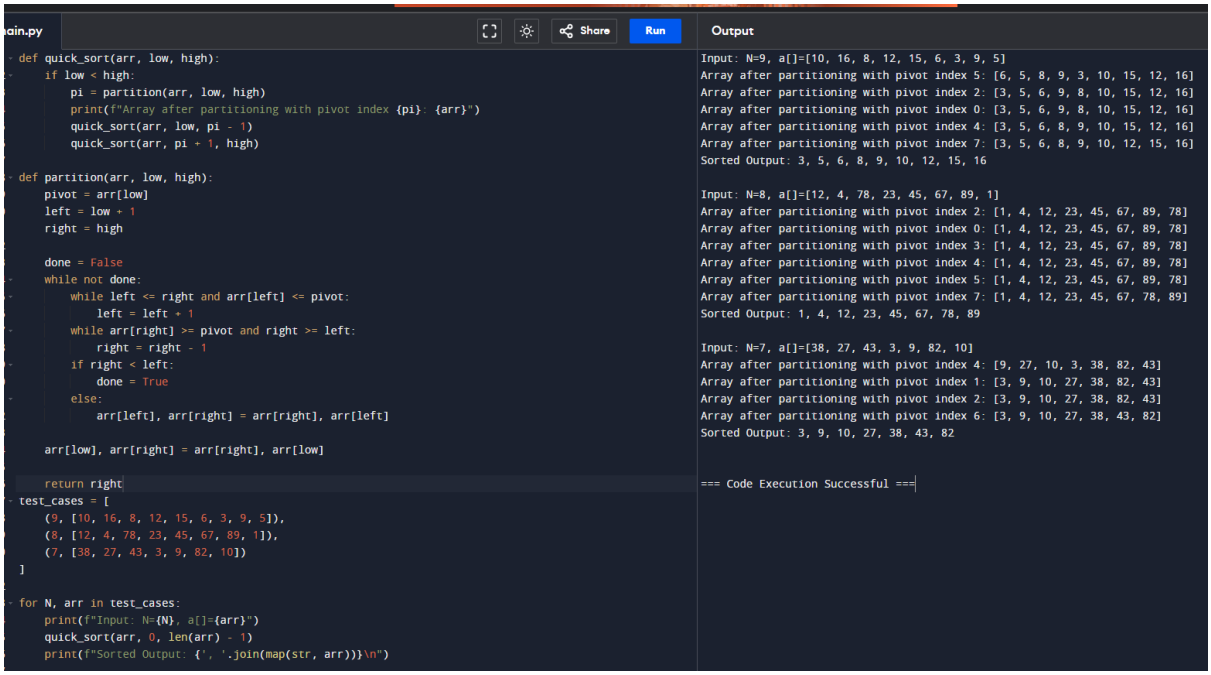
4.You are given an unsorted array 31,23,35,27,11,21,15,28. Write a program for Merge Sort and implement using any programming language of your choice. Test Cases : Input : N= 8, a[] = {31,23,35,27,11,21,15,28} Output : 11,15,21,23,27,28,31,35 Test Cases : Input : N= 10, a[] = {22,34,25,36,43,67, 52,13,65,17} Output : 13,17,22,25,34,36,43,52,65,67.



```python
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]

        merge_sort(left_half)
        merge_sort(right_half)
        i = j = k = 0
        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1
        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1

test_cases = [
    (8, [31, 23, 35, 27, 11, 21, 15, 28]),
    (10, [22, 34, 25, 36, 43, 67, 52, 13, 65, 17])
]

for N, arr in test_cases:
    print(f"Input: N={N}, a[]={arr}")
    merge_sort(arr)
    print(f"Output: {', '.join(map(str, arr))}\n")
```

```
Input: N=8, a[]=[31, 23, 35, 27, 11, 21, 15, 28]
Output: 11, 15, 21, 23, 27, 28, 31, 35

Input: N=10, a[]=[22, 34, 25, 36, 43, 67, 52, 13, 65, 17]
Output: 13, 17, 22, 25, 34, 36, 43, 52, 65, 67


=== Code Execution Successful ===
```

5.Given an unsorted array 10,16,8,12,15,6,3,9,5 Write a program to perform Quick Sort. Choose the first element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display the array after each recursive call until the entire array is sorted. Input : N= 9, a[]= {10,16,8,12,15,6,3,9,5} Output : 3,5,6,8,9,10,12,15,16 Test Cases : Input : N= 8, a[] = {12,4,78,23,45,67,89,1} Output : 1,4,12,23,45,67,78,89 Test Cases : Input : N= 7, a[] = {38,27,43,3,9,82,10} Output : 3,9,10,27,38,43,82,



6. Implement the Quick Sort algorithm in a programming language of your choice and test it on the array 19,72,35,46,58,91,22,31. Choose the middle element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display the array after each recursive call until the entire array is sorted. Execute your code and show the sorted array. Input : N= 8, a[] = {19,72,35,46,58,91,22,31} Output : 19,22,31,35,46,58,72,91 Test Cases : Input : N= 8, a[] = {31,23,35,27,11,21,15,28} Output : 11,15,21,23,27,28,31,35 Test Cases : Input : N= 10, a[] = {22,34,25,36,43,67, 52,13,65,17} Output : 13,17,22,25,34,36,43,52,65,67

main.py

```python
def quick_sort(arr, low, high):
    if low < high:
        pi = partition(arr, low, high)
        print(f"Array after partitioning with pivot index {pi}: {arr}")
        quick_sort(arr, low, pi - 1)
        quick_sort(arr, pi + 1, high)

def partition(arr, low, high):
    mid = (low + high) // 2
    pivot = arr[mid]
    arr[mid], arr[high] = arr[high], arr[mid]
    left = low
    right = high - 1
    while left <= right:
        while left <= right and arr[left] < pivot:
            left += 1
        while left <= right and arr[right] >= pivot:
            right -= 1
        if left <= right:
            arr[left], arr[right] = arr[right], arr[left]
            left += 1
            right -= 1
    arr[left], arr[high] = arr[high], arr[left]
    return left

test_cases = [
    (8, [19, 72, 35, 46, 58, 91, 22, 31]),
    (8, [31, 23, 35, 27, 11, 21, 15, 28]),
    (10, [22, 34, 25, 36, 43, 67, 52, 13, 65, 17])
]
for N, arr in test_cases:
    print(f"Input: N={N}, a[]={arr}")
    quick_sort(arr, 0, len(arr) - 1)
    print(f"Sorted Output: {', '.join(map(str, arr))}\n")
```

Output

```
Input: N=8, a[]=[19, 72, 35, 46, 58, 91, 22, 31]
Array after partitioning with pivot index 4: [19, 22, 35, 31, 46, 91, 72, 58]
Array after partitioning with pivot index 1: [19, 22, 35, 31, 46, 91, 72, 58]
Array after partitioning with pivot index 3: [19, 22, 31, 35, 46, 91, 72, 58]
Array after partitioning with pivot index 6: [19, 22, 31, 35, 46, 58, 72, 91]
Sorted Output: 19, 22, 31, 35, 46, 58, 72, 91

Input: N=8, a[]=[31, 23, 35, 27, 11, 21, 15, 28]
Array after partitioning with pivot index 4: [15, 23, 21, 11, 27, 35, 31, 28]
Array after partitioning with pivot index 3: [15, 11, 21, 23, 27, 35, 31, 28]
Array after partitioning with pivot index 0: [11, 21, 15, 23, 27, 35, 31, 28]
Array after partitioning with pivot index 2: [11, 15, 21, 23, 27, 35, 31, 28]
Array after partitioning with pivot index 6: [11, 15, 21, 23, 27, 28, 31, 35]
Sorted Output: 11, 15, 21, 23, 27, 28, 31, 35

Input: N=10, a[]=[22, 34, 25, 36, 43, 67, 52, 13, 65, 17]
Array after partitioning with pivot index 6: [22, 34, 25, 36, 17, 13, 43, 67, 65, 52]
Array after partitioning with pivot index 3: [22, 17, 13, 25, 34, 36, 43, 67, 65, 52]
Array after partitioning with pivot index 1: [13, 17, 22, 25, 34, 36, 43, 67, 65, 52]
Array after partitioning with pivot index 4: [13, 17, 22, 25, 34, 36, 43, 67, 65, 52]
Array after partitioning with pivot index 8: [13, 17, 22, 25, 34, 36, 43, 52, 65, 67]
Sorted Output: 13, 17, 22, 25, 34, 36, 43, 52, 65, 67


=== Code Execution Successful ===
```

7.Implement the Binary Search algorithm in a programming language of your choice and test it on the array 5,10,15,20,25,30,35,40,45 to find the position of the element 20. Execute your code and provide the index of the element 20. Modify your implementation to count the number of comparisons made during the search process. Print this count along with the result. Input : N= 9, a[] = {5,10,15,20,25,30,35,40,45}, search key = 20 Output : 4 Test cases Input : N= 6, a[] = {10,20,30,40,50,60}, search key = 50 Output : 5 Input : N= 7, a[] = {21,32,40,54,65,76,87}, search key = 32 Output : 2

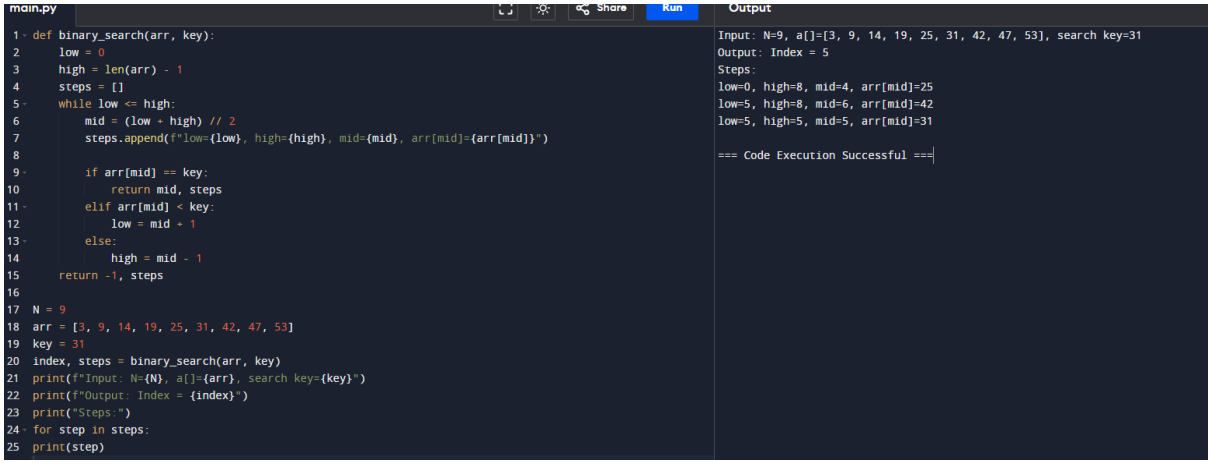
main.py

```python
def binary_search(arr, key):
    low = 0
    high = len(arr) - 1
    comparisons = 0
    while low <= high:
        mid = (low + high) // 2
        comparisons += 1
        if arr[mid] == key:
            return mid, comparisons
        elif arr[mid] < key:
            low = mid + 1
        else:
            high = mid - 1
    return -1, comparisons
test_cases = [
    (9, [5, 10, 15, 20, 25, 30, 35, 40, 45], 20),
    (6, [10, 20, 30, 40, 50, 60], 50),
    (7, [21, 32, 40, 54, 65, 76, 87], 32)
]
for N, arr, key in test_cases:
    index, comparisons = binary_search(arr, key)
    print(f"Input: N={N}, a[]={arr}, search key={key}")
    print(f"Output: Index = {index}, Comparisons = {comparisons}\n")
```

Output

```
Input: N=9, a[]=[5, 10, 15, 20, 25, 30, 35, 40, 45], search key=20
Output: Index = 3, Comparisons = 4

Input: N=6, a[]=[10, 20, 30, 40, 50, 60], search key=50
Output: Index = 4, Comparisons = 2

Input: N=7, a[]=[21, 32, 40, 54, 65, 76, 87], search key=32
Output: Index = 1, Comparisons = 2


=== Code Execution Successful ===
```

8. . You are given a sorted array 3,9,14,19,25,31,42,47,53 and asked to find the position of the element 31 using Binary Search. Show the mid-point calculations and the steps involved in finding the element. Display, what would happen if the array was not sorted, how would this impact the performance and correctness of the Binary Search algorithm? Input : N= 9, a[] = {3,9,14,19,25,31,42,47,53}, search key = 31 Output : 6 Test cases Input : N= 7, a[] = {13,19,24,29,35,41,42}, search key = 42.



9. Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k closest points to the origin (0, 0). (i) Input : points = [[1,3],[-2,2],[5,8],[0,1]],k=2 Output:[[-2, 2], [0, 1]] (ii) Input: points = [[1, 3], [-2, 2]], k = 1 Output: [[-2, 2]] (iii) Input: points = [[3, 3], [5, -1], [-2, 4]], k = 2 Output: [[3, 3], [-2, 4]].

```python
import heapq
def k_closest_points(points, k):
    def distance(point):
        return point[0]**2 + point[1]**2
    max_heap = []
    for point in points:
        dist = distance(point)
        heapq.heappush(max_heap, (-dist, point))
        if len(max_heap) > k:
            heapq.heappop(max_heap)
    return [point for _, point in max_heap]
test_cases = [
    ([[1, 3], [-2, 2], [5, 8], [0, 1]], 2),
    ([[1, 3], [-2, 2]], 1),
    ([[3, 3], [5, -1], [-2, 4]], 2)
]

for points, k in test_cases:
    result = k_closest_points(points, k)
    print(f"Points: {points}, k = {k}")
    print(f"Output: {result}\n")
```

```
Points: [[1, 3], [-2, 2], [5, 8], [0, 1]], k = 2
Output: [[-2, 2], [0, 1]]

Points: [[1, 3], [-2, 2]], k = 1
Output: [[-2, 2]]

Points: [[3, 3], [5, -1], [-2, 4]], k = 2
Output: [[-2, 4], [3, 3]]


=== Code Execution Successful ===
```

10. Given four lists A, B, C, D of integer values,Write a program to compute how many tuples n(i, j, k, l) there are such that A[i] + B[j] + C[k] + D[l] is zero. (i) Input: A = [1, 2], B = [-2, -1], C = [-1, 2], D = [0, 2] Output: 2 (ii) Input: A = [0], B = [0], C = [0], D = [0] Output:1.

main.py | Share | Run | Output

```python
from collections import defaultdict
def count_tuples(A, B, C, D):
    sum_map = defaultdict(int)

    for a in A:
        for b in B:
            sum_map[a + b] += 1
    count = 0
    for c in C:
        for d in D:
            target_sum = -(c + d)
            if target_sum in sum_map:
                count += sum_map[target_sum]

    return count
test_cases = [
    ([1, 2], [-2, -1], [-1, 2], [0, 2]),
    ([0], [0], [0], [0])
]
for A, B, C, D in test_cases:
    result = count_tuples(A, B, C, D)
    print(f"A = {A}, B = {B}, C = {C}, D = {D}")
    print(f"Output: {result}\n")
```

```
A = [1, 2], B = [-2, -1], C = [-1, 2], D = [0, 2]
Output: 2

A = [0], B = [0], C = [0], D = [0]
Output: 1


=== Code Execution Successful ===
```