main.py

Share Run

Output Clear

```python
def find_min_max(arr):
    return min(arr), max(arr)
test_cases = [
    [5, 7, 3, 4, 9, 12, 6, 2],
    [1, 3, 5, 7, 9, 11, 13, 15, 17],
    [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
]
for arr in test_cases:
    min_val, max_val = find_min_max(arr)
    print("Input:", arr)
    print("Output: Min =", min_val, ", Max =", max_val)

```

```
Input: [5, 7, 3, 4, 9, 12, 6, 2]
Output: Min = 2 , Max = 12
Input: [1, 3, 5, 7, 9, 11, 13, 15, 17]
Output: Min = 1 , Max = 17
Input: [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
Output: Min = 12 , Max = 67

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

main.py

Share Run

```python
def find_min_max(arr):
    return arr[0], arr[-1]
test_cases = [
    [2, 4, 6, 8, 10, 12, 14, 18],
    [11, 13, 15, 17, 19, 21, 23, 35, 37],
    [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
]
for arr in test_cases:
    min_val, max_val = find_min_max(arr)
    print("Input:", arr)
    print("Output: Min =", min_val, ", Max =", max_val)

```

Output Clear

```
Input: [2, 4, 6, 8, 10, 12, 14, 18]
Output: Min = 2 , Max = 18
Input: [11, 13, 15, 17, 19, 21, 23, 35, 37]
Output: Min = 11 , Max = 37
Input: [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
Output: Min = 22 , Max = 17

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

main.py

Share Run

Output Clear

```python
def k_closest(points, k):
    points.sort(key=lambda x: x[0]**2 + x[1]**2)
    return points[:k]
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, k)
    print("Closest points:", result)

```

```
Closest points: [[0, 1], [-2, 2]]
Closest points: [[-2, 2]]
Closest points: [[3, 3], [-2, 4]]

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

main.py

Share Run

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    pivot = arr[mid]
    less = [x for i, x in enumerate(arr) if x < pivot and i !=
        mid]
    greater = [x for i, x in enumerate(arr) if x > pivot and i !=
        mid]
    sorted_arr = less + [pivot] + greater
    print("After partitioning:", sorted_arr)
    return quick_sort(less) + [pivot] + quick_sort(greater)
for arr in [[19, 72, 35, 46, 58, 91, 22, 31],
            [31, 23, 35, 27, 11, 21, 15, 28],
            [22, 34, 25, 36, 43, 67, 52, 13, 65, 17]]:
    print("Initial array:", arr)
    sorted_arr = quick_sort(arr)
    print("Sorted:", sorted_arr)
```

Output

Clear

```
Initial array: [19, 72, 35, 46, 58, 91, 22, 31]
After partitioning: [19, 35, 46, 22, 31, 58, 72, 91]
After partitioning: [19, 35, 22, 31, 46]
After partitioning: [19, 22, 35, 31]
After partitioning: [31, 35]
After partitioning: [72, 91]
Sorted: [19, 22, 31, 35, 46, 58, 72, 91]
Initial array: [31, 23, 35, 27, 11, 21, 15, 28]
After partitioning: [11, 31, 23, 35, 27, 21, 15, 28]
After partitioning: [23, 21, 15, 27, 31, 35, 28]
After partitioning: [15, 21, 23]
After partitioning: [31, 28, 35]
After partitioning: [28, 31]
Sorted: [11, 15, 21, 23, 27, 28, 31, 35]
Initial array: [22, 34, 25, 36, 43, 67, 52, 13, 65, 17]
After partitioning: [22, 34, 25, 36, 43, 52, 13, 65, 17, 67]
After partitioning: [22, 34, 25, 36, 13, 17, 43, 52, 65]
After partitioning: [22, 34, 25, 13, 17, 36]
After partitioning: [22, 13, 17, 25, 34]
After partitioning: [13, 22, 17]
```

main.py

Share Run

```python
def four_sum_count(A, B, C, D):
    ab_sums = {}
    for a in A:
        for b in B:
            ab_sums[a + b] = ab_sums.get(a + b, 0) + 1
    count = 0
    for c in C:
        for d in D:
            count += ab_sums.get(-(c + d), 0)
    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 = four_sum_count(A, B, C, D)
    print("Output:", result)
```

Output

Clear

```
Output: 2
Output: 1

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

main.py

Share Run

Output Clear

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr, 0
    mid = len(arr) // 2
    left, left_count = merge_sort(arr[:mid])
    right, right_count = merge_sort(arr[mid:])
    merged, count = [], left_count + right_count
    i = j = 0
    while i < len(left) and j < len(right):
        count += 1
        merged.append(left[i] if left[i] < right[j] else right[j]
            )
        if left[i] < right[j]: i += 1
        else: j += 1
    return merged + left[i:] + right[j:], count
for arr in [[12,4,78,23,45,67,89,1],[38, 27, 43, 3, 9, 82, 10]]:
    sorted_arr, comparisons = merge_sort(arr)
    print("Sorted:", sorted_arr, "Comparisons:", comparisons)
```

```
Sorted: [1, 4, 12, 23, 45, 67, 78, 89] Comparisons: 16
Sorted: [3, 9, 10, 27, 38, 43, 82] Comparisons: 13

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

main.py

Share Run

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[0]
    less = [x for x in arr[1:] if x <= pivot]
    greater = [x for x in arr[1:] if x > pivot]
    sorted_arr = less + [pivot] + greater
    print(sorted_arr)
    return quick_sort(less) + [pivot] + quick_sort(greater)
for arr in [[10, 16, 8, 12, 15, 6, 3, 9, 5],
            [12, 4, 78, 23, 45, 67, 89, 1],
            [38, 27, 43, 3, 9, 82, 10]]:
    print("Initial array:", arr)
    sorted_arr = quick_sort(arr)
    print("Sorted:", sorted_arr)

```

Output Clear

```
Initial array: [10, 16, 8, 12, 15, 6, 3, 9, 5]
[8, 6, 3, 9, 5, 10, 16, 12, 15]
[6, 3, 5, 8, 9]
[3, 5, 6]
[3, 5]
[12, 15, 16]
[12, 15]
Sorted: [3, 5, 6, 8, 9, 10, 12, 15, 16]
Initial array: [12, 4, 78, 23, 45, 67, 89, 1]
[4, 1, 12, 78, 23, 45, 67, 89]
[1, 4]
[23, 45, 67, 78, 89]
[23, 45, 67]
[45, 67]
Sorted: [1, 4, 12, 23, 45, 67, 78, 89]
Initial array: [38, 27, 43, 3, 9, 82, 10]
[27, 3, 9, 10, 38, 43, 82]
[3, 9, 10, 27]
[3, 9, 10]
[9, 10]
```

main.py

Share Run

```python
def binary_search(arr, key):
    low, high, comparisons = 0, len(arr) - 1, 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
arrays = [
    ([5, 10, 15, 20, 25, 30, 35, 40, 45], 20),
    ([10, 20, 30, 40, 50, 60], 50),
    ([21, 32, 40, 54, 65, 76, 87], 32)]
for arr, key in arrays:
    index, comparisons = binary_search(arr, key)
    print("Key:",key,"Index:",index,"Comparisons:",comparisons)
```

Output Clear

```
Key: 20 Index: 3 Comparisons: 4
Key: 50 Index: 4 Comparisons: 2
Key: 32 Index: 1 Comparisons: 2

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

main.py

Share Run

```python
def binary_search(arr, key):
    low, high, comparisons = 0, len(arr) - 1, 0
    while low <= high:
        mid = (low + high) // 2
        comparisons += 1
        print("Checking middle index", mid, ":", arr[mid])
        if arr[mid] == key:
            return mid, comparisons
        low, high = (mid + 1, high) if arr[mid] < key else (low,
            mid - 1)
    return -1, comparisons
arrays = [([3, 9, 14, 19, 25, 31, 42, 47, 53], 31),
          ([13, 19, 24, 29, 35, 41, 42], 42),
          ([20, 40, 60, 80, 100, 120], 60)]
for arr, key in arrays:
    index, comparisons = binary_search(arr, key)
    print("Key", key, "found at index", index, "with",
        comparisons, "comparisons.\n")
```

Output Clear

```
Checking middle index 4 : 25
Checking middle index 6 : 42
Checking middle index 5 : 31
Key 31 found at index 5 with 3 comparisons.

Checking middle index 3 : 29
Checking middle index 5 : 41
Checking middle index 6 : 42
Key 42 found at index 6 with 3 comparisons.

Checking middle index 2 : 60
Key 60 found at index 2 with 1 comparisons.


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

main.py

Share Run

```python
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        merge_sort(arr[:mid])
        merge_sort(arr[mid:])
        i = j = k = 0
        while i < len(arr[:mid]) and j < len(arr[mid:]):
            if arr[i] < arr[mid + j]: arr[k] = arr[i]; i += 1
            else: arr[k] = arr[mid + j]; j += 1
            k += 1
        while i < len(arr[:mid]): arr[k] = arr[i]; i += 1; k += 1
        while j < len(arr[mid:]): arr[k] = arr[mid + j]; j += 1;
            k += 1
for arr in [[31, 23, 35, 27, 11, 21, 15, 28], [22, 34, 25, 36, 43
    , 67, 52, 13, 65, 17]]:
    merge_sort(arr)
    print(arr)

```

Output

Clear

```
[11, 11, 11, 11, 11, 21, 15, 28]
[22, 34, 25, 36, 43, 67, 52, 13, 65, 17]

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