



main.py



Share

Run

Output

Clear

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

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

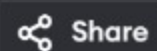
Input: [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]

Output: Min = 12 , Max = 67

=== Code Execution Successful ===



main.py



Run

Output

Clear

```
1 def find_min_max(arr):
2     return arr[0], arr[-1]
3
4 test_cases = [
5     [2, 4, 6, 8, 10, 12, 14, 18],
6     [11, 13, 15, 17, 19, 21, 23, 35, 37],
7     [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]
8 ]
9
10 for arr in test_cases:
11     min_val, max_val = find_min_max(arr)
12     print("Input:", arr)
13     print("Output: Min =", min_val, ", Max =", max_val)
```

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

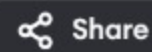
Input: [22, 34, 35, 36, 43, 67, 12, 13, 15, 17]

Output: Min = 22 , Max = 17

=== Code Execution Successful ===



main.py



Run

Output

Clear

```
1 def k_closest(points, k):
2     points.sort(key=lambda x: x[0]**2 + x[1]**2)
3     return points[:k]
4 test_cases = [
5     ([[1, 3], [-2, 2], [5, 8], [0, 1]], 2),
6     ([[1, 3], [-2, 2]], 1),
7     ([[3, 3], [5, -1], [-2, 4]], 2)
8 ]
9 for points, k in test_cases:
10     result = k_closest(points, k)
11     print("Closest points:", result)
12
```

Closest points: [[0, 1], [-2, 2]]

Closest points: [[-2, 2]]

Closest points: [[3, 3], [-2, 4]]

=== Code Execution Successful ===



main.py



Share

Run

Output

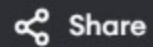
Clear

```
1 def quick_sort(arr):
2     if len(arr) <= 1:
3         return arr
4     mid = len(arr) // 2
5     pivot = arr[mid]
6     less = [x for i, x in enumerate(arr) if x < pivot and i !=
7             mid]
8     greater = [x for i, x in enumerate(arr) if x > pivot and i !=
9               mid]
10    sorted_arr = less + [pivot] + greater
11    print("After partitioning:", sorted_arr)
12    return quick_sort(less) + [pivot] + quick_sort(greater)
13 for arr in [[19, 72, 35, 46, 58, 91, 22, 31],
14             [31, 23, 35, 27, 11, 21, 15, 28],
15             [22, 34, 25, 36, 43, 67, 52, 13, 65, 17]]:
16     print("Initial array:", arr)
17     sorted_arr = quick_sort(arr)
18     print("Sorted:", sorted_arr)
```

```
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



Run

Output

Clear

```
1 def four_sum_count(A, B, C, D):
2     ab_sums = {}
3     for a in A:
4         for b in B:
5             ab_sums[a + b] = ab_sums.get(a + b, 0) + 1
6     count = 0
7     for c in C:
8         for d in D:
9             count += ab_sums.get(-(c + d), 0)
10    return count
11 test_cases = [
12     ([1, 2], [-2, -1], [-1, 2], [0, 2]),
13     ([0], [0], [0], [0])
14 ]
15 for A, B, C, D in test_cases:
16     result = four_sum_count(A, B, C, D)
17     print("Output:", result)
```

Output: 2

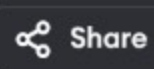
Output: 1

=== Code Execution Successful ===





main.py



Share

Run

Output

Clear

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

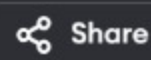
Sorted: [1, 4, 12, 23, 45, 67, 78, 89] Comparisons: 16

Sorted: [3, 9, 10, 27, 38, 43, 82] Comparisons: 13

=== Code Execution Successful ===



main.py



Run

Output

Clear



JS

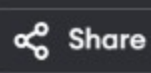
GO

```
1 def quick_sort(arr):
2     if len(arr) <= 1:
3         return arr
4     pivot = arr[0]
5     less = [x for x in arr[1:] if x <= pivot]
6     greater = [x for x in arr[1:] if x > pivot]
7     sorted_arr = less + [pivot] + greater
8     print(sorted_arr)
9     return quick_sort(less) + [pivot] + quick_sort(greater)
10 for arr in [[10, 16, 8, 12, 15, 6, 3, 9, 5],
11             [12, 4, 78, 23, 45, 67, 89, 1],
12             [38, 27, 43, 3, 9, 82, 10]]:
13     print("Initial array:", arr)
14     sorted_arr = quick_sort(arr)
15     print("Sorted:", sorted_arr)
16
```

```
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

Output

Clear

```
1 def binary_search(arr, key):
2     low, high, comparisons = 0, len(arr) - 1, 0
3     while low <= high:
4         mid = (low + high) // 2
5         comparisons += 1
6         if arr[mid] == key:
7             return mid, comparisons
8         elif arr[mid] < key:
9             low = mid + 1
10        else:
11            high = mid - 1
12    return -1, comparisons
13 arrays = [
14     ([5, 10, 15, 20, 25, 30, 35, 40, 45], 20),
15     ([10, 20, 30, 40, 50, 60], 50),
16     ([21, 32, 40, 54, 65, 76, 87], 32)]
17 for arr, key in arrays:
18     index, comparisons = binary_search(arr, key)
19     print("Key:",key,"Index:",index,"Comparisons:",comparisons)
```

Key: 20 Index: 3 Comparisons: 4

Key: 50 Index: 4 Comparisons: 2

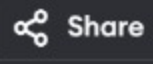
Key: 32 Index: 1 Comparisons: 2

=== Code Execution Successful ===





main.py



Run

Output

Clear



JS



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

```
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

Output

Clear



JS

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

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

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