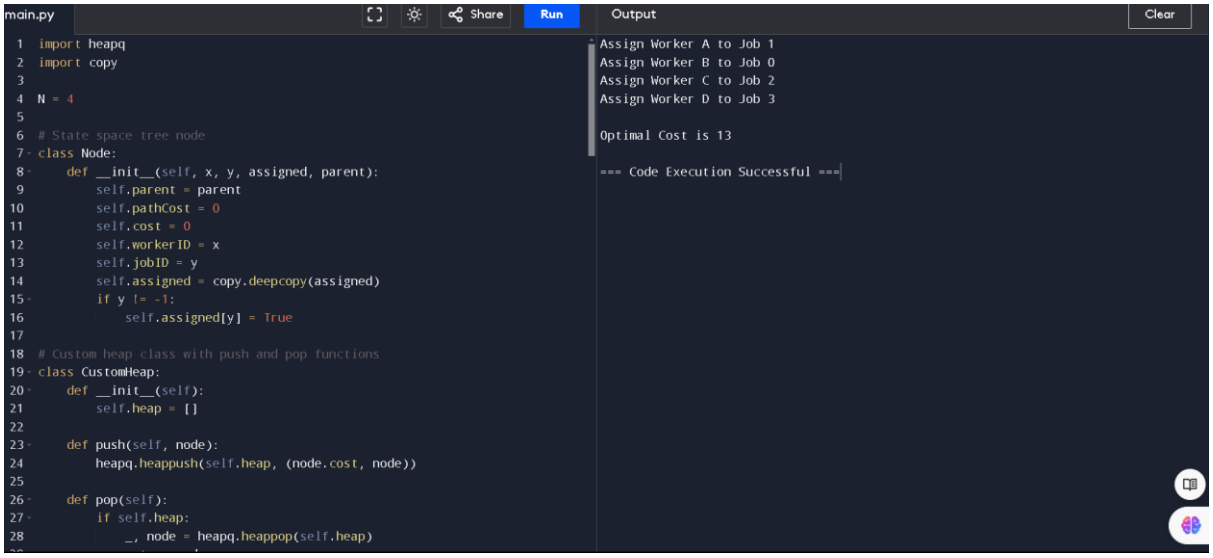
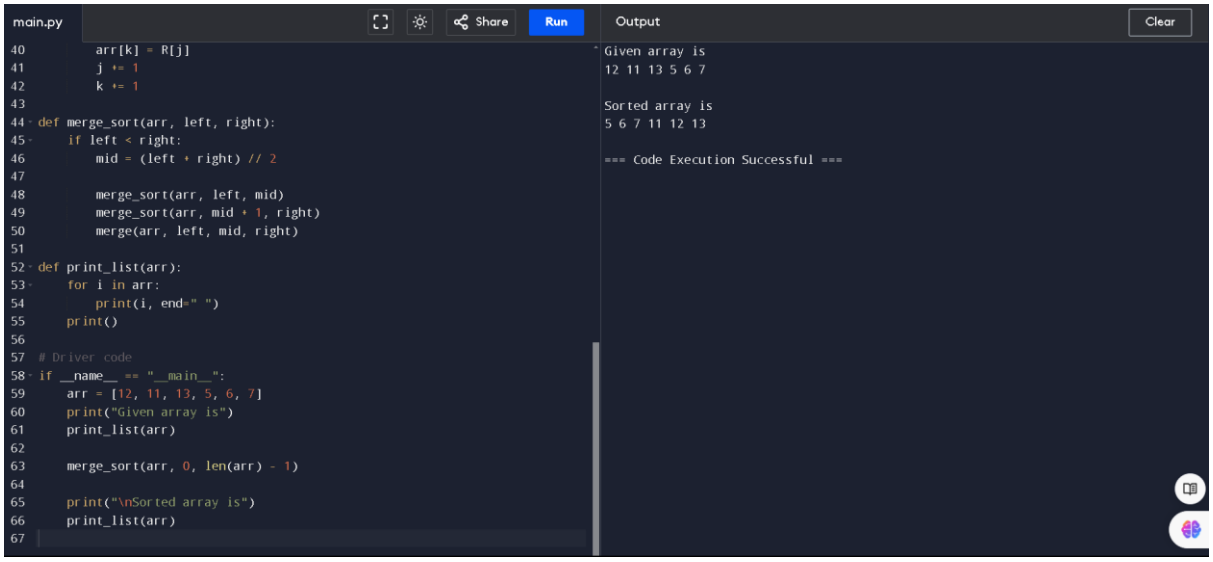
1. Write a Program to find both the maximum and minimum values in the array. Implement using any programming language of your choice. Execute your code and provide the maximum and minimum values found. Input : N= 8, a[] = {5,7,3,4,9,12,6,2} Output : Min = 2, Max = 12 Test Cases : Input : N= 9, a[] =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

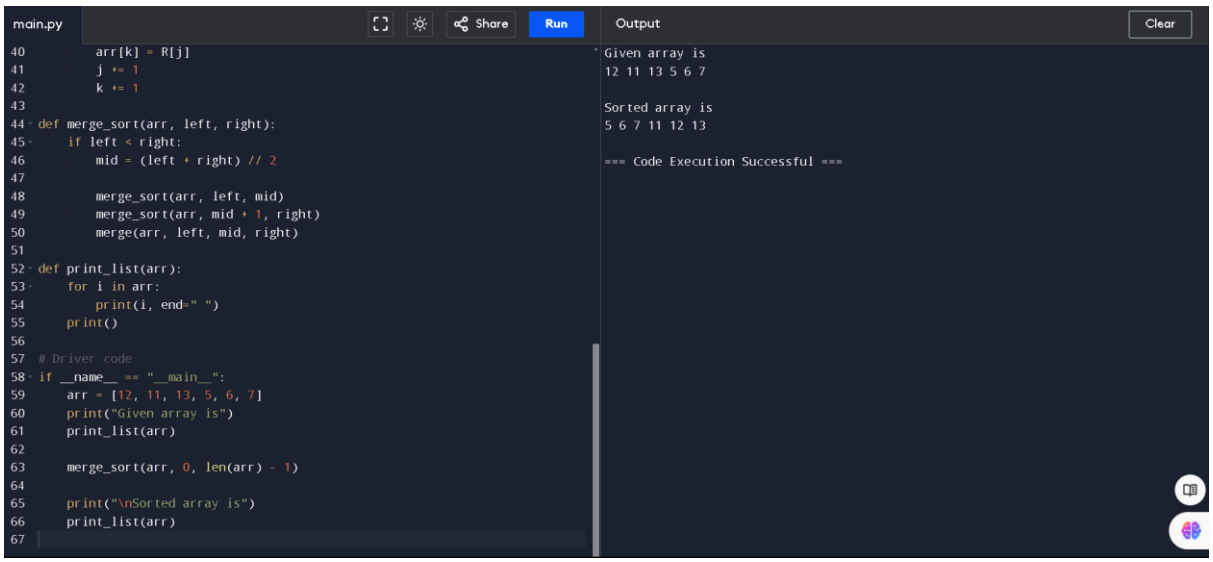


2.Consider an array of integers sorted in ascending order: 2,4,6,8,10,12,14,18. Write a Program to find both the maximum and minimum values in the array. Implement using any programming language of your choice. Execute your code and provide the maximum and minimum values found. Input : N=8, 2,4,6,8,10,12,14,18.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



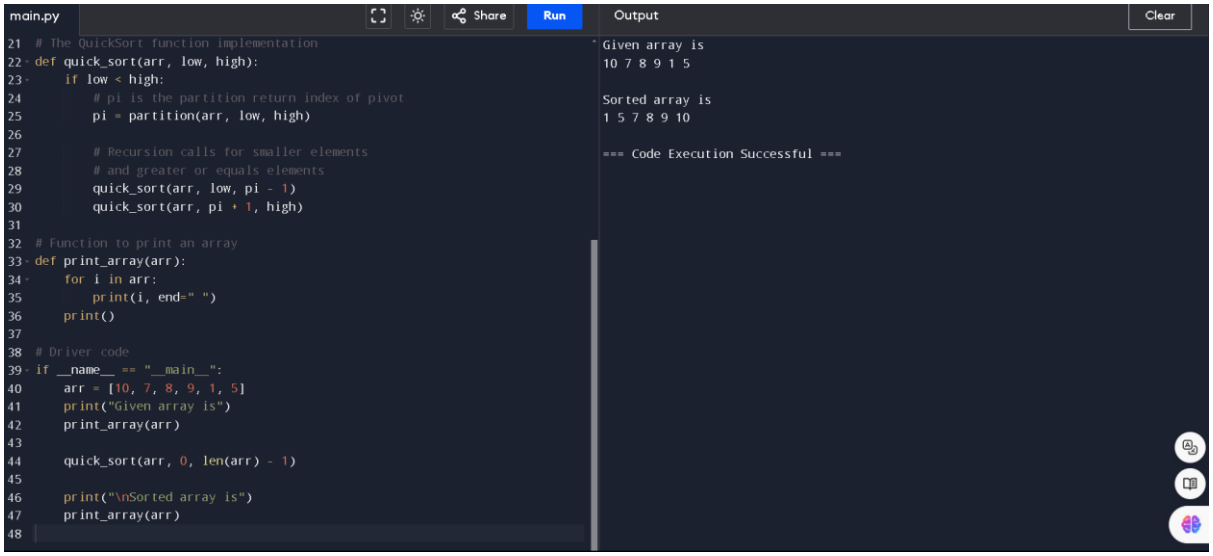
3. You are given an unsorted array 31,23,35,27,11,21,15,28. Write a program for Merge Sort and implement using any programming language of your choice. Test Cases : Input : N= 8, a[] = {31,23,35,27,11,21,15,28} Output : 11,15,21,23,27,28,31,35 Test Cases : Input : N= 10, a[] = {22,34,25,36,43,67, 52,13,65,17}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

4. Implement the Merge Sort algorithm in a programming language of your choice and test it on the array 12,4,78,23,45,67,89,1. Modify your implementation to count the number of comparisons made during the sorting process. Print this count along with the sorted array. 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,

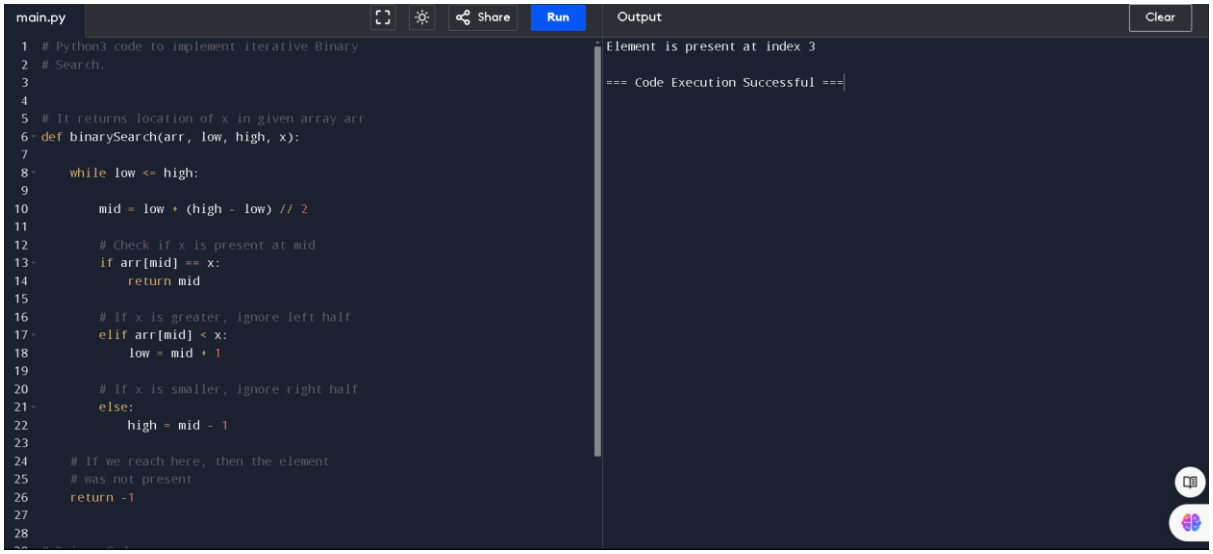


5. Given an unsorted array 10,16,8,12,15,6,3,9,5 Write a program to perform Quick Sort. Choose the first element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display the array after each recursive call until 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,

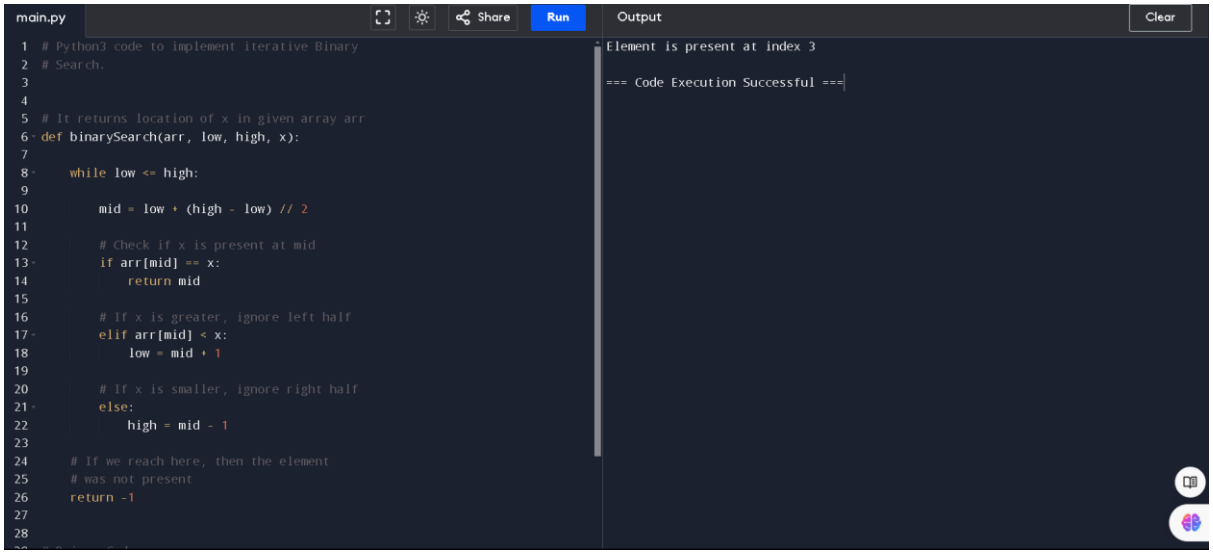
6. Implement the Quick Sort algorithm in a programming language of your choice and test it on the array 19,72,35,46,58,91,22,31. Choose the middle element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display 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



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



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 Output : 7 Test cases Input : N= 6, a[] = {20,40,60,80,100,120}, search key = 60 Output : 3



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]] 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

```python
# Python3 program for implementation of
# above approach

# Function to return required answer
def pClosest(points, K):

    points.sort(key = lambda K: K[0]**2 + K[1]**2)

    return points[:K]

# Driver program
points = [[3, 3], [5, -1], [-2, 4]]

K = 2

print(pClosest(points, K))
```

Output Clear

```
[[3, 3], [-2, 4]]

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