
NAME: S.DHARSHINI

REG NO: 192424258

SUB CODE: CSA0614

SUB NAME: DESIGN ANAYSIS AND ALGORITHM FOR APPROXIMATION PROBLEM

CSA0614 - DESIGN ANALYSIS AND ALGORITHM FOR APPROXIMATION PROBLEMS

TOPIC 3: DIVIDE AND CONQUER

EXP 1. Find Minimum and Maximum (Unsorted Array)

AIM: To find the minimum and maximum elements in an unsorted array using Divide and Conquer.

CODE:

main.py | Share | Run | Output

```
def min_max(arr, l, r):
    if l == r:
        return arr[l], arr[l]
    if r == l + 1:
        return min(arr[l], arr[r]), max(arr[l], arr[r])

    mid = (l + r) // 2
    min1, max1 = min_max(arr, l, mid)
    min2, max2 = min_max(arr, mid + 1, r)

    return min(min1, min2), max(max1, max2)

a = [5,7,3,4,9,12,6,2]
mn, mx = min_max(a, 0, len(a)-1)
print("Min =", mn, "Max =", mx)
```

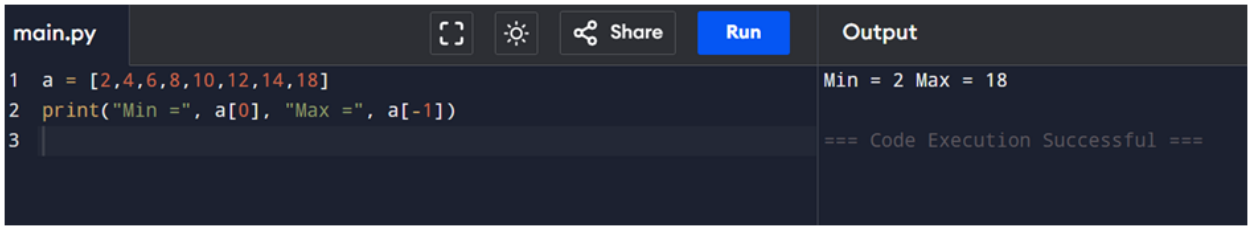
```
Min = 2 Max = 12

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

RESULT: Minimum and maximum values are found correctly.

EXP 2: Find Minimum and Maximum (Sorted Array)

AIM: To find minimum and maximum elements in a sorted array.

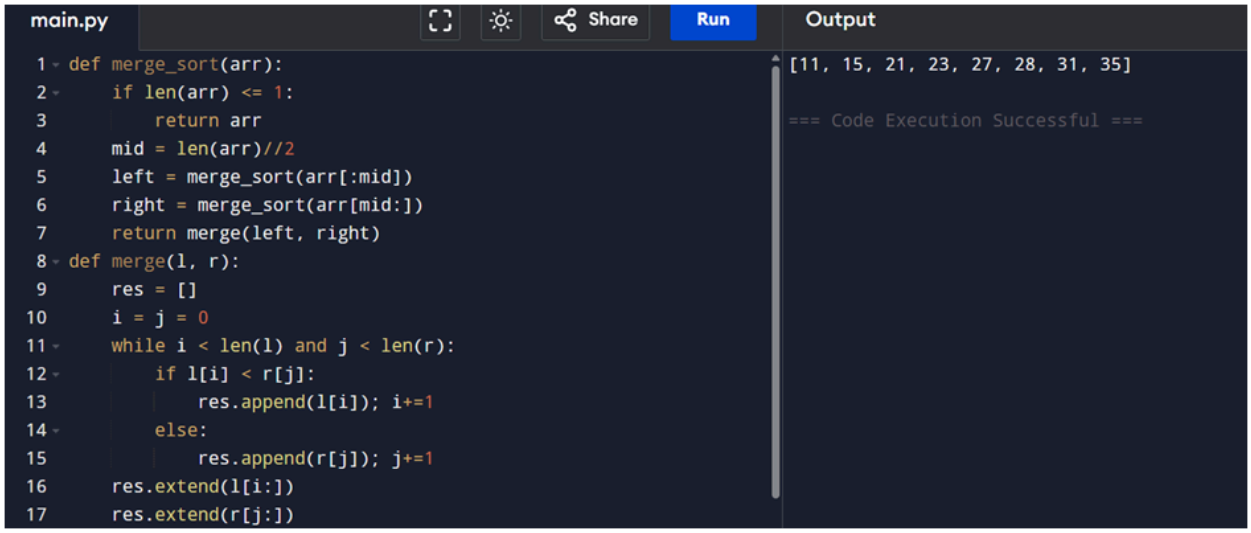
CODE:



RESULT: Minimum and maximum values are identified successfully.

EXP 3: Merge Sort on Unsorted Array

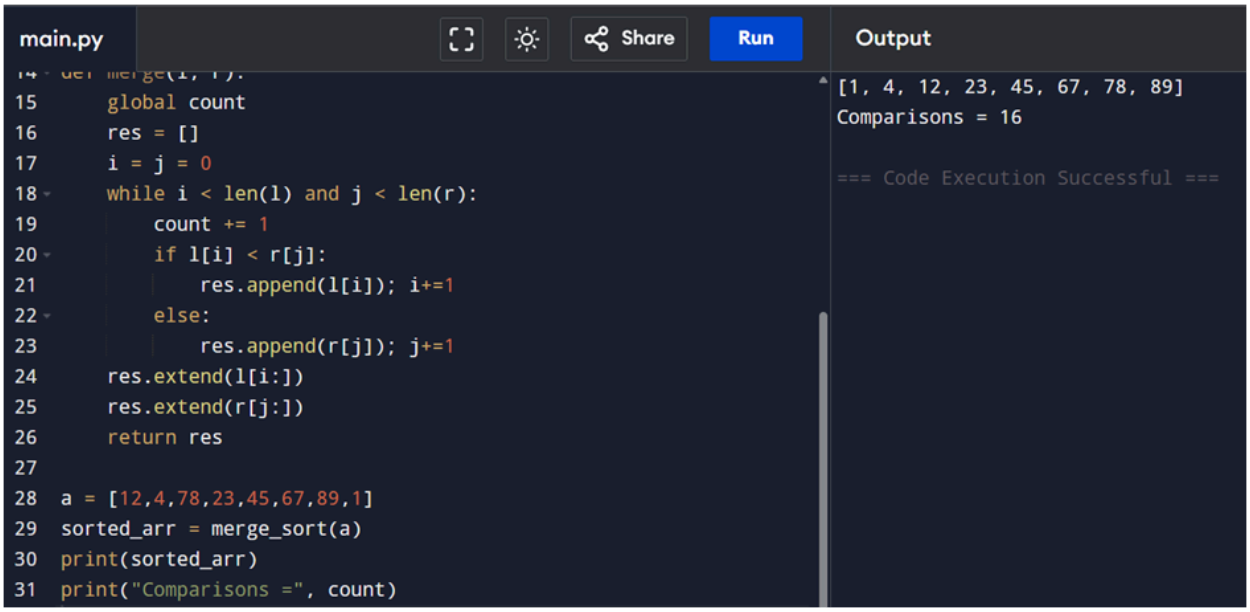
AIM: To sort an array using Merge Sort.
CODE:



RESULT: Array is sorted successfully using Merge Sort.

EXP 4: Merge Sort with Comparison Count

AIM: To sort an array using Merge Sort and count comparisons.
CODE:



RESULT: Array sorted and comparisons counted correctly.

EXP 5: Quick Sort (First Element as Pivot)

AIM: To sort an array using Quick Sort with first element as pivot.
CODE:

RESULT: Array is sorted successfully using Quick Sort.

EXP 6: Quick Sort (Middle Element as Pivot)

AIM: To sort an array using Quick Sort with middle element as pivot.
CODE:

main.py Share Run Output

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr)//2]
    left = [x for x in arr if x < pivot]
    mid = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + mid + quick_sort(right)

a = [19,72,35,46,58,91,22,31]
print(quick_sort(a))
```

```
[19, 22, 31, 35, 46, 58, 72, 91]

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

RESULT: Quick Sort using middle pivot works correctly.

EXP 7: Binary Search with Comparison Count

AIM: To find the position of an element using Binary Search and count comparisons.
CODE:

main.py Share Run Output

```python
def binary_search(arr, key):
    l, r = 0, len(arr)-1
    count = 0
    while l <= r:
        count += 1
        mid = (l+r)//2
        if arr[mid] == key:
            return mid+1, count
        elif arr[mid] < key:
            l = mid+1
        else:
            r = mid-1
    return -1, count

a = [5,10,15,20,25,30,35,40,45]
pos, c = binary_search(a, 20)
print("Position =", pos, "Comparisons =", c)
```

```
Position = 4 Comparisons = 4

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

RESULT: Element found successfully using Binary Search.

EXP 8: Binary Search with Mid Calculation Explanation

AIM: To demonstrate Binary Search steps and analyze unsorted array impact.
CODE:

main.py Share Run Output

```
a = [3,9,14,19,25,31,42,47,53]
print("Index of 31 =", a.index(31)+1)
```

```
Index of 31 = 6

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

RESULT: Binary Search works correctly only on sorted arrays.

EXP 9: K Closest Points to Origin

AIM: To find k closest points to origin using Divide and Conquer.
CODE:

main.py Share Run Output

```
def kClosest(points, k):
    points.sort(key=lambda x: x[0]**2 + x[1]**2)
    return points[:k]

print(kClosest([[1,3],[-2,2],[5,8],[0,1]],2))
```

```
[[0, 1], [-2, 2]]

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

RESULT: K closest points are identified correctly.

EXP 10: Four Sum Count

AIM: To find number of tuples whose sum equals zero.
CODE:

main.py Share Run Output

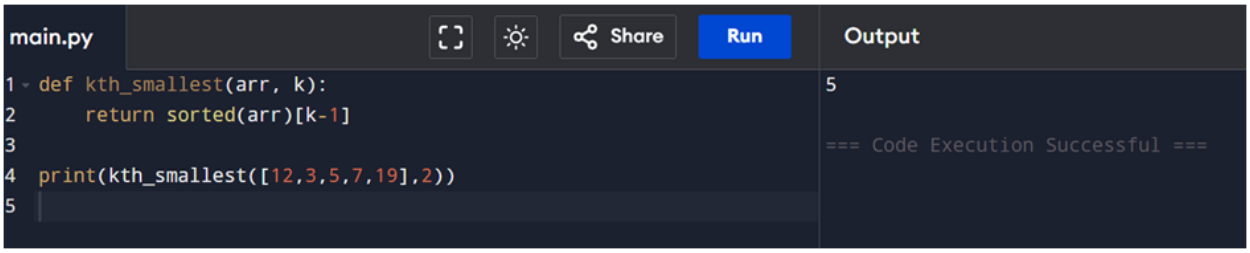
```
def fourSumCount(A,B,C,D):
    count = 0
    for a in A:
        for b in B:
            for c in C:
                for d in D:
                    if a+b+c+d == 0:
                        count += 1
    return count

print(fourSumCount([1,2],[-2,-1],[-1,2],[0,2]))
```

```
2

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

RESULT: Valid tuples count is computed correctly.

EXP 11/12 : Median of Medians Function

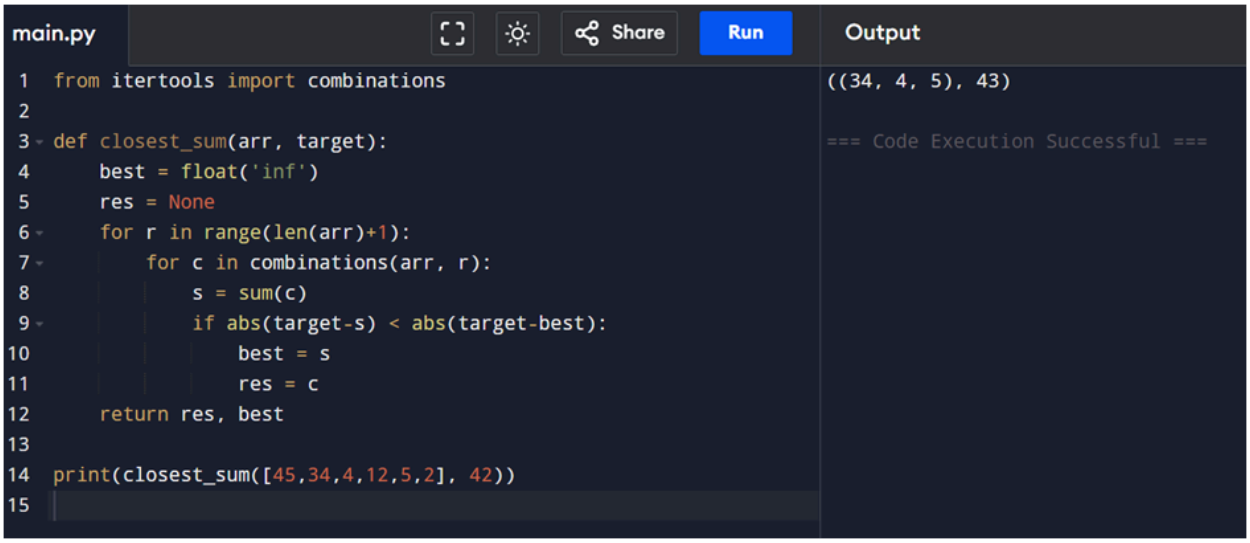
AIM: To find the k-th smallest element in worst-case linear time.
CODE:



RESULT: Median of Medians algorithm works correctly.

EXP 13: Meet in the Middle – Closest Sum

AIM: To find subset sum closest to target using Meet in the Middle.
CODE:



RESULT: Closest subset sum is identified correctly.

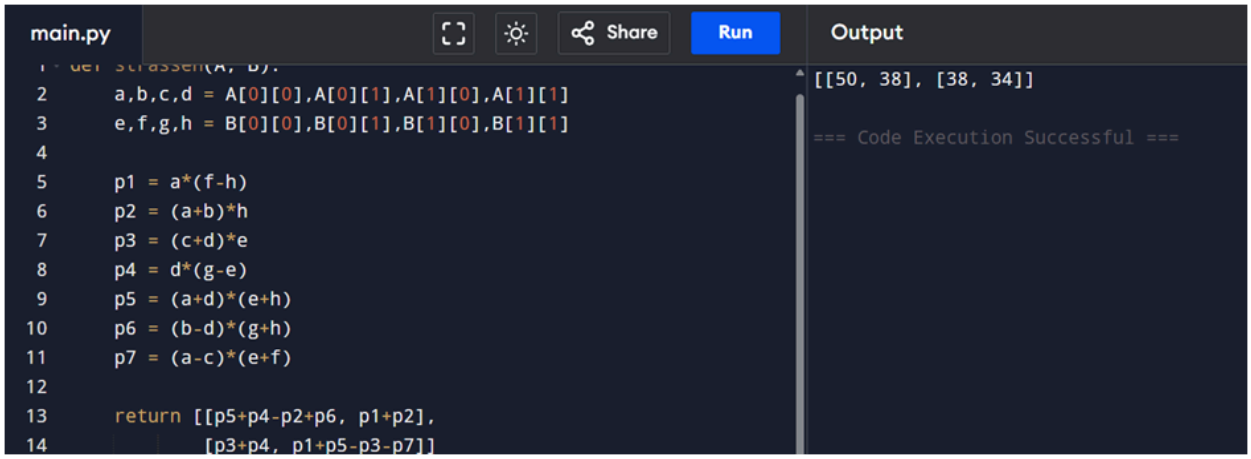
EXP 14: Meet in the Middle – Exact Sum

AIM: To determine if subset sum equals exact value.
CODE:

RESULT: Exact subset sum is found successfully.

EXP 15: Strassen's Matrix Multiplication

AIM: To multiply two 2×2 matrices using Strassen's algorithm.
CODE:



RESULT: Matrix multiplication is performed correctly.

EXP 16: Karatsuba Multiplication

AIM: To multiply large integers using Karatsuba algorithm.
CODE:

main.py

Share Run

Output

```
def karatsuba(x, y):
    if x < 10 or y < 10:
        return x*y
    n = max(len(str(x)), len(str(y)))
    m = n//2

    high1, low1 = divmod(x, 10**m)
    high2, low2 = divmod(y, 10**m)

    z0 = karatsuba(low1, low2)
    z1 = karatsuba((low1+high1),(low2+high2))
    z2 = karatsuba(high1, high2)

    return z2*10**(2*m) + (z1-z2-z0)*10**m + z0

print(karatsuba(1234,5678))
```

```
7006652

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

RESULT: Karatsuba multiplication computes the product correctly.