main.py

Share Run Output Clear

```python
def median_of_medians(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k-1]

    medians = [sorted(arr[i:i+5])[len(arr[i:i+5])//2] for i in range(0, len(arr), 5)]
    pivot = median_of_medians(medians, len(medians) // 2)

    low = [x for x in arr if x < pivot]
    high = [x for x in arr if x > pivot]
    k_index = len(low)

    if k_index == k - 1:
        return pivot
    elif k_index > k - 1:
        return median_of_medians(low, k)
    else:
        return median_of_medians(high, k - k_index - 1)

print(median_of_medians([12, 3, 5, 7, 19], 2))
print(median_of_medians([12, 3, 5, 7, 4, 19, 26], 3))
print(median_of_medians([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 6))
```

```
5
5
6

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

main.py

Share Run Output

```python
def median_of_medians(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k-1]
    medians = [sorted(arr[i:i+5])[len(arr[i:i+5])//2] for i in range(0, len(arr), 5)]
    pivot = median_of_medians(medians, len(medians)//2)
    low = [x for x in arr if x < pivot]
    high = [x for x in arr if x > pivot]
    if k <= len(low):
        return median_of_medians(low, k)
    elif k > len(arr) - len(high):
        return median_of_medians(high, k - (len(arr) - len(high)))
    return pivot

print(median_of_medians([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 6))
print(median_of_medians([23, 17, 31, 44, 55, 21, 20, 18, 19, 27], 5))
```

```
6
21

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

main.py

Share Run

```python
from itertools import chain, combinations

def subsets(arr):
    return chain(*map(lambda x: combinations(arr, x), range(0, len
        (arr)+1)))

def closest_sum(arr, target):
    n = len(arr)
    half = n // 2
    left_half = list(subsets(arr[:half]))
    right_half = list(subsets(arr[half:]))

    left_half_sums = [sum(subset) for subset in left_half]
    right_half_sums = [sum(subset) for subset in right_half]

    closest = float('inf')
    closest_subset = None

    for left_sum in left_half_sums:
        for right_sum in right_half_sums:
            current_sum = left_sum + right_sum
            if abs(target - current_sum) < abs(target - closest):
                closest = current_sum
                closest_subset = (left_sum, right_sum)

    return closest_subset
```

Output Clear

```
Closest subset for set1: (34, 7)
Closest subset for set2: (0, 10)

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

main.py

Share Run

```python
from itertools import combinations

def meet_in_the_middle(arr, target):
    n = len(arr)
    mid = n // 2

    left_half = arr[:mid]
    right_half = arr[mid:]

    left_sums = {sum(comb) for r in range(len(left_half) + 1) for
        comb in combinations(left_half, r)}
    right_sums = {sum(comb) for r in range(len(right_half) + 1) for
        comb in combinations(right_half, r)}

    for left_sum in left_sums:
        if (target - left_sum) in right_sums:
            return True
    return False

E1 = [1, 3, 9, 2, 7, 12]
exact_sum1 = 15
print(meet_in_the_middle(E1, exact_sum1))

E2 = [3, 34, 4, 12, 5, 2]
exact_sum2 = 15
print(meet_in_the_middle(E2, exact_sum2))
```

Output Clear

```
True
True

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

main.py

Share Run

```python
import numpy as np

def strassen(A, B):
    if len(A) == 1:
        return A * B

    mid = len(A) // 2

    A11 = A[:mid, :mid]
    A12 = A[:mid, mid:]
    A21 = A[mid:, :mid]
    A22 = A[mid:, mid:]

    B11 = B[:mid, :mid]
    B12 = B[:mid, mid:]
    B21 = B[mid:, :mid]
    B22 = B[mid:, mid:]

    M1 = strassen(A11 + A22, B11 + B22)
    M2 = strassen(A21 + A22, B11)
    M3 = strassen(A11, B12 - B22)
    M4 = strassen(A22, B21 - B11)
    M5 = strassen(A11 + A12, B22)
    M6 = strassen(A21 - A11, B11 + B12)
    M7 = strassen(A12 - A22, B21 + B22)

    C11 = M1 + M4 - M5 + M7
```

Output

```
[[34. 22.]
 [38. 34.]]

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

main.py

Share Run

```python
def karatsuba(x, y):
    if x < 10 or y < 10:
        return x * y

    n = max(len(str(x)), len(str(y)))
    half_n = n // 2

    a = x // 10**half_n
    b = x % 10**half_n
    c = y // 10**half_n
    d = y % 10**half_n

    ac = karatsuba(a, c)
    bd = karatsuba(b, d)
    abcd = karatsuba(a + b, c + d)

    return ac * 10**(2 * half_n) + (abcd - ac - bd) * 10**half_n + bd

x = 1234
y = 5678
z = karatsuba(x, y)
print(f"z={z}")
```

Output

```
z=7006652

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