main.py

Share Run

```python
def generate_subset_sums(arr):
    subset_sums = []
    n = len(arr)
    for i in range(1 << n):
        subset_sum = 0
        for j in range(n):
            if i & (1 << j):
                subset_sum += arr[j]
        subset_sums.append(subset_sum)
    return subset_sums
def meet_in_the_middle(arr, target):
    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]
    left_sums = generate_subset_sums(left)
    right_sums = generate_subset_sums(right)
    right_sums.sort()
    closest_sum = float('-inf')
    for sum_left in left_sums:
        for sum_right in right_sums:
```

Output

Clear

```
10

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

main.py

Share Run

Output Clear

```python
def partition(arr, pivot):
    left = [x for x in arr if x < pivot]
    right = [x for x in arr if x > pivot]
    return left, pivot, right
def median_of_medians(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k]
    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)
    left, pivot, right = partition(arr, pivot)
    if k < len(left):
        return median_of_medians(left, k)
    elif k == len(left):
        return pivot
    else:
        return median_of_medians(right, k - len(left) - 1)
arr1 = [23, 17, 31, 44, 55, 21, 20, 18, 19, 27]
print(median_of_medians(arr1, 5))
```

```
23

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

main.py

Share Run

```python
def strassen_2x2(A, B):
    a, b, c, d = A[0][0], A[0][1], A[1][0], A[1][1]
    e, f, g, h = B[0][0], B[0][1], B[1][0], B[1][1]
    P1 = a * (f - h)
    P2 = (a + b) * h
    P3 = (c + d) * e
    P4 = d * (g - e)
    P5 = (a + d) * (e + h)
    P6 = (b - d) * (g + h)
    P7 = (a - c) * (e + f)
    C11 = P5 + P4 - P2 + P6
    C12 = P1 + P2
    C21 = P3 + P4
    C22 = P1 + P5 - P3 - P7
    return [[C11, C12], [C21, C22]]
A1 = [[1, 7], [3, 5]]
B1 = [[6, 8], [4, 2]]
A2 = [[1, 7], [3, 5]]
B2 = [[1, 3], [7, 5]]
print("Test Case 1 Output:", strassen_2x2(A1, B1))
```

Output

Clear

```
Test Case 1 Output: [[34, 22], [38, 34]]
Test Case 2 Output: [[50, 38], [38, 34]]

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

main.py

Share Run

```python
def partition(arr, pivot):
    left = [x for x in arr if x < pivot]
    right = [x for x in arr if x > pivot]
    return left, pivot, right
def select(arr, k):
    if len(arr) <= 5:
        return sorted(arr)[k]
    medians = [sorted(arr[i:i + 5])[len(arr[i:i + 5]) // 2] for
        i in range(0, len(arr), 5)]
    pivot = select(medians, len(medians) // 2)
    left, pivot, right = partition(arr, pivot)
    if k < len(left):
        return select(left, k)
    elif k < len(left) + 1:
        return pivot
    else:
        return select(right, k - len(left) - 1)
arr1 = [12, 3, 5, 7, 19]
print(select(arr1, 2))

```

Output

Clear

```
7

=== 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)))
    m = n // 2
    a = x // 10**m
    b = x % 10**m
    c = y // 10**m
    d = y % 10**m
    P1 = karatsuba(a, c)
    P2 = karatsuba(b, d)
    P3 = karatsuba(a + b, c + d)
    return P1 * 10**(2 * m) + (P3 - P1 - P2) * 10**m + P2
x = 1234
y = 5678
print("Output:", karatsuba(x, y))
```

Output Clear

```
Output:7016652

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

main.py

Share Run

```
    return subset_sums
def meet_in_the_middle(arr, target):
    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]
    left_sums = generate_subset_sums(left)
    right_sums = generate_subset_sums(right)
    right_sums.sort()
    closest_sum = float('-inf')
    for sum_left in left_sums:
        for sum_right in right_sums:
            current_sum = sum_left + sum_right
            if abs(target - current_sum) < abs(target -
                closest_sum):
                closest_sum = current_sum
    return closest_sum
set1 = [1, 3, 2, 7, 4, 6]
target1 = 10
print(meet_in_the_middle(set1, target1))
```

Output

Clear

```
10

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

main.py

Share Run

```
        subset_sums.append(subset_sum)
    return subset_sums
def meet_in_the_middle(arr, target):
    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]
    left_sums = generate_subset_sums(left)
    right_sums = generate_subset_sums(right)
    right_sums_set = set(right_sums)
    for sum_left in left_sums:
        if (target - sum_left) in right_sums_set:
            return True
    return False
arr1 = [1, 3, 9, 2, 7, 12]
target1 = 15
arr2 = [3, 34, 4, 12, 5, 2]
target2 = 15
print(meet_in_the_middle(arr1, target1))
print(meet_in_the_middle(arr2, target2))
```

Output

Clear

```
True
True

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

main.py

Share

Run

Output

Clear

```python
def generate_subset_sums(arr):
    subset_sums = []
    n = len(arr)
    for i in range(1 << n):
        subset_sum = 0
        for j in range(n):
            if i & (1 << j):
                subset_sum += arr[j]
        subset_sums.append(subset_sum)
    return subset_sums
def meet_in_the_middle(arr, target):
    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]
    left_sums = generate_subset_sums(left)
    right_sums = generate_subset_sums(right)
    right_sums_set = set(right_sums)
    for sum_left in left_sums:
        if (target - sum_left) in right_sums_set:
            return True
```

```
True
True

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