3.14 SUBSET SUM USING MEET IN THE MIDDLE TECHNIQUE

Question:

Write a program to implement Meet in the Middle Technique. Given a large array of integers and an exact sum E, determine if there is any subset that sums exactly to E. Utilize the Meet in the Middle technique to handle the potentially large size of the array. Return true if there is a subset that sums exactly to E, otherwise return false.

AIM

To implement the Meet in the Middle algorithm to determine whether a subset of a large array of integers sums exactly to a given target value E.

ALGORITHM

1. Split the array into two halves: left and right.
2. Generate all possible subset sums for both halves.
3. Sort one half (say, right_sums) to enable binary search.
4. For each sum in left_sums, check if E - sum exists in right_sums.
5. If such a pair is found, return True; otherwise, return False.

PROGRAM

```python
def has_exact_subset_sum(arr, target):
    from itertools import combinations
    n = len(arr)
    left = arr[:n//2]
    right = arr[n//2:]

    def subset_sums(nums):
        return set(sum(comb) for r in range(len(nums)+1) for comb in combinations(nums, r))

    left_sums = subset_sums(left)
    right_sums = subset_sums(right)

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

def run_exact_sum():
    arr = list(map(int, input("Enter array: ").split()))
    target = int(input("Enter exact sum: "))
    print("Subset with exact sum exists:", has_exact_subset_sum(arr, target))

run_exact_sum()
```

Input:

[13,6,1,73,10,2,28] || 29

Output:

```
Enter array: 13 6 1 73 10 2 28
Enter exact sum: 29
Subset with exact sum exists: True
>>>
==========================================
Enter array: 13 6 1 73 10 2 28
Enter exact sum: 62
Subset with exact sum exists: False
>>>
```

RESULT:

Thus program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

· Time Complexity: $O(2^{\{n/2\}} \backslash log(2^{\{n/2\}})$

· Space Complexity: $O(2^{\{n/2\}})$