Minimum Subset Sum Diff

minimum subset sum differece explain every thing and give code in c++

Minimum Subset Sum Difference

Problem Explanation

Given a set of integers, partition the set into two subsets such that the absolute difference between the sums of the subsets is minimized.

The goal is to find the minimum difference between the two subset sums.

Key Concepts

1. **Total Sum of Array**: Let sumTotal be the sum of all elements in the array. If we split the array into two subsets $S1$ and $S2$, their sums are sum(S1) and sum(S2).
2. **Difference**: The difference between the two subsets is:

$$\text{diff} = |\text{sum(S1)} - \text{sum(S2)}|$$

Since:

$$\text{sum(S1)} + \text{sum(S2)} = \text{sumTotal}$$

We can rewrite the difference as:

$$\text{diff} = |\text{sumTotal} - 2 \times \text{sum(S1)}|$$

The problem then becomes finding a subset $S1$ such that sum(S1) is as close as possible to $\text{sumTotal}/2$.

Approach Using Dynamic Programming

We use a **Subset Sum Problem** approach to solve this:

1. **Subset Sum Table**: We create a boolean DP table $dp[n+1][\text{sumTotal}/2+1]$, where:
   - $dp[i][j]$ indicates whether it's possible to achieve a sum j using the first i elements of the array.
2. **Filling the DP Table**:
   - $dp[0][0] = \text{true}$ because a sum of 0 is always possible with no elements.

- $dp[i][j] = dp[i-1][j]$ (excluding the element) or $dp[i-1][j - arr[i-1]]$ (including the element).

3. **Finding the Closest Sum**: After filling the table, check the maximum sum $s1$ achievable from 0 to $\text{sumTotal}/2$.
4. **Calculate Minimum Difference**:

$$\text{minDiff} = \text{sumTotal} - 2 \times s1$$

C++ Code

cpp

```cpp
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;

int minSubsetSumDifference(vector<int>& arr) {
    int n = arr.size();
    int sumTotal = 0;
    for (int num : arr) sumTotal += num;

    // DP table: dp[i][j] tells if sum j can be achieved with first i elements
    vector<vector<bool>> dp(n + 1, vector<bool>(sumTotal / 2 + 1, false));

    // Initialize DP table
    for (int i = 0; i <= n; i++) dp[i][0] = true;

    // Fill DP table
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= sumTotal / 2; j++) {
            if (arr[i - 1] <= j) {
                dp[i][j] = dp[i - 1][j] || dp[i - 1][j - arr[i - 1]];
            } else {
                dp[i][j] = dp[i - 1][j];
            }
        }
    }

    // Find the closest sum to sumTotal / 2
    int s1 = 0;
    for (int j = sumTotal / 2; j >= 0; j--) {
        if (dp[n][j]) {
            s1 = j;
            break;
        }
    }

    int s2 = sumTotal - s1;
    return abs(s2 - s1);
}

int main() {
    vector<int> arr = {1, 6, 11, 5};
    cout << "Minimum Subset Sum Difference: " << minSubsetSumDifference(arr) << endl;
    return 0;
}
```

Explanation of the Code

1. **Input**: The array of integers.

2. **Calculate Total Sum**: Compute the total sum of the array.
3. **Dynamic Programming Table**: Construct a 2D DP table to find possible subset sums.
4. **Find Closest Sum** $s1$: Determine the maximum $s1$ possible, such that $s1 \leq \text{sumTotal}/2$.
5. **Compute Minimum Difference**: Calculate the absolute difference $|\text{sumTotal} - 2 \times s1|$.
6. **Output Result**: Print the minimum subset sum difference.

Complexity Analysis

- **Time Complexity**: $O(n \times \text{sumTotal}/2)$
- **Space Complexity**: $O(n \times \text{sumTotal}/2)$

This approach ensures an optimal solution to the problem.