1. Description:

This algorithm is used to find two subsets of a given array of integers and a total sum where the two subsets contain numbers that have an equal sum in the parent set.

Solution Method: Dynamic Programming Approach

**Example:**

Total Sum = 15

Array = {1,2,4,5,9}

We Initialize a 2D array with number of columns equal to the total sum provided and the number of rows equal to the length of array.

We Assume each cell as m[i][j].

i = row

j = column

if m[i][j] = 1 ; True

if m[i][j] = 0 ; False

Every cell shows that using the “Ni” numbers can we make the “SUM” for the given cell. So we start filling the cell using this technique. ( The Pseudo code is provided in the pseudo code section of this document ).

Step 1:

For 0th column.

We can make sum 0 with any given digit so 1 ( True ) for first column.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ni/SUM | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **1** | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4** | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5** | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **9** | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Step 2:

Similarly we fill the whole table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ni/SUM | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **1** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **4** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **9** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

2. Pseudo Code

Input array

Input sum

m[n][sum+1];

for(i=0; i<n; i++)

m[i][0] = 1;

for(int i=0;i<=sum;i++)

for(int j=0;j<=sum;j++)

m[i][j] = m[i-1][j] || m[i-1][j-a[j]];

return m[n-1][sum];

3. Time Complexity

Time complexity = O(arraysize\*totalSum)

4. Simulation Result

Note:

* Simulation Code and Output file for one RUN for each data size is saved in the Question’s Folder separately.
* Code Running Note: Kindly **Change File Path Of Output** before running the code.
* Code for Data Size = 10,000 Could not be compiled because Heap Size Exceeded on Java.

|  |  |
| --- | --- |
| **Data Size** | **Execution Time (ns)** |
| 10 | 946217 |
| 100 | 16881617 |
| 1000 | 618156404 |
| 10,000 | - |