Subset Sum Problem

Given a multiset of integers, is there a non-empty subset whose sum is zero? For example, given the set {-7, -3, -2, 9000, 5, 8}, the answer is *yes* because the subset {-3, -2, 5} sums to zero.

The subset sum problem is NP-complete, but there are several algorithms that can solve it reasonably fast in practice.

The most naïve algorithm would be to cycle through all subsets of *n* numbers and, for every one of them, check if the subset sums to the right number. The running time is of order $O(2^n * n)$, since there are 2^n subsets and, to check each subset, we need to sum at most *n* elements.

Horowitz and Sahni

Horowitz and Sahni published a faster exponential-time algorithm, which runs in time $O(2^{\frac{n}{2}} * \frac{n}{2})$, but requires much more space - $O(2^{\frac{n}{2}})$.

The algorithm splits arbitrarily the *n* elements into two sets. For each of these two sets, it stores a list of the sums of all possible subsets of its elements. Each of these two lists is then sorted. Given the two sorted lists, the algorithm can check if an element of the first array and an element of the second array sum up to *0*. Whenever the sum of the current element in the first array and the current element in the second array is more than *0*, the algorithm moves to the next element in the first array. If two elements that sum to *0* are found, it stops.

Pseudo-polynomial time dynamic programming solution

So we will create a 2D array of size (arr.size() + 1) * (target + 1) of type boolean. The state DP[i][j] will be true if there exists a subset of elements from A[0….i] with sum value = ‘j’**.**

This means that if current element has value greater than ‘current sum value’ we will copy the answer for previous cases. And if the current sum value is greater than the ‘ith’ element we will see if any of previous states have already experienced the sum=’j’ OR any previous states experienced a value ‘j – A[i]’ which will solve our purpose.

Time Complexity: O(sum*n), where sum is the ‘target sum’ and ‘n’ is the size of array.

Results

Run on (4 X 1800 MHz CPU s)