Find the smallest positive integer value that cannot be represented as sum of any subset of a given array

Given a sorted array (sorted in non-decreasing order) of positive numbers, find the smallest positive integer value that cannot be represented as sum of elements of any subset of given set.
Expected time complexity is O(n).

Examples:

```
Input:  arr[] = {1, 3, 6, 10, 11, 15};
Output: 2

Input:  arr[] = {1, 1, 1, 1};
Output: 5

Input:  arr[] = {1, 1, 3, 4};
Output: 10

Input:  arr[] = {1, 2, 5, 10, 20, 40};
Output: 4

Input:  arr[] = {1, 2, 3, 4, 5, 6};
Output: 22
```

We strongly recommend to minimize the browser and try this yourself first.

A **Simple Solution** is to start from value 1 and check all values one by one if they can sum to values in the given array. This solution is very inefficient as it reduces to subset sum problem which is a well known NP Complete Problem.

We can solve this problem **in O(n) time** using a simple loop. Let the input array be arr[0..n-1]. We initialize the result as 1 (smallest possible outcome) and traverse the given array. Let the smallest element that cannot be represented by elements at indexes from 0 to (i-1) be ‘res’, there are following two possibilities when we consider element at index i:

1) We decide that 'res' is the final result: If arr[i] is greater than 'res', then we found the gap which is 'res' because the elements after arr[i] are also going to be greater than 'res'.

2) The value of 'res' is incremented after considering arr[i]: The value of 'res' is incremented by arr[i] (why? If elements from 0 to (i-1) can represent 1 to 'res-1', then elements from 0 to i can represent from 1 to 'res + arr[i] – 1' be adding 'arr[i]' to all subsets that represent 1 to 'res')

Following is C++ implementation of above idea.

```
// C++ program to find the smallest positive value that cannot be
// represented as sum of subsets of a given sorted array
#include <iostream>
using namespace std;

// Returns the smallest number that cannot be represented as sum
// of subset of elements from set represented by sorted array arr[0..n-1]
int findSmallest(int arr[], int n)
{
   int res = 1; // Initialize result

   // Traverse the array and increment 'res' if arr[i] is
   // smaller than or equal to 'res'.
   for (int i = 0; i < n && arr[i] <= res; i++)
       res = res + arr[i];

   return res;
}

// Driver program to test above function
int main()
{
   int arr1[] = {1, 3, 4, 5};
   int n1 = sizeof(arr1)/sizeof(arr1[0]);
   cout << findSmallest(arr1, n1) << endl;

   int arr2[] = {1, 2, 6, 10, 11, 15};
   int n2 = sizeof(arr2)/sizeof(arr2[0]);
   cout << findSmallest(arr2, n2) << endl;

   int arr3[] = {1, 1, 1, 1};
   int n3 = sizeof(arr3)/sizeof(arr3[0]);
   cout << findSmallest(arr3, n3) << endl;

   int arr4[] = {1, 1, 3, 4};
   int n4 = sizeof(arr4)/sizeof(arr4[0]);
   cout << findSmallest(arr4, n4) << endl;

   return 0;
}
```

Run on IDE

Output:

```
2
4
5
10
```

Time Complexity of above program is O(n).

This article is contributed by **Rahul Gupta**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

64 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

3.6 Average Difficulty : **3.6/5.0**
Based on **9** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.