Tug of War - GeeksforGeeks

Given a set of n integers, divide the set in two subsets of n/2 sizes each such that the difference of the sum of two subsets is as minimum as possible. If n is even, then sizes of two subsets must be strictly n/2 and if n is odd, then size of one subset must be (n-1)/2 and size of other subset must be (n+1)/2.

For example, let given set be {3, 4, 5, -3, 100, 1, 89, 54, 23, 20}, the size of set is 10. Output for this set should be {4, 100, 1, 23, 20} and {3, 5, -3, 89, 54}. Both output subsets are of size 5 and sum of elements in both subsets is same (148 and 148).
Let us consider another example where n is odd. Let given set be {23, 45, -34, 12, 0, 98, -99, 4, 189, -1, 4}. The output subsets should be {45, -34, 12, 98, -1} and {23, 0, -99, 4, 189, 4}. The sums of elements in two subsets are 120 and 121 respectively.

The following solution tries every possible subset of half size. If one subset of half size is formed, the remaining elements form the other subset. We initialize current set as empty and one by one build it. There are two possibilities for every element, either it is part of current set, or it is part of the remaining elements (other subset). We consider both possibilities for every element. When the size of current set becomes n/2, we check whether this solutions is better than the best solution available so far. If it is, then we update the best solution.

Following is C++ implementation for Tug of War problem. It prints the required arrays.

```
#include <iostream>#include <stdlib.h>#include <limits.h>
using namespace std;
// function that tries every possible solution by calling itself
recursively
void TOWUtil(int* arr, int n, bool* curr_elements, int
no_of_selected_elements,
             bool* soln, int* min_diff, int sum, int curr_sum, int
curr_position)
{
    // checks whether the it is going out of bound
    if (curr_position == n)
        return;
    // checks that the numbers of elements left are not less than the
    // number of elements required to form the solution
    if ((n/2 - no_of_selected_elements) > (n - curr_position))
        return;
    // consider the cases when current element is not included in the
solution
    TOWUtil(arr, n, curr_elements, no_of_selected_elements,
              soln, min_diff, sum, curr_sum, curr_position+1);
    // add the current element to the solution
    no_of_selected_elements++;
    curr_sum = curr_sum + arr[curr_position];
    curr_elements[curr_position] = true;
    // checks if a solution is formed
    if (no_of_selected_elements == n/2)
    {
        // checks if the solution formed is better than the best solution
so far
        if (abs(sum/2 - curr_sum) < *min_diff)
        {
            *min_diff = abs(sum/2 - curr_sum);
            for (int i = 0; i<n; i++)
                soln[i] = curr_elements[i];
        }
    }
    else
    {
        // consider the cases where current element is included in the
solution
        TOWUtil(arr, n, curr_elements, no_of_selected_elements, soln,
                  min_diff, sum, curr_sum, curr_position+1);
    }
    // removes current element before returning to the caller of this
function
    curr_elements[curr_position] = false;
}// main function that generate an arr
void tugOfWar(int *arr, int n)
{
    // the boolen array that contains the inclusion and exclusion of an
element
    // in current set. The number excluded automatically form the other set
    bool* curr_elements = new bool[n];
    // The inclusion/exclusion array for final solution
    bool* soln = new bool[n];
    int min_diff = INT_MAX;
    int sum = 0;
    for (int i=0; i<n; i++)
    {
        sum += arr[i];
        curr_elements[i] =  soln[i] = false;
    }
    // Find the solution using recursive function TOWUtil()
    TOWUtil(arr, n, curr_elements, 0, soln, &min_diff, sum, 0, 0);
    // Print the solution
    cout << "The first subset is: ";
    for (int i=0; i<n; i++)
    {
        if (soln[i] == true)
            cout << arr[i] << " ";
    }
    cout << "\nThe second subset is: ";
    for (int i=0; i<n; i++)
    {
        if (soln[i] == false)
            cout << arr[i] << " ";
    }
}// Driver program to test above functions
int main()
{
    int arr[] = {23, 45, -34, 12, 0, 98, -99, 4, 189, -1, 4};
    int n = sizeof(arr)/sizeof(arr[0]);
    tugOfWar(arr, n);
    return 0;
}
```

Output:

```
The first subset is: 45 -34 12 98 -1
The second subset is: 23 0 -99 4 189 4
```

This article is compiled by Ashish Anand and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.