Laboratory work #1

Student: CAO Xinyang  
Student ID: 20321308  
Timus Name: hduads2022\_20321308

Mail: c.x\_yang@foxmail.com

Problem #1005

Screenshot from Timus:



Explanation of algorithm:

We can go through all the possible scenarios using a traversal method. When we have n numbers, we know that there are 2^n (1<<n) possibilities. So in order to avoid omission and duplication, we use binary encoding. Write all cases as binary numbers from 0 to (2^n-1). Each bit of binary represents its corresponding bit in the array. If the bit is 1, the weight represented by this bit is added to SUM1. If it is 0, it is not added. For example, when n=5, 10011 represents SUM1 plus the weight of weight[0], weight[1], weight[2]. Then we can get ((1 & (i >> j)) != 0) to extract the j th bit of i and determine whether it is 0. Add them up to calculate the difference and compare it with Min. After all the comparisons, we store the final result in Min.

Computational complexity of algorithm:

The algorithm iterates over all possible (2 ^ n) cases.

T(N) = O(2 ^ N)

Source code:

import java.util.Scanner;

import java.util.stream.Stream;

public class App {

public static void main(String[] args) throws Exception {

Scanner scan = new Scanner(System.in);

int n = Integer.valueOf(scan.nextLine());

String input = scan.nextLine();

String[] arr = input.split(" ");

Integer[] weights = Stream.of(arr).mapToInt(Integer::parseInt).boxed().toArray(Integer[]::new);

int total\_Sum = 0;

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

total\_Sum += weights[i] ;

int sum1;

int Min = total\_Sum;

for(int i = 0; i < (1 << n); i++){

sum1 = 0;

for(int j = 0;j < n;j++){

if((1 & (i >> j)) != 0){

sum1 += weights[j];

}

}

if(Math.abs(2 \* sum1 - total\_Sum) < Min){

Min = Math.abs(2 \* sum1 - total\_Sum);

}

}

System.out.println(Min);

}

}