2. Carrying the grocery bags

You want to minimize the number of trip needed to carry your grocery bags. After buying your groceries, all of the bags weights between 1.01 kg and 3.00 kg. All bags must be then taken to to your house. One trip is described as selecting a number of bags which together do not weigh more than 3.00 kg, to store them inside your house and then return to your car. Given the number of bags n, and the weights of each bag, determine the minimum number of trips you have to make.

Example

n = 5

weight = [1.01, 1.99, 2.5, 1.5, 1.01]

You can carry all bags out in *3* trips: [1.01 + 1.99 , 2.5, 1.5 + 1.01].

Function Description

Complete the function *minimumTravel* in the editor below.

minimumTravel has the following parameter(s):

float weight[n]: weights of the bags

Returns

int: the minimum number of trips required

Constraints

- $1 \leq n \leq 1000$
- $1.01 \leq weight[i] \leq 3.0$

Hint

What is the maximum number of bags that can be carried at any point?

It's probably wise to match the heaviest bags with the lightest.

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the number of elements in *weight*.

Each line *i* of the *n* subsequent lines (where $0 \leq i < n$) contains a floating point number that describes *weight[i]*.

describes *weight[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
5       →   weight[] size n = 5
1.01    →   weight = [1.01, 1.01, 1.01,
1.4, 2.4]
1.01
1.01
1.4
2.4
```

Sample Output

```
3
```

Explanation

You can carry all bags out in *3* trips:

The first *2* bags together, the 3^{rd} and 4^{th} together and the last one alone

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN       Function
-----       --------
4       →   weight[] size n = 4
1.01    →   weight = [1.01. 1.991, 1.32,
1.4]
1.991
1.32
1.4
```

Sample Output

```
3
```

Explanation

You can carry all bags out in *3* trips:
The 1^{st} and 2^{nd} bags separately and the 3^{rd} and 4^{th} together