

children & rides

In an amusement park, there are rides being constructed for N children. Each ride has a maximum capacity limit of B for the total weight of the children on board.

You are given an array of weights of children, where $A[i]$ is the weight of the i -th child, and an infinite number of rides. Each ride can carry **at most two** children at the same time, provided the sum of their weights is at most the ride's capacity limit.

Find the minimum number of rides to accommodate all the children.

$$[3, 2, 2, 0, 1], \quad B = 4$$

$$\begin{array}{r} 2 \ 2 \\ \hline 4 \text{ kg} \end{array}$$

$$[3, 2, 2, 0, 1]$$



$$[0, 1, 2, 2, 3]$$

↑ - - - ↑

$$(0, 3) \rightarrow 1$$

$$(1, 2) \rightarrow 2$$

$$(2) \rightarrow \underline{3}$$

$$[0, 1, 2, 2, 3] \quad B = 4$$

$$[1, 1, 2, 2, 3, 4] \quad B = 4$$

$$n = \underline{10^5}$$

```
count = 0;
```

```
Arrays.sort(arr);
```

```
l = 0;
```

```
r = arr.length - 1;
```

```
while (l < r) {
```

```
    sum = arr[l] + arr[r];
```

```
    if (sum <= B) {
```

```
        count++;
```

```
        l++;
```

```
        r--;
```

```
    }
```

```
    else {
```

```
        count++;
```

```
        r--;
```

```
    }
```

```
}
```

```
if (l == r) {
```

```
    count++;
```

```
}
```

Max Product of 3 Elements

Given an array of integers **A** of size **N**, return the maximum product of any three numbers from the array.

-1, 2, 1, 5 } best

the

3 elements

-ve the

2 neg

1+ve

-ve

-5 -4 -3 -2 -1

-6

first two & last one

-20

arrays, best (ans) :

return max (arr[0]*arr[1]*arr[n-1],

arr[n-1]*arr[n-2]*arr[n-3])

Search for a Range.

Given a **sorted array** of integers A (0-indexed) of size N, find the **left most** and the **right most** index of a given integer B in the array A.

Return an array of size 2, such that

First element = Left most index of B in A

Second element = Right most index of B in A.

If B is not found in A, return [-1, -1].

[5, 7, 7, 8, 8, 10] B = 8

Incorrect.

int sum = 10⁹

10⁸

10⁹

overflowed.

```
Java 8 (Array Support)
29
30
31 public boolean check(int[] arr, int total, int lenOfArray){
32     long sum = 0L;
33     int lengthOfWindow = 0;
34
35     for(int i=0; i<lenOfArray; i++){
36         sum += arr[i];
37     }
38     if((int)sum>total) return false;
39
40     for(int j=lenOfArray, x=0; j<arr.length; j++,x++){
41         sum += arr[j] - arr[x];
42         if((int)sum>total) return false;
43     }
44
45     /*for(int i=0; i<arr.length; i++){
```

correct

```
Java 8 (Array Support)
29
30
31 public boolean check(int[] arr, int total, int lenOfArray){
32     long sum = 0L;
33     int lengthOfWindow = 0;
34
35     for(int i=0; i<lenOfArray; i++){
36         sum += arr[i];
37     }
38     if(sum>(long)total) return false;
39
40     for(int j=lenOfArray, x=0; j<arr.length; j++,x++){
41         sum += arr[j] - arr[x];
42         if(sum>(long)total) return false;
43     }
44
45     /*for(int i=0; i<arr.length; i++){
```

Java 8 (Array Support)

```
32     int sum = 0;
33     int lengthOfWindow = 0;
34     int j=0;
35     /* ...
36     */
37     for(int i=0; i<arr.length; i++){
38         sum += arr[i];
39         ++lengthOfWindow;
40
41         if(i==arr.length-1 && (lengthOfWindow<=lenOfArray && sum>total)) return false;
42
43         if(lengthOfWindow==lenOfArray){
44             if(sum>total) return false;
45
46             lengthOfWindow = 0;
47             sum = 0;
48             i=j;
49         }
50     }
51     return true;
52 }
```

weird Test case.