Find Polygon with the Largest Perimeter (LEETCODE-2971)

GitHub: **github.com/BCAPATHSHALA**
Date: 15-02-2024

Given polygon

- At least 3 sides $\Rightarrow (k >= 3)$
- longest side (a_k) < sum of other sides

$\hookrightarrow a_k < a_1 + a_2 + \cdots + a_{k-1}$

This condition is Applied for checking is polygon or Not

where $a_1 <= a_2 <= a_3 <= a_4 \cdots a_{k-1} <= a_k$

$\hookrightarrow$ Given Array ko SORT KARNA PADEGA To find the Longest side

Ex

3
2
2
polygon

sides

$a_1 = 2$
$a_2 = 2$
$a_3 = 3$

Perimeter
$= a_1 + a_2 + a_3$
$= 2 + 2 + 3$
$= 7$

```
Input: nums = [1,12,1,2,5,50,3]
Output: 12
```

Explanation

a_1	a_2	a_3	a_4	a_5	a_6	a_7
1	1	2	3	5	12	50
0	1	2	3	4	5	6

↖ largest side $= a_K$

Check polygon OR not

↳ $a_K < a_1 + a_2 + a_3 + \dots + a_{K-1}$

Case 1 $a_7 = 50$ sum $= 24$ $\Rightarrow$ $50 < 24$ ✗

Not polygon

Case 2 $a_6 = 12$ sum $= 12$ $\Rightarrow$ $12 < 12$ ✗

Not polygon

Case 3 $a_5 = 5$ sum $= 7$ $\Rightarrow$ $5 < 7$ ✓

This is polygon

↳ perimeter $= 5 + 7$
$= 12$

Case 4 $a_4 = 3$ sum $= 4$ $\Rightarrow$ $3 < 4$ ✓

↳ perimeter $= 3 + 4$
$= 7$

Case 5 $a_3 = 2$ sum $= 2$ $\Rightarrow$ $2 < 2$ ✗

Case 6 a_2 STOP $\boxed{K >= 3}$

$2 >= 3$ ✗

output

return $\max(7, 12)$
$= 12$

Approach Backward

STEP1 SORT Array

STEP2 Get sum of All elements

STEP3 Traverse the array backwards

STEP4 check (sum - largest side) > largest side

STEP5 This condition is met

↳ return perimeter

STEP6 if No suitable perimeter found

↳ return -1

SORT

Nums	1	1	2	3	5	12	50
	0	1	2	3	4	5	6 ← i

i=6

sum = 74

largest side = 50

sum = 74 - 50
= 24
24 > 50 ✗

i=5

sum = 24

largest side = 12

sum = 24 - 12
= 12
12 > 12 ✗

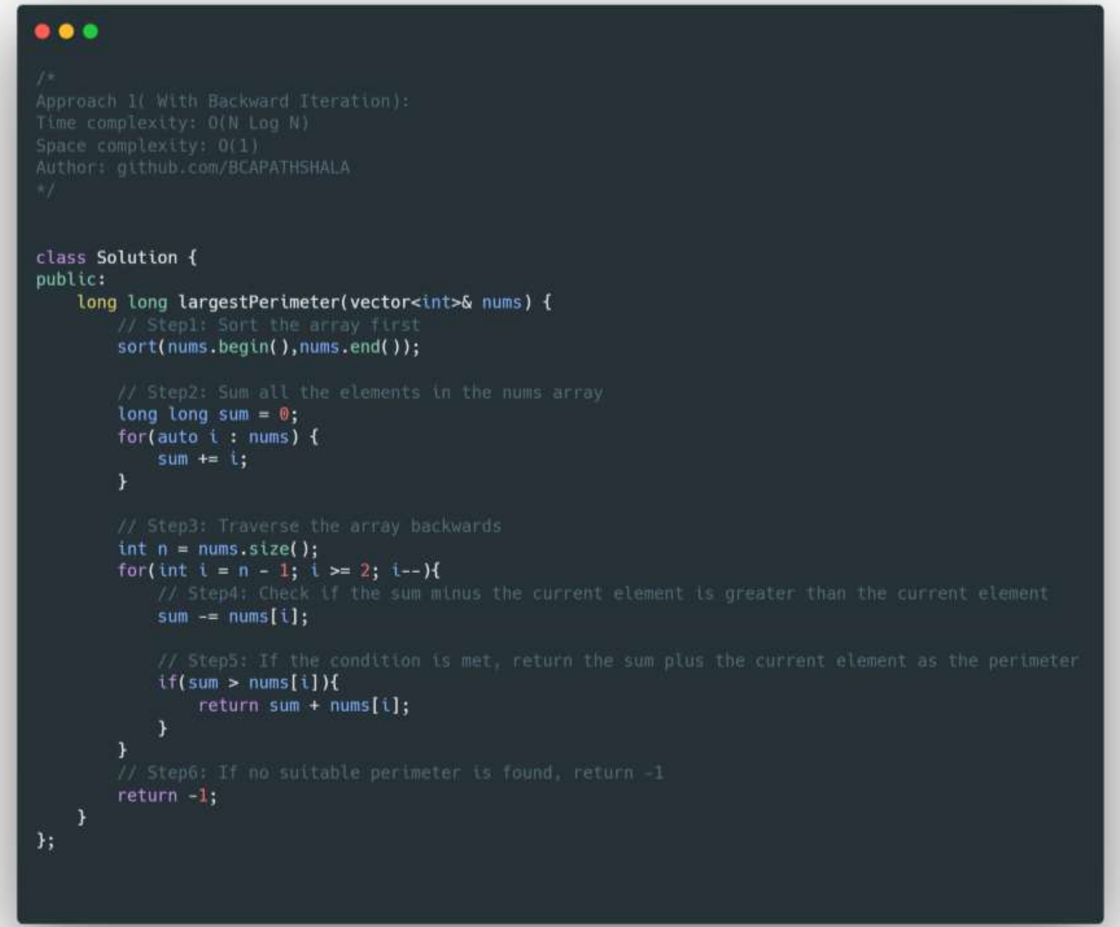
i=4

sum = 12

largest side = 5

sum = 12 - 5
= 7
7 > 5 ✓

↳ return (sum + largest side)
= 7 + 5
= $\boxed{12}$



```cpp
/*
Approach 1( With Backward Iteration):
Time complexity: O(N Log N)
Space complexity: O(1)
Author: github.com/BCAPATHSHALA
*/


class Solution {
public:
    long long largestPerimeter(vector<int>& nums) {
        // Step1: Sort the array first
        sort(nums.begin(),nums.end());

        // Step2: Sum all the elements in the nums array
        long long sum = 0;
        for(auto i : nums) {
            sum += i;
        }

        // Step3: Traverse the array backwards
        int n = nums.size();
        for(int i = n - 1; i >= 2; i--){
            // Step4: Check if the sum minus the current element is greater than the current element
            sum -= nums[i];

            // Step5: If the condition is met, return the sum plus the current element as the perimeter
            if(sum > nums[i]){
                return sum + nums[i];
            }
        }
        // Step6: If no suitable perimeter is found, return -1
        return -1;
    }
};
```

T.C. $\Rightarrow$ $O(\log N)$ for sorting

T.C. $\Rightarrow$ $O(N)$ for loop

$$\boxed{\text{overall T.C.} \Rightarrow O(N\log N)}$$

S.C. $\Rightarrow$ $O(1)$

where N is length of input Array.