HW 08: EKO SPOJ

Problem Statement:
*LAKSHAY BHAIYA needs to chop down M **metres of wood**.*
It is an easy job for him since he has a nifty new wood-cutting machine that can take down forests like wildfire.
However, LAKSHAY BHAIYA is only allowed to cut a single row of trees.

LAKSHAY BHAIYA's machine works as follows:
He sets a height parameter H (in meters), and the machine raises a giant sawblade to that height
and cuts off all tree parts higher than H (of course, trees not higher than H meters remain intact).
He then takes the parts that were cut off.

***He is ecologically minded**, so he doesn't want to cut off more wood than necessary.*
That's why he wants to set his sawblade as high as possible.
*Help LAKSHAYA find the '**maximum integer height of the sawblade**' that still allows him to cut off '**at least M metres of wood**'.*

Example 01:
Input:
4 7
20 15 10 17

Output:
15

📌***Observation:***
Number of trees/Array's size => N = 4
At least M metres of wood => M = 7
Trees Array => trees [20, 15, 10, 17]

1st tree *ki height = trees[0] = 20 meters*
2nd tree *ki height = trees[1] = 15 meters*
3rd tree *ki height = trees[2] = 10 meters*
4rt tree *ki height = trees[3] = 17 meters*

🖊️***PROBLEM KYA HAI:*** *We have to find* ***maximum integer height of sawblade*** *that still allows to cut off* ***at least M meters of wood.***

hame sawblade ki height ko aisa rakhna hai ki trees ko iss tarike se kate jo required height of wood se kam to bilkul na ho or usse bahut jyada bhi na ho.
To meaning yeh niklta hai ki hame sawblade ki ***highest maximum height*** *me se* ***(Possible)least maximum height*** *btani hai.*

📌***agar saw blade ki height = 0 meter hai to*** *=> ek bhi tree jungle nhi rahega to hum kah skte hai ki yeh* ***least minimum height*** *hai*
(wood height = 20+15+10+17 = 62 meters)

📌***agar sow blade ki height = 20 meter hai to*** *=> all trees will save tu hum kah skte hai ki yeh* ***highest maximum height*** *hai*
(wood height = 0 meters)

Required (sow blade height) *answer always less than or equal to 20 meters in this case Janha par M meters of wood ya usse thodi jyada M meters of wood milegi. DRY RUN KARNE SE OR BHI SAMJH AYEGA ABHI...*

OPTIMAL APPROACH: ***Define search space and predicate function***
Step 01: *Find maximum height tree to create search space's end point (highest maximum height of sawblade)*
Step 02: *Now, Applying Binary Search on search space BinarySearch()*
Step 03: *create predicate function isPossibbleSol()*

Time Complexity: *O(N*log(end)), Here N is size of array trees and end is the maximum heighted tree in array*
Space Complexity: *O(1), no extra space used*

Resource: https://www.spoj.com/problems/EKO/

Example:01

Example 01:
Input:
4 7
20 15 10 17

sawblade ki possible hight → Output:
15

OBSERVATION

Number of trees = N = 4

At Least Required wood = M = 7 meters

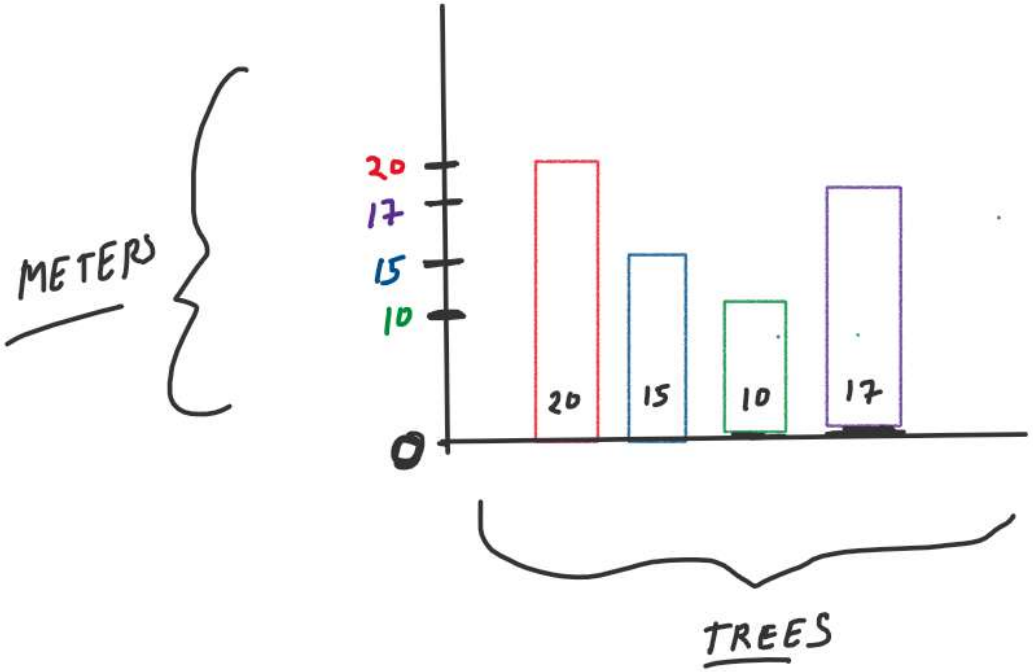
Trees = [20 , 15 , 10 , 17]

1st Tree Height 20

2nd Tree Height 15

3Rd Tree Height 10

4rt Tree Height 17

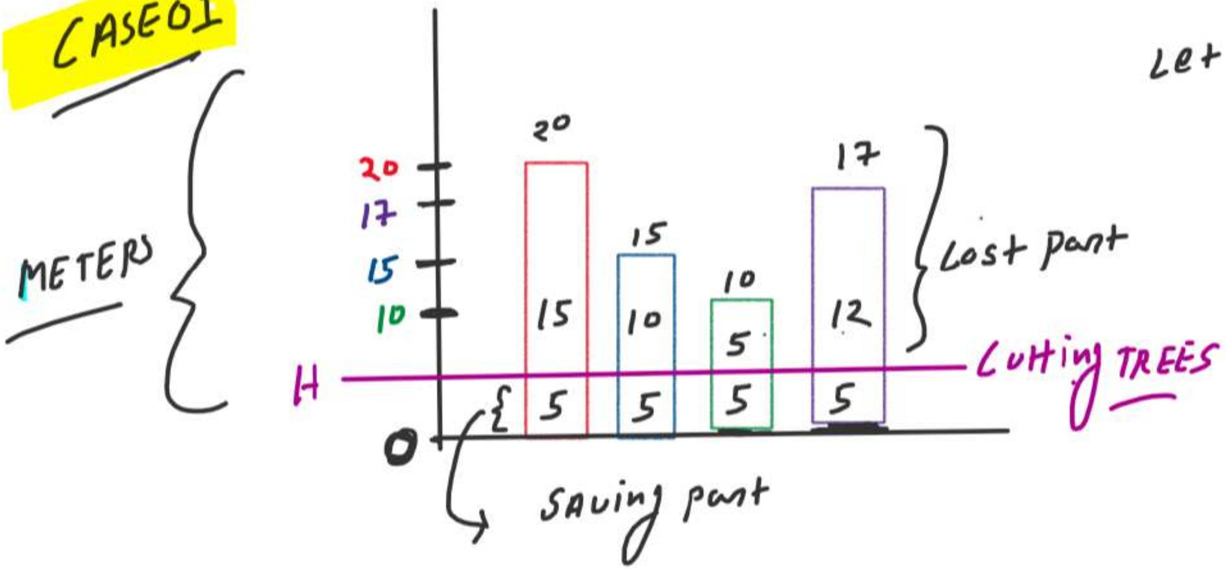


Note

{Total wood = when sawblade hight H}

EXPLANATION

CASE 01



possible sawblade height

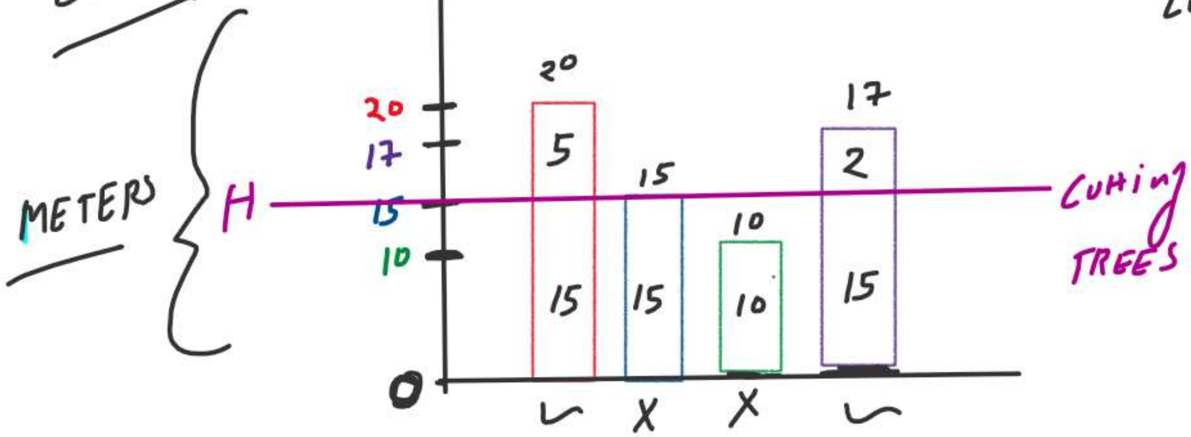
Let ⇒ H = 5 meters

$$\text{Total wood} = \overset{T_1}{(20-5)} + \overset{T_2}{(15-5)} + \overset{T_2}{(10-5)} + \overset{T_3}{(17-5)}$$

$$= 15 + 10 + 5 + 12$$

$$= 42 \text{ meters wood}$$

CASE 02



possible ANS

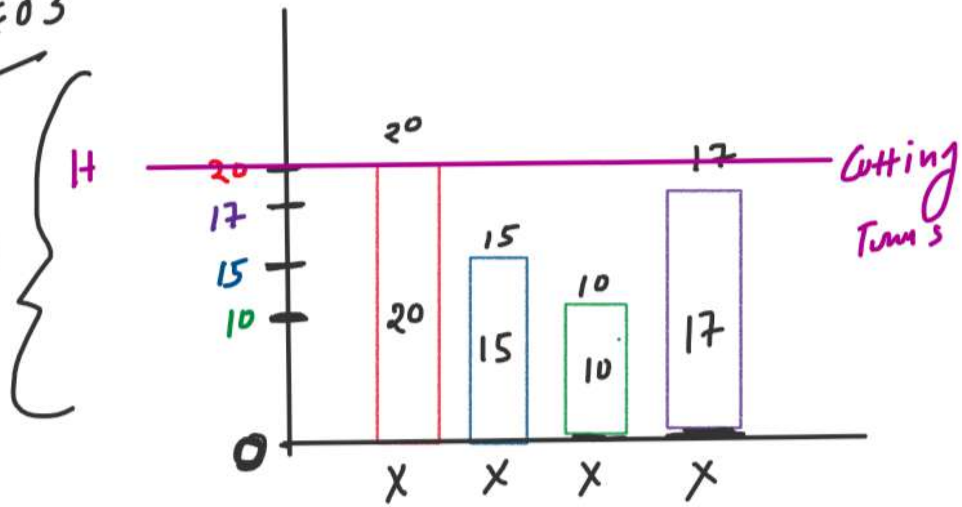
Let ⇒ $H = 15$ meters

$$\text{Total wood} = \overset{T_1}{(20-15)} + \overset{T_2}{(15-15)} + \overset{T_2}{(10-15)} + \overset{T_3}{(17-15)}$$

$$= 5 + 0 + 0 + 2$$

$$= \boxed{7} \text{ meters wood}$$

CASE 03



All Trees will saw in jungle

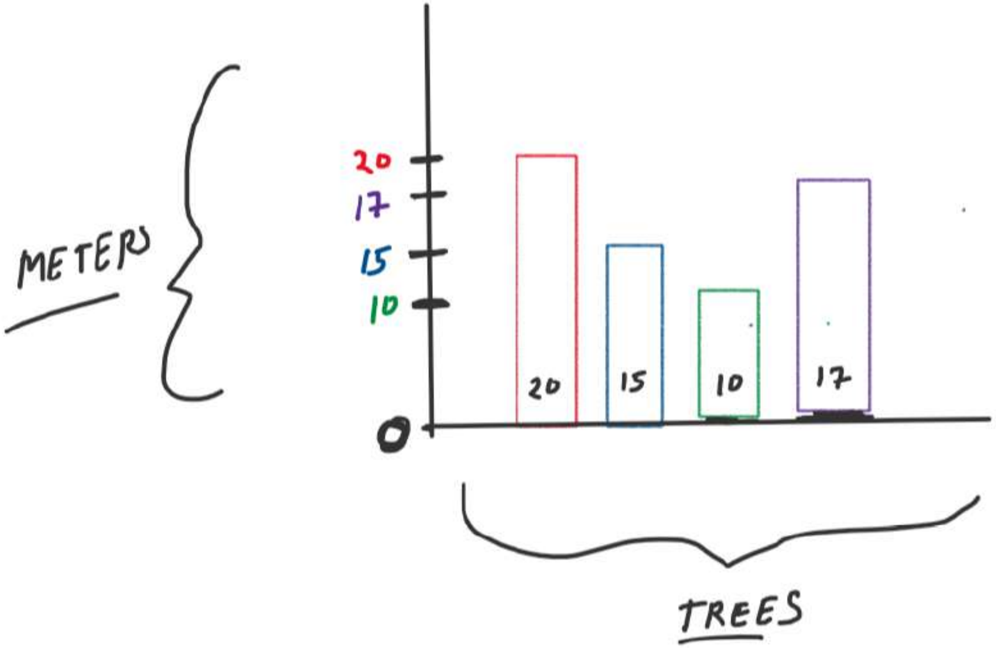
Let ⇒ H = 20 ← Not possible sawblade height

$$\text{Total wood} = \overset{T1}{(20-20)} + \overset{T2}{(15-20)} + \overset{T3}{(10-20)} + \overset{T4}{(17-20)}$$

$$= 0 + 0 + 0 + 0$$

$$= \boxed{0} \text{ meter wood}$$

DRY RUN



$M = 7$ and $N = 4$ $H = ?$

arr

20	15	10	17
0	1	2	3

STEP 01 Find maximum height of tree

$END = arr[0]$

$= 20$ M

max Height tree

SEARCH SPACE

0 START —— 20 END

Maximum No. of wood

Least No. of wood

STEP 02

Now, Applying Binary Search on search space

$\boxed{\text{Iteration : 1}}$

$M = 7$

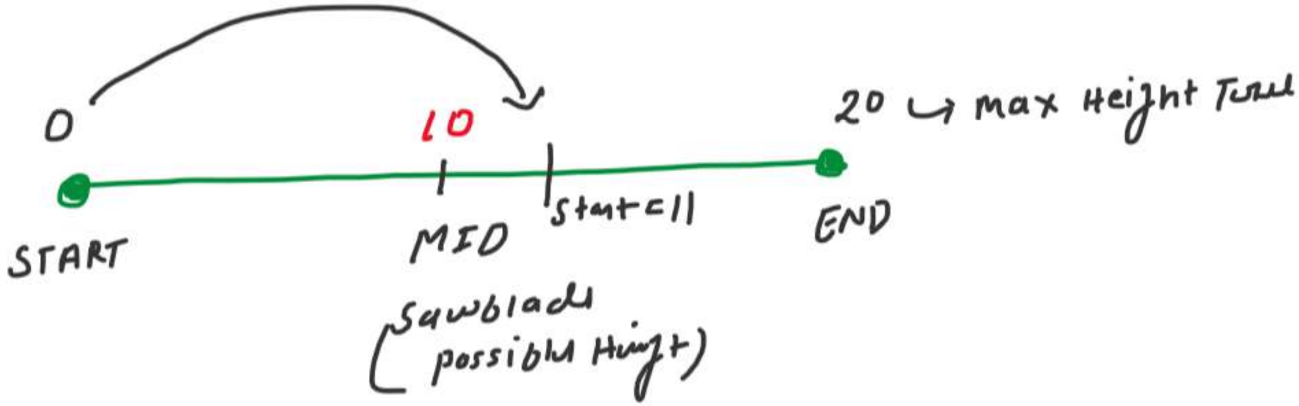
ANS (H) = ~~-1~~ 10

START = ~~0~~ 11

END = 20

$$MID = \frac{0+20}{2}$$

$= 10$



Total wood = (20 - 10) + (15 - 10) + (10 - 10) + (17 - 10)

= 10 + 5 + 0 + 7 = 22 Meters

possible sol.^n when

(Total wood $\geq$ M) TRUE

START = mid + 1 and ANS = mid

STEP 02

Now, Applying Binary Search on search space

$\boxed{\text{Itern : 2}}$

$M = 7$

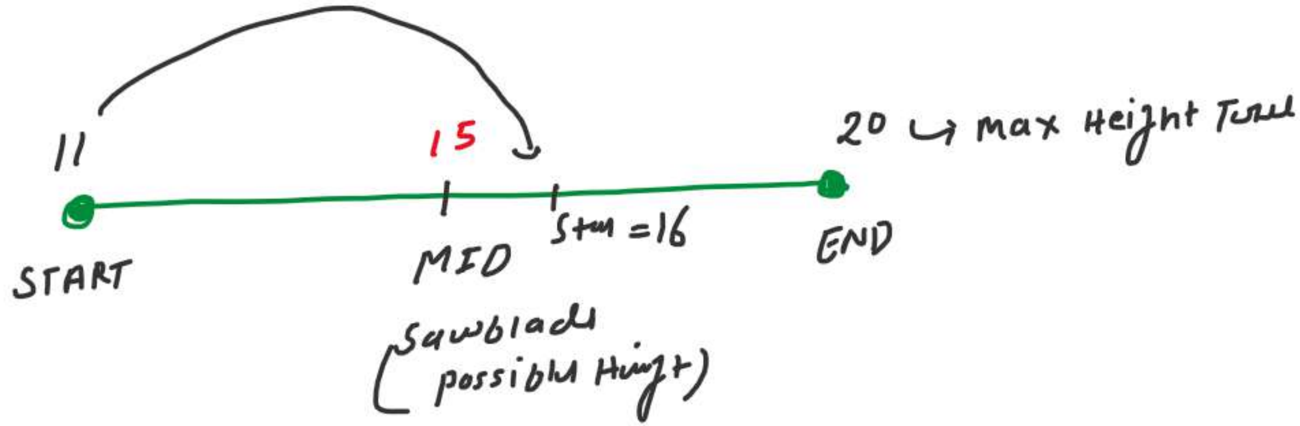
$ANS(H) = \cancel{10}\ 15$

$START = \cancel{11}\ 16$

$END = 20$

$$MID = \frac{11+20}{2}$$

$$= 15$$



Total wood = $(20-15) + (15-15) + (10-15) + (17-15)$

$= 5 + 0 + 0 + 2 = 7$ Meter

possible sol.^{n} when

(Total wood $\geq$ M) TRUE

START = mid + 1 and ANS = mid

STEP 02

Now, Applying Binary Search on search space

$\boxed{\text{Itr}: 3}$

$M = 7$

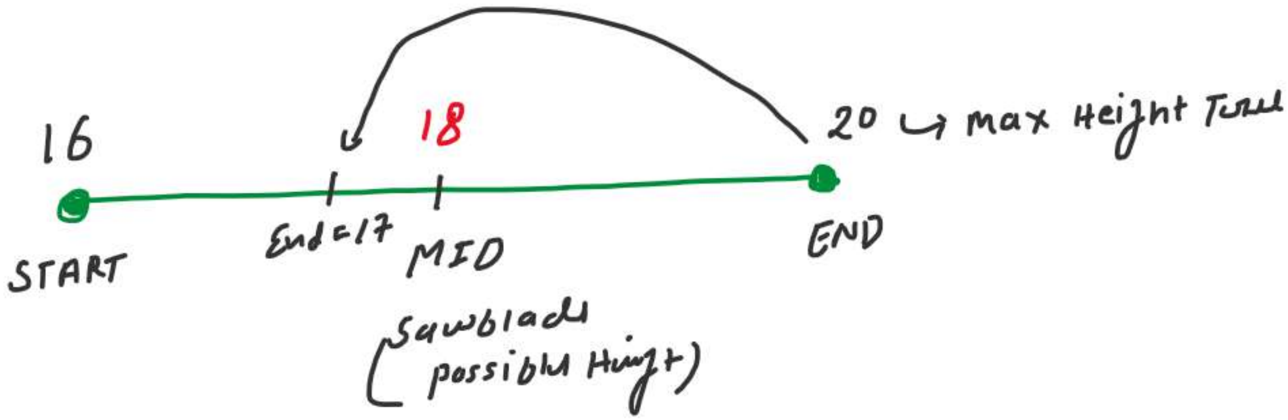
$ANS(H) = 15$

$START = 16$

$END = \cancel{20}\ 17$

$$MID = \frac{16+20}{2}$$

$$= 18$$



Total wood $= (20-18) + (15-18) + (10-18) + (17-18)$

$= 2 + 0 + 0 + 0 = 2$ Meter

possible sol.n when

(Total wood $\geq$ M) False

$END = mid - 1$

STEP 02

Now, Applying Binary Search on search space

Iter : 4

$M = 7$

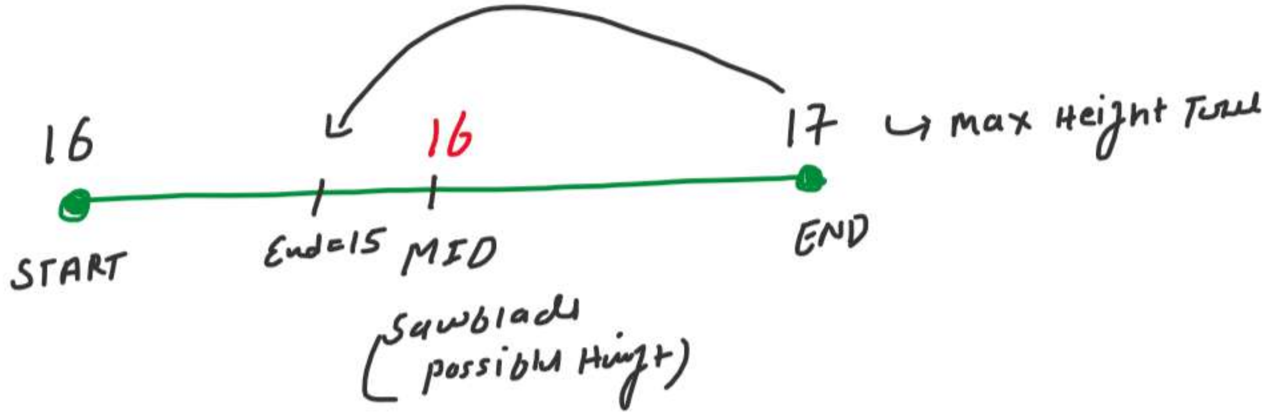
$ANS(H) = 15$

$START = 16$

$END = \cancel{17}\ 15$

$MID = \frac{16+17}{2}$

$= 16$



$Total\ wood = (20-16) + (15-16) + (10-16) + (17-16)$

$= 4 + 0 + 0 + 0 = 4$ Metre

possible sol.^{n} when

(Total wood $\geq$ M) False

$END = mid - 1$

STEP 02

Now, Applying Binary Search on search space

$\boxed{\text{Iter}: 5}$

M = 7

ANS(H) = 15

START = 16

END = 15 } STOP → start > end

Output { possible (least) maximum height of sawblade is = H ⇒ 15

```cpp
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;

// Step 03: create predicate function isPossibbleSol()
bool isPossibleSol(vector<long long int> trees,long long int m,long long int mid){
    long long int totalWood = 0;
    for(int i=0; i<trees.size(); i++){
        if(trees[i] > mid){
            totalWood = totalWood + (trees[i]-mid);
            if(totalWood >= m){
                return true;
            }
        }
    }
    return false;
}

// Step 02: Now, Applying Binary Search on search space BinarySearch()
int BinarySearch(vector<long long int> trees,long long int m,long long int end){
    long long int start = 0, ansH = -1, mid = start + (end - start)/2;

    while(start<=end){
        // Step 03: create predicate function isPossibbleSol()
        if(isPossibleSol(trees,m,mid)){
            ansH = mid;
            start = mid + 1;
        }
        else{
            end = mid - 1;
        }
        mid = start + (end - start)/2;
    }
    return ansH;

}


int maxSawBladeHeight(vector<long long int> trees,long long int m){
    // Step 01: Find maximum height of tree
    long long int end = *max_element(trees.begin(), trees.end());

    // Step 02: Now, Applying Binary Search
    long long int sawbladeMaxHeight = BinarySearch(trees,m,end);

    return sawbladeMaxHeight;
}

int main(){
    long long int n, m;
    cin>>n>>m;
    vector<long long int> trees;
    while(n--){
        long long int height;
        cin>> height;
        trees.push_back(height);
    }

    cout <<maxSawBladeHeight(trees,m);
   return 0;
}
```