1873F - Money Trees

Devan Ferrel

8 April 2024

1 Problem

Problem Description : https://codeforces.com/contest/1873/problem/F

2 Objective

Given n trees where the i tree has a_i fruit and height of h_i, find the maximum contiguous subarray that sastify h_i is divisible by h_{i+1} and we can collect a_i to a_r fruits if only the total of the fruits we collected doesnt exceed k fruits

3 Solution

Firstly, we can check if there is atleast one tree that sastifies $a_i \leq k$. If there is no tree that sastifies this condition, we can safely say there is no such subarray that sastifies the objective requirement, which is that we can't collects more than k fruits

Now after that, we can use sliding window apporach for this problem. We create 2 pointer initialy they point at index 0 and then we create variable that holds the current sum of all fruits that we traveled.

We check if h_r is divisible by h_{r+1} and adding a_{r+1} to current sum is still less than k, then we add a_{r+1} to the current sum

But, dont forget edge case that if h_r is still divisble by h_{r+1} but when we add a_{r+1} to the current sum, the current sum will exceed k, we just have to subtract the value of a_l from the current sum and add a_{r+1} to the current sum. This is to counter edge case where we cant take more than k fruits anymore but still h_i is divisible by h_{i+1}, because we can get more length by doing this.

After that, we can safely say h_i is not divisible by h_{i+1} and we must calculate the max length by doing $max(len, r-l+1)$ where len is the current max length of the subarray. Move the pointer of l to $r+1$ and assign a_i to current sum.

Dont forget to increment the right pointer r

4 Code

```java
import java.util.Scanner;

public class main {
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int t = in.nextInt();
        while(t-- > 0) {
            int n = in.nextInt();
            int k = in.nextInt();

            long[] A = new long[n];
            long[] B = new long[n];

            boolean f = false;

            for(int i = 0; i < n; i++) {
                A[i] = in.nextLong();
                if (A[i] <= k) f = true;
            }
            for(int i = 0; i < n; i++) {
                B[i] = in.nextLong();
            }

            if(!f) {
                System.out.println(0);
                continue;
            }

            int r = 0, l = 0,len=0; long tmp=A[0];
            while(r < n - 1) {
                if(B[r] % B[r+1] == 0 && tmp + A[r+1] <= k) {
                    tmp += A[r+1];
                } else if(B[r] % B[r+1] == 0) {
                    tmp += A[r+1];
                    tmp -= A[l++];
                } else {
                    len = Math.max(len, r - l + 1);
                    l = r + 1;
                    tmp = A[l];
                }

                r++;
```

```
            }
            len = Math.max(len, r - l + 1);
            System.out.println(len);
        }
    }
}
```