Using List and Dictionaries

```
import os
import sys

# Complete the solve function below.
def solve(arr,c=0):
    arr.append(2**64)
    idx, routes = 0, 0
    stack, m = [], {}
    while idx < len(arr):
        if stack == [] or arr[idx] <= stack[-1]:
            stack.append(arr[idx])
            if arr[idx] in m:
                m[arr[idx]] += 1
            else:
                m[arr[idx]] = 1
            idx += 1
        else:
            top = stack.pop()
            if top in m and top < arr[idx]:
                routes += m[top] * (m[top] - 1)
                del m[top]
    return(routes)
```

Using Stack

```python
import os
import sys
import math

# Complete the solve function below.
def empty(stk):
    return len(stk) == 0

def top(stk):
    if empty(stk):
        return None
    return stk[-1]

def fixStack(stk, h, debug=False):
    prev = None
    count = 0
    res = 0
    while not(empty(stk)) and h > top(stk):
        cur = stk.pop()
        if prev == None:
            count = 1
            prev = cur
        elif prev != None and cur == prev:
            count += 1
            prev = cur
        elif prev != None and cur != prev:
            res += max(0, count * (count - 1))
            count = 1
            prev = cur
    res += max(0, count*(count-1))
    return stk,res
```

```
def solve(H, debug=False):
    stk = []
    res = 0
    for i,h in enumerate(H):
        if empty(stk) or h <= top(stk):
            stk.append(h)
        elif h > top(stk):
            stk, paths = fixStack(stk, h, debug)
            stk.append(h)
            res += paths
    stk, rem = fixStack(stk, math.inf, debug=False)
    res += rem
    return res
```