Dynamic Programming: Good Sequence

There are **N** towers. The height of the i^{th} tower is H_i. A sequence of towers is **Good** if there are not any two adjacent towers that have the same heights. i.e for every **i** ($2 \leq i \leq N$) $H_{i-1} \neq H_i$ condition must hold.
You can increase the height of i^{th} tower but it will cost M_i, to increase the height by **1**.
Find the minimum cost to make the sequence good.

Note
You can increase the height of a tower any number of times you want. You have to just minimize the cost to make the sequence of towers **Good.**

Function Description
In the provided code snippet, implement the `minCost(...)` method and print the minimum cost to make the sequence good. You can write code in the given space below the phrase "`WRITE YOUR LOGIC HERE`".

There will be multiple test cases running so the Input and Output should match exactly as provided.
The base Output variable `result` is set to a default value of `-404` which can be modified. Additionally, you can add or remove these output variables.

Input Format
The first line of input consists of an integer **N.**
The next **N** lines of input contain the description of towers. The i^{th} line contains H_i and M_i - the height of i^{th} tower and the cost to increase the height of the tower by 1 respectively.

Sample Input

```
3                    --denotes N.
2 4                  --denotes two integers Hi and Mi
2 1                  --denotes two integers Hi and Mi
3 5                  --denotes two integers Hi and Mi
```

Constraints
$1 \leq N \leq 10^5$
$1 \leq H_i , M_i \leq 10^9$

Output Format
The output contains a single integer denoting the minimum cost to make a sequence Good.

Sample Output

```
2
```

Explanation
In the sample input, you have to increase the height of the second tower by 2.
Cost = $M_2 + M_2 = 1 + 1 = 2$.
Hence, the minimum cost to make the sequence Good is ***2***.

```
function minCost(N,H,M) {
    //this is default OUTPUT. You can change it.
    var result =-404;
    //write your Logic here:
    return result;
}
// INPUT [uncomment & modify if required]
var temp = gets().trim('\n').split(/\n|\s/)
var N = parseInt(temp[0]);
var H = [];
for(var i = 1; i < 1+N; i++) {
    H.push(parseInt(temp[i]));
}
var M = [];
for(var i = 1+N; i < 1+N+N; i++) {
    M.push(parseInt(temp[i]));
}

// OUTPUT [uncomment & modify if required]
console.log(minCost(N,H,M));
```

Input

```
4
1 7
3 3
2 6
1000000000 2
```

Expected Output

```
0
```

Input

```
3
2 3
2 10
2 6
```

Expected Output

```
9
```

Input

3

2 4

2 1

3 5

Expected Output

2