```cpp
//
//   There are N flowers arranged in a row. For each i (1≤i≤N), the height and the beauty of
//   the i-th flower from the left is h[i] and a[i], respectively.
//   Here, h[1], h[2], ... h[N] are all distinct.
//   Taro is pulling out some flowers so that the following condition is met:
//      - The heights of the remaining flowers are monotonically increasing from left to right.
//   Find the maximum possible sum of the beauties of the remaining flowers.
//
//   Constrains:
//     h[i] ⩽ N
//
//   Time Complexity: O(N * log(N))
//


#include <bits/stdc++.h>
#define ll long long

using namespace std;

// Max Fenwick Tree
struct FenwickTree {
    vector<ll> fwt;

    FenwickTree(int n) {
        fwt.resize(n, 0);
    }

    void maxFWT(int ind, ll val = 1) {
        for (ind++; ind < fwt.size(); ind+=ind&-ind)
            fwt[ind] = max(fwt[ind], val);
    }

    ll getFWT(int ind) {
        ll s = 0;
        for (ind++; ind > 0; ind-=ind&-ind)
            s = max(s, fwt[ind]);
        return s;
    }
};

int main() {
    int n;
    cin >> n;

    vector<ll> h(n), a(n);
    for (int i = 0; i < n; i++) scanf("%d", &h[i]);
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);

    FenwickTree tree(n+10);

    for (int i = 0; i < n; i++) {
        // query best = max (1 ⇒ h[i] - 1)
        // update at h[i] = best + a[i];

        ll best = tree.getFWT(h[i]); // get maximum
        ll new_best = best + a[i];
        tree.maxFWT(h[i], new_best);
    }

    cout << tree.getFWT(n) << endl;
    return 0;
}
```