Highest

In an alternate universe, Vlad is stuck inside a futuristic version of the Poenari Fortress, now spanning n floors, numbered 0 through $n-1$. From each floor i ($0 \leq i \leq n-1$), he can only go up, either by taking the stairs and paying 1 drop of blood (this is the currency that vampires use to pay in Romania), or by turning into a bat and traversing the vents, for which he has to pay 2 drops of blood. The stairs can take him up to $v[i]$ floors upwards, while the vents span up to $w[i]$ floors upwards, where v and w are two given arrays: $v = v[0], v[1], \ldots, v[n-1]$ and $w = w[0], w[1], \ldots, w[n-1]$.

Formally, from floor i ($0 \leq i \leq n-1$), Vlad can go:

- anywhere from floor $i+1$ to floor $i + v[i]$ without exceeding $n-1$, for a cost of 1
- anywhere from floor $i+1$ to floor $i + w[i]$ without exceeding $n-1$, for a cost of 2

Furthermore, his brothers Radu and Mircea proposed m scenarios for Vlad, each one consisting of two floors A and B ($A \leq B$). Vlad has to answer their m questions: what is the least amount of blood that he has to sacrifice to get from floor A to floor B?

Implementation Details

You will have to implement the function solve:

```
std::vector<int> solve(std::vector<int> &v, std::vector<int> &w,
    std::vector<std::pair<int,int>> &queries);
```

- Receives the vectors v, the heights of the flights of stairs, and w, the heights of the vent systems, starting at each floor, both of them of size n.
- Also receives the queries, a vector of pairs of size m. Each pair contains A and B as described in the statement.
- Returns a vector of size m, consisting of the answers to the m queries.

Constraints

- $1 \leq n, m \leq 500\,000$.
- $1 \leq v[i], w[i] \leq n$ for all $0 \leq i \leq n-1$.
- $0 \leq A \leq B \leq n-1$ for all queries.

Subtasks

1. (5 points) $1 \le n \le 300,\ 1 \le m \le 500\,000$
2. (7 points) $1 \le n \le 3\,000,\ 1 \le m \le 3\,000$
3. (11 points) $1 \le n \le 20\,000,\ 1 \le m \le 20\,000$
4. (44 points) $1 \le n \le 200\,000,\ 1 \le m \le 200\,000$
5. (8 points) $1 \le n \le 500\,000,\ 1 \le m \le 500\,000,\ v[i] \le v[j]$ and $w[i] \le w[j]$ for all $0 \le i < j \le n-1$
6. (25 points) No further restrictions.

Examples

Example 1

Consider the following call:

```
solve({2, 3, 1, 1, 1, 1, 2}, {3, 4, 1, 2, 1, 2, 2},
    {{0, 4}, {0, 5}, {0, 6}})
```

Here we have $n = 7$ and 3 queries, $v = [2, 3, 1, 1, 1, 1, 2]$ and $w = [3, 4, 1, 2, 1, 2, 2]$.

For the first query $(0, 4)$, Vlad has to make two 1-cost jumps: 0 to 1 (even though he can jump to 2, floor 1 will then take him further), then 1 to 4. Total cost: $1 + 1 = 2$.

For the second query $(0, 5)$, there are 2 optimal paths: 0 to 1 (cost 1), 1 to 4 (cost 1), 4 to 5 (cost 1); the second path is 0 to 1 (cost 1), 1 to 5 (cost 2). Total cost: $1 + 1 + 1 = 1 + 2 = 3$.

For the third query $(0, 6)$, one example path of cost 4 is 0 to 1 (cost 1), 1 to 5 (cost 2), 5 to 6 (cost 1). Total cost: $1 + 2 + 1 = 4$

So the vector that the function will return must be:

```
{2, 3, 4}
```

Example 2

Consider the following call:

```
solve({1, 1, 1, 2, 3, 2, 1, 1, 2, 3}, {2, 4, 1, 4, 1, 4, 1, 3, 2, 3},
    {{3, 9}, {0, 9}, {0, 7}, {0, 4}, {3, 5}})
```

These are the optimal paths for the queries:

(3,9): 3 to 5 (cost 1), 5 to 9 (cost 2) $\Longrightarrow$ total: 3

(0,9): 0 to 1 (cost 1), 1 to 5 (cost 2), 5 to 9 (cost 2) $\Longrightarrow$ total: 5

(0,7): 0 to 1 (cost 1), 1 to 5 (cost 2), 5 to 7 (cost 1) $\Longrightarrow$ total: 4

(0,4): 0 to 1 (cost 1), 1 to 4 (cost 2) $\Longrightarrow$ total: 3

(3,5): 3 to 5 (cost 1) $\Longrightarrow$ total: 1

So the vector that the function will return must be:

```
{3, 5, 4, 3, 1}
```

Sample grader

The sample grader reads the input in the following format:

- line 1: n
- line 2: $v[0]\ v[1] \ldots v[n-1]$
- line 3: $w[0]\ w[1] \ldots w[n-1]$
- line 4: m
- line $5+i (0 \leq i \leq m-1)$: $A\ B$

and outputs m lines, the result of the call to `solve`.