Das Höchste

In einem alternativen Universum sitzt Vlad in einer futuristischen Version der Festung Poenari fest, die sich über n Etagen erstreckt, die von 0 bis $n-1$ nummeriert sind. Von jedem Stockwerk i ($0 \leq i \leq n-1$) kann er nur nach oben gehen, und zwar indem er entweder die Treppe nimmt und 1 Blutstropfen bezahlt (das ist die Währung, mit der Vampire in Rumänien bezahlen), oder indem er sich in eine Fledermaus verwandelt und die Lüftungsschächte durchquert, wofür er 2 Blutstropfen bezahlen muss. Die Treppe kann ihn bis zu v_i Etagen nach oben bringen, während die Lüftungsschächte bis zu w_i Etagen nach oben reichen, wobei v und w zwei gegebene Arrays sind: $v = v[0], v[1], \ldots, v[n-1]$ und $w = w[0], w[1], \ldots, w[n-1]$.

Formal kann Vlad von der Etage i ($0 \leq i \leq n-1$) aus

- zu einer beliebigen der Etagen $i+1$ bis $i+v_i$ (inklusive) gehen (ohne $n-1$ zu überschreiten), mit Kosten von 1, oder
- zu einer beliebigen der Etagen $i+1$ bis $i+w_i$ (inklusive) gehen (ohne $n-1$ zu überschreiten), mit Kosten von 2.

Vlads Brüder Radu und Mircea haben ihm m Szenarien vorgelegt, bei denen jeweils zwei Stockwerke A und B ($A \leq B$) gegeben sind. Vlad soll nun für jedes der m Szenarien die Frage beantworten: Wie viel Blut muss er mindestens zahlen, um von Stockwerk A zu Stockwerk B zu gelangen?

Implementierungsdetails

Implementiere die Funktion `solve`:

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

- Der Vektor `v` und `w` haben beide die Länge n. Dabei gibt `v` die Höhen der in jedem Stockwerk beginnenden Treppenläufe an, und `w` die Höhen der Lüftungsanlagen.
- Der Vektor `queries` hät Länge m und enthält die Abfragen. Jede Abfrage ist ein Paar aus den Werten A und B wie in der Aufgabenstellung beschrieben.
- Die Funktion muss einen Vektor der Länge m zurückgeben, der aus den Antworten auf die m Abfragen besteht.

Beschränkungen

- $1 \leq n, m \leq 500\,000$.
- $1 \leq v_i, w_i \leq n$ für alle $0 \leq i \leq n-1$.
- $0 \leq A \leq B \leq n-1$ für alle Abfragen.

Teilaufgaben

1. (5 Punkte) $1 \leq n \leq 300,\ 1 \leq m \leq 500\,000$
2. (7 Punkte) $1 \leq n \leq 3\,000,\ 1 \leq m \leq 3\,000$
3. (11 Punkte) $1 \leq n \leq 20\,000,\ 1 \leq m \leq 20\,000$
4. (44 Punkte) $1 \leq n \leq 200\,000,\ 1 \leq m \leq 200\,000$
5. (8 Punkte) $1 \leq n \leq 500\,000,\ 1 \leq m \leq 500\,000$, $v_i \leq v_j$ und $w_i \leq w_j$ für alle $0 \leq i < j \leq n-1$
6. (25 Punkte) Keine weiteren Beschränkungen.

Beispiele

Beispiel 1

Betrachte den folgenden Aufruf:

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

Hier haben wir $n = 7$ Etagen, $m = 3$ Abfragen, $v = [2, 3, 1, 1, 1, 1, 2]$ und $w = [3, 4, 1, 2, 1, 2, 2]$.

Für die erste Abfrage $(0, 4)$ muss Vlad zwei Kosten-1-Sprünge machen: 0 nach 1 (obwohl er nach 2 springen könnte; Etage 1 bringt ihn weiter), dann 1 nach 4. Gesamtkosten: $1 + 1 = 2$.

Für die zweite Anfrage $(0, 5)$ gibt es 2 optimale Wege: Zum einen 0 bis 1 (Kosten 1), 1 bis 4 (Kosten 1), 4 bis 5 (Kosten 1); oder alternativ auch 0 bis 1 (Kosten 1), 1 bis 5 (Kosten 2). Gesamtkosten: $1 + 1 + 1 = 1 + 2 = 3$.

Für die dritte Abfrage $(0, 6)$ ist ein Beispielpfad mit den Kosten 4: 0 bis 1 (Kosten 1), 1 bis 5 (Kosten 2), 5 bis 6 (Kosten 1). Gesamtkosten: $1 + 2 + 1 = 4$

Der Vektor, den die Funktion zurückgibt, muss also lauten:

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

Beispiel 2

Betrachte den folgenden Aufruf:

```
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}})
```

Dies sind die optimalen Wege für die Abfragen:

(3,9): 3 bis 5 (Kosten 1), 5 bis 9 (Kosten 2) $\implies$ insgesamt: 3

(0,9): 0 bis 1 (Kosten 1), 1 bis 5 (Kosten 2), 5 bis 9 (Kosten 2) $\implies$ insgesamt: 5

(0,7): 0 bis 1 (Kosten 1), 1 bis 5 (Kosten 2), 5 bis 7 (Kosten 1) $\implies$ insgesamt: 4

(0,4): 0 bis 1 (Kosten 1), 1 bis 4 (Kosten 2) $\implies$ insgesamt: 3

(3,5): 3 bis 5 (Kosten 1) $\implies$ insgesamt: 1

Der Vektor, den die Funktion zurückgibt, muss also lauten:

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

Beispiel-Grader

Der Beispiel-Grader liest die Eingabe im folgenden Format:

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

Er gibt m Zeilen aus: das Ergebnis des Aufrufs von `solve`.