Високо, високо, високо

В алтернативна вселена, Влад е заключен във футуристична версия на Пойенарската крепост, обхващаща n етажа, номерирани с числата от 0 до $n-1$. От всеки етаж i ($0 \leq i \leq n-1$) той може да се качва единствено нагоре, използвайки един от два метода. Ако използва стълбите, Влад трябва да плати 1 капка кръв (официалната валута на вампирите в Румъния), а ако се превърне в прилеп и мине през вентилационната шахта, то той трябва да плати 2 капки кръв. Стълбите могат да го качат до някой от следващите v_i етажа, а вентилационната шахта до някой от следващите w_i етажа, където v и w са два дадени масива: $v = v[0], v[1], \ldots, v[n-1]$ и $w = w[0], w[1], \ldots, w[n-1]$.

Формално, от етаж i ($0 \leq i \leq n-1$), Влад може да се придвижи до:

- който и да е етаж с номер от $i+1$ до $i+v_i$ без да преминава над етаж $n-1$, за цена 1
- който и да е етаж с номер от $i+1$ до $i+w_i$ без да преминава над етаж $n-1$, за цена 2

Сега неговите братя Раду и Мирчя предлагат m различни сценарии на Влад, всеки от които се състои от два етажа A и B ($A \leq B$). Влад трябва да отговори на всеки от тези m въпроса: какво е най-малкото количество кръв нужно за да се стигне от етаж A до етаж B?

Детайли по имплементацията

Трябва да имплементирате следната функция:

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

- Приема два вектора с дължина n, където v задава височините на стълбите, достъпни от всеки етаж, а w - височините на вентилационните системи, достъпни от всеки етаж.
- Приема също заявките като вектор с дължина m, състоящ се от двойки числа. Всяка двойка съдържа числата A и B описани в условието.
- Функцията трябва да връща вектор с размер m, съдържащ отговорите на всяка заявка.

Ограничения

- $1 \leq n, m \leq 500\,000$.
- $1 \leq v_i, w_i \leq n$ за всяко $0 \leq i \leq n-1$.
- $0 \leq A \leq B \leq n-1$ за всяка заявка.

Подзадачи

1. (5 точки) $1 \leq n \leq 300,\ 1 \leq m \leq 500\,000$
2. (7 точки) $1 \leq n \leq 3\,000,\ 1 \leq m \leq 3\,000$
3. (11 точки) $1 \leq n \leq 20\,000,\ 1 \leq m \leq 20\,000$
4. (44 точки) $1 \leq n \leq 200\,000,\ 1 \leq m \leq 200\,000$
5. (8 точки) $1 \leq n \leq 500\,000,\ 1 \leq m \leq 500\,000$, $v_i \leq v_j$ и $w_i \leq w_j$ за всеки $0 \leq i < j \leq n - 1$
6. (25 точки) Няма допълнителни ограничения.

Примери

Пример 1

Да разгледаме следното извикване:

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

Тук имаме $n = 7$ и 3 заявки, $v = [2, 3, 1, 1, 1, 1, 2]$ и $w = [3, 4, 1, 2, 1, 2, 2]$.

За първата заявка $(0, 4)$, Влад може да използва стълби (с цена 1) два пъти: от етаж 0 до 1 (въпреки че може да стигне до етаж 2, етаж 1 е по-добрият избор); след което от 1 до 4. Общата цена е $1 + 1 = 2$.

За втората заявка $(0, 5)$ има две оптимални решения: първото е от 0 до 1 (цена 1), от 1 до 4 (цена 1), от 4 до 5 (цена 1); второто е от 0 до 1 (цена 1), от 1 до 5 (цена 2). Общата цена е $1 + 1 + 1 = 1 + 2 = 3$.

За третата заявка $(0, 6)$, едно примерно решение с цена 4 е от етаж 0 до 1 (цена 1), от 1 до 5 (цена 2), от 5 до 6 (цена 1). Общата цена е $1 + 2 + 1 = 4$

Съответно векторът, който функцията трябва да върне, е:

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

Пример 2

Да разгледаме следното извикване:

```
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}})
```

Оптималните придвижвания за заявките са:

(3,9): от 3 до 5 (цена 1), от 5 до 9 (цена 2) $\Longrightarrow$ общо: 3

(0,9): от 0 до 1 (цена 1), от 1 до 5 (цена 2), от 5 до 9 (цена 2) $\Longrightarrow$ общо: 5

(0,7): от 0 до 1 (цена 1), от 1 до 5 (цена 2), от 5 до 7 (цена 1) $\Longrightarrow$ общо: 4

(0,4): от 0 до 1 (цена 1), от 1 до 4 (цена 2) $\Longrightarrow$ общо: 3

(3,5): от 3 до 5 (цена 1) $\Longrightarrow$ общо: 1

Съответно векторът, който функцията трябва да върне, е:

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

Локален грейдър

Локалният грейдър чете от стандартния вход в следния формат:

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

и извежда на стандартния изход m реда, резултатите от извикванията на `solve` за всяка заявка.