Najwyższy

W alternatywnym wszechświecie Wład utknął w futurystycznej wersji Twierdzy Poenari. Twierdza ma n pięter ponumerowanych od 0 do $n-1$. Z każdego piętra i $(0 \leq i \leq n-1)$ może on poruszyć sie w górę używając schodów lub zamieniając się w nietoperza i przelatując przez szyby wentylacyjne. Użycie schodów kosztuje go 1 kroplę krwi (waluta używana przez wampiry) i może go przenieść o co najwyżej $v[i]$ pięter wyżej. Natomiast zamiana w nietoperza kosztuje 2 krople krwi i może się w ten sposób przenieść o co najwyżej w_i pięter wyżej. Tablice v i w są dane: $v = v[0], v[1], \ldots, v[n-1]$ i $w = w[0], w[1], \ldots, w[n-1]$.

Formalnie, dla dowolnego i ($0 \leq i \leq n-1$), Wład może przenieść się:

- gdziekolwiek pomiędzy poziomami od $i+1$ do $\min(i+v[i], n-1)$ kosztem 1
- gdziekolwiek pomiędzy poziomami od $i+1$ do $\min(i+w[i], n-1)$ kosztem 2

Ponadto, jego bracia Radu i Mircea zadali Władowi m zapytań. Każde z nich składa się z pary liczb A i B ($A \leq B$). Wład musi odpowiedzieć na każde z tych m zapytań: jaka jest minimalna ilość krwi, którą musi poświęcić, żeby dostać się z poziomu A do poziomu B?

Szczegóły implementacyjne

Musisz zaimplementować następującą funkcję:

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

- Funkcja otrzymuje wektory v i w o długości n, oznaczające zasięgi ruchów w górę o kosztach 1 i 2.
- Funkcja dostaje też zapytania, będące wektorem $queries$, składającym się z m par (A, B) opisanych z treści.
- Powinna zwrócić wektor długości m, składający się z odpowiedzi na m zapytań.

Ograniczenia

- $1 \leq n, m \leq 500\,000$.
- $1 \leq v_i, w_i \leq n$ dla $1 \leq i \leq n$.
- $0 \leq A \leq B \leq n-1$ dla wszystkich zapytań.

Pozadania

1. (5 punktów) $1 \leq n \leq 300,\ 1 \leq m \leq 500\,000$
2. (7 punktów) $1 \leq n \leq 3\,000,\ 1 \leq m \leq 3\,000$
3. (11 punktów) $1 \leq n \leq 20\,000,\ 1 \leq m \leq 20\,000$
4. (44 punkty) $1 \leq n \leq 200\,000,\ 1 \leq m \leq 200\,000$
5. (8 punktów) $1 \leq n \leq 500\,000,\ 1 \leq m \leq 500\,000$ oraz $v_i \leq v_j\ \forall\, i \leq j$
6. (25 punktów) Brak dodatkowych ograniczeń.

Przykłady

Przykład 1

Rozważmy następujące wywołanie:

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

Mamy $n = 7$ i 3 zapytania, $v = [2, 3, 1, 1, 1, 1, 2]$ i $w = [3, 4, 1, 2, 1, 2, 2]$.

W pierwszym zapytaniu $(0, 4)$ Wład może po prostu wykonać dwa skoki o kosztach równych 1: z 0 do 1 (mimo że może przenieść się na poziom 2, poziom 1 zabierze go dalej), następnie z 1 do 4. Łączny koszt: $1 + 1 = 2$.

W drugim zapytaniu $(0, 5)$ są dwie optymalne ścieżki o kosztach $1 + 1 + 1 = 1 + 2 = 3$: z 0 do 1 (o koszcie 1), z 1 do 4 (o koszcie 1), z 4 do 5 (o koszcie 1); druga ścieżka jest z 0 do 1 (o koszcie 1), z 1 do 5 (o koszcie 2).

W trzecim zapytaniu $(0, 6)$ przykładowa ścieżka o koszcie 4 to z 0 do 1 (o koszcie 1), z 1 do 5 (o koszcie 2), z 5 do 6 (o koszcie 1). Łączny koszt: $1 + 2 + 1 = 4$

Zatem, funkcja musi zwrócić wektor:

```
[2, 3, 4]
```

Przykład 2

Rozważmy następujące wywołanie:

```
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}])
```

Optymalne ścieżki w zależności od zapytania to:

- $(3{,}9)$: z 3 do 5 (o koszcie 1), z 5 do 9 (o koszcie 2) $\implies$ łącznie: 3
- $(0{,}9)$: z 0 do 1 (o koszcie 1), z 1 do 5 (o koszcie 2), 5 do 9 (o koszcie 2) $\implies$ łącznie: 5
- $(0{,}7)$: z 0 do 1 (o koszcie 1), z 1 do 5 (o koszcie 2), 5 do 7 (o koszcie 1) $\implies$ łącznie: 4
- $(0{,}4)$: z 0 do 1 (o koszcie 1), z 1 do 4 (o koszcie 2) $\implies$ łącznie: 3
- $(3{,}5)$: z 3 do 5 (o koszcie 1) $\implies$ łącznie: 1

Funkcja musi zatem zwrócić wektor:

```
[3, 5, 4, 3, 1]
```

Przykładowa sprawdzaczka

Przykładowa sprawdzaczka przyjmuje następujący format wejścia:

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

i wypisuje m linii, wynik wywołania funkcji `solve`.