Le Plus Haut

Dans un univers parallèle, Vlad est coincé dans une version futuriste de la forteresse de Poenari, qui s'étend maintenant sur n étages, numérotés de 0 à $n-1$. Depuis chaque étage i ($0 \leq i < n-1$), il ne peut monter que vers le haut, soit en prenant les escaliers et en payant 1 goutte de sang (c'est la monnaie que les vampires utilisent pour payer en Roumanie), soit en se transformant en chauve-souris et en traversant les conduits d'aération, ce qui lui coûte 2 gouttes de sang. Les escaliers peuvent le faire monter jusqu'à $v[i]$ étages vers le haut, tandis que les conduits d'aération s'étendent jusqu'à $w[i]$ étages vers le haut, où v et w sont deux tableaux donnés : $v = v[0], v[1], \ldots, v[n-1]$ et $w = w[0], w[1], \ldots, w[n-1]$.

Formellement, depuis l'étage i ($0 \leq i < n-1$), Vlad peut aller :

- vers n'importe quel étage entre $i+1$ et $i+v[i]$ (inclus), sans dépasser $n-1$, pour un coût de 1
- vers n'importe quel étage entre $i+1$ et $i+w[i]$ (inclus), sans dépasser $n-1$, pour un coût de 2

De plus, ses frères Radu et Mircea ont proposé m scénarios à Vlad, chacun consistant en deux étages A et B ($A \leq B$). Vlad doit répondre à leurs m questions : quelle est la quantité minimale de sang qu'il doit sacrifier pour aller de l'étage A à l'étage B ?

Détails d'implémentation

Vous devrez implémenter la fonction solve :

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

- Reçoit les vecteurs v, les hauteurs des volées d'escaliers, et w, les hauteurs des systèmes de conduits d'aération, mesurées à partir de chaque étage, tous deux de taille n.
- Reçoit également les requêtes, un vecteur de paires de taille m. Chaque paire contient A et B comme décrit dans l'énoncé.
- Renvoie un vecteur de taille m, contenant les réponses aux m requêtes.

Limites

- $1 \leq n, m \leq 500\,000$.

- $1 \leq v[i], w[i] \leq n$ pour tout $0 \leq i \leq n-1$.
- $0 \leq A \leq B \leq n-1$ pour toutes les requêtes.

Sous-tâches

1. (5 points) $1 \leq n \leq 300,\ 1 \leq m \leq 500\,000$
2. (7 points) $1 \leq n \leq 3\,000,\ 1 \leq m \leq 3\,000$
3. (11 points) $1 \leq n \leq 20\,000,\ 1 \leq m \leq 20\,000$
4. (44 points) $1 \leq n \leq 200\,000,\ 1 \leq m \leq 200\,000$
5. (8 points) $1 \leq n \leq 500\,000,\ 1 \leq m \leq 500\,000,$ avec $v[i] \leq v[j]$ et $w[i] \leq w[j]$ pour tout $0 \leq i < j \leq n-1$
6. (25 points) Aucune autre restriction.

Exemples

Exemple 1

Considérons l'appel suivant :

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

Ici, nous avons $n = 7$ et 3 requêtes, avec $v = [2, 3, 1, 1, 1, 1, 2]$ et $w = [3, 4, 1, 2, 1, 2, 2]$.

Pour la première requête $(0, 4)$, Vlad doit effectuer deux sauts à coût 1 : de 0 à 1 (même s'il peut sauter jusqu'à 2, l'étage 1 lui permettra ensuite d'aller plus loin), puis de 1 à 4. Coût total : $1 + 1 = 2$.

Pour la deuxième requête $(0, 5)$, il existe 2 chemins optimaux : de 0 à 1 (coût 1), de 1 à 4 (coût 1), puis de 4 à 5 (coût 1) ; le deuxième chemin est de 0 à 1 (coût 1), puis de 1 à 5 (coût 2). Coût total : $1 + 1 + 1 = 1 + 2 = 3$.

Pour la troisième requête $(0, 6)$, un exemple de chemin de coût 4 est : de 0 à 1 (coût 1), de 1 à 5 (coût 2), puis de 5 à 6 (coût 1). Coût total : $1 + 2 + 1 = 4$.

Ainsi, le vecteur que la fonction doit retourner est :

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

Exemple 2

Considérons l'appel suivant :

```
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}})
```

Voici les chemins optimaux pour les requêtes :

$(3{,}9)$: de 3 à 5 (coût 1), de 5 à 9 (coût 2) $\implies$ total : 3

$(0{,}9)$: de 0 à 1 (coût 1), de 1 à 5 (coût 2), de 5 à 9 (coût 2) $\implies$ total : 5

$(0{,}7)$: de 0 à 1 (coût 1), de 1 à 5 (coût 2), de 5 à 7 (coût 1) $\implies$ total : 4

$(0{,}4)$: de 0 à 1 (coût 1), de 1 à 4 (coût 2) $\implies$ total : 3

$(3{,}5)$: de 3 à 5 (coût 1) $\implies$ total : 1

Ainsi, le vecteur que la fonction doit retourner est :

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

Grader d'exemple

Le grader d'exemple lit l'entrée selon le format suivant :

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

et affiche m lignes, correspondant au résultat de l'appel à `solve`.