Kolejka górska (Roller Coaster Railroad)

Anna pracuje w parku rozrywki i zajmuje się budową nowej kolejki górskiej. Zaprojektowała już n specjalnych sekcji (dla wygody ponumerowanych od 0 do $n-1$), które zmieniają prędkość wagoników kolejki: są to górki, dolinki i wiele innych. Teraz musi zaproponować ostateczny kształt kolejki, używając wszystkich zaprojektowanych sekcji. W tym zadaniu dla uproszczenia zakładamy, że długość samej kolejki wynosi 0.

Dla każdego i pomiędzy 0 a $n-1$ włącznie, specjalna sekcja i ma następujące właściwości:

- kiedy kolejka wjeżdża do danej sekcji, nie może przekraczać ustalonego limitu prędkości: prędkość wagoników musi wynosić **co najwyżej** s_i km/h (kilometrów na godzinę),
- kiedy kolejka opuszcza sekcję, jej prędkość wynosi **dokładnie** t_i km/h, niezależnie od prędkości, z jaką kolejka wjechała do tej sekcji.

Ostateczny projekt kolejki powinien zawierać każdą z n już zaprojektowanych sekcji. Każdej sekcji należy użyć dokładnie raz. Ponadto pomiędzy każdymi dwiema sąsiadującymi sekcjami można wybudować tory. Anna wybiera kolejność n sekcji, a następnie decyduje ona o długości poszczególnych torów. Długość torów jest mierzona w metrach i może być równa dowolnej nieujemnej liczbie całkowitej (w szczególności może być to 0).

Każdy metr torów pomiędzy specjalnymi sekcjami spowalnia kolejkę o 1 km/h. Na początku trasy kolejka wjeżdża do pierwszej specjalnej sekcji z prędkością 1 km/h.

Ostateczny projekt musi spełniać następujące warunki:

- w chwili wjeżdżania do specjalnej sekcji kolejka nie może przekraczać limitu prędkości;
- prędkość kolejki musi być dodatnia w każdym momencie trasy.

We wszystkich podzadaniach oprócz trzeciego Twoim zadaniem jest znaleźć kolejność n specjalnych sekcji i dobrać długości torów pomiędzy nimi, tak aby całkowita długość torów była jak najmniejsza. W trzecim podzadaniu musisz jedynie sprawdzić, czy istnieje prawidłowy projekt kolejki, w którym wszystkie tory mają długość 0.

Szczegóły implementacji

Powinieneś zaimplementować następującą funkcję (metodę):

- `int64 plan_roller_coaster(int[] s, int[] t)`

- s: tablica długości n opisująca maksymalne prędkości wejścia do sekcji.
- t: tablica długości n opisująca prędkości wyjścia z sekcji.
- We wszystkich podzadaniach poza trzecim funkcja powinna zwracać minimalną sumaryczną długość wszystkich torów pomiędzy specjalnymi sekcjami. Natomiast w trzecim podzadaniu funkcja powinna zwracać 0, jeżeli istnieje poprawny projekt kolejki, w którym wszystkie tory pomiędzy sekcjami mają długość zero, natomiast dowolną dodatnią liczbę całkowitą, jeżeli taki projekt nie istnieje.

W języku C sygnatura funkcji jest minimalnie inna:

- `int64 plan_roller_coaster(int n, int[] s, int[] t)`
  - n: rozmiar tablic s oraz t (tj. liczba specjalnych sekcji),
  - pozostałe parametry są takie same jak powyżej.

Przykład

`int64 plan_roller_coaster([1, 4, 5, 6], [7, 3, 8, 6])`

W tym przykładzie mamy cztery specjalne sekcje. Najlepszym możliwym rozwiązaniem jest wybudowanie ich w kolejności $0, 3, 1, 2$ i połączenie ich torami o długościach, odpowiednio, $1, 2, 0$. Kolejka wtedy porusza się następująco:

- Początkowa prędkość kolejki wynosi 1 km/h.
- Kolejka rozpoczyna trasę, wjeżdżając do specjalnej sekcji nr 0.
- Kolejka opuszcza sekcję 0 z prędkością 7 km/h.
- Następnie kolejka wjeżdża na tory o długości 1 m. Po ich przejechaniu ma prędkość 6 km/h.
- Kolejka wjeżdża do specjalnej sekcji nr 3 z prędkością 6 km/h i opuszcza ją z tą samą prędkością.
- Po opuszczeniu sekcji 3 kolejka jedzie przez 2 m torów. Prędkość maleje do 4 km/h.
- Kolejka wjeżdża do specjalnej sekcji nr 1 z prędkością 4 km/h i opuszcza ją z prędkością 3 km/h.
- Natychmiast po opuszczeniu sekcji specjalnej nr 1 kolejka wjeżdża do sekcji nr 2.
- Kolejka wyjeżdża z sekcji nr 2. Ostateczna prędkość kolejki wynosi 8 km/h.

Funkcja powinna zwrócić sumaryczną długość torów pomiędzy specjalnymi sekcjami: $1 + 2 + 0 = 3$.

Podzadania

We wszystkich podzadaniach $1 \leq s_i \leq 10^9$ oraz $1 \leq t_i \leq 10^9$.

1. (11 punktów): $2 \leq n \leq 8$,
2. (23 punkty): $2 \leq n \leq 16$,

3. (30 punktów): $2 \leq n \leq 200\,000$. W tym podzadaniu Twój program musi jedynie sprawdzić, czy wynikiem jest zero, czy też nie. Jeżeli wynikiem nie jest zero, każda dodatnia liczba całkowita jest uznawana za poprawną.
4. (36 punktów): $2 \leq n \leq 200\,000$.

Przykładowy program sprawdzający

Przykładowy program sprawdzający wczytuje dane w następującym formacie:

- wiersz 1: liczba całkowita n,
- wiersz $2+i$, dla i pomiędzy 0 i $n-1$: liczby całkowite s_i i t_i.