Überholen

Es gibt eine einspurige Einbahnstraße vom Flughafen Budapest zum Hotel Forrás. Die Straße ist L Kilometer lang.

Während der IOI 2023 fahren $N + 1$ Transferbusse auf dieser Straße. Die Busse sind von 0 bis N nummeriert. Bus i ($0 \leq i < N$) verlässt den Flughafen $T[i]$ Sekunden nach Beginn der IOI und kann als Höchstgeschwindigkeit 1 Kilometer in $W[i]$ Sekunden fahren. Bus N ist ein Reservebus, der als Höchstgeschwindigkeit 1 Kilometer in X Sekunden fahren kann. Die Zeit Y, zu der er den Flughafen verlässt, wurde noch nicht festgelegt.

Überholen ist auf der Straße nicht erlaubt. Aber die Busse dürfen einander bei **Sortierstationen** überholen. Es gibt M ($M > 1$) Sortierstationen, nummeriert von 0 bis $M - 1$, an verschiedenen Stellen der Straße. Sortierstation j ($0 \leq j < M$) ist $S[j]$ Straßenkilometer vom Flughafen entfernt. Die Sortierstationen sind nach aufsteigender Entfernung vom Flughafen sortiert: $S[j] < S[j + 1]$ für jedes $0 \leq j \leq M - 2$. Die erste Sortierstation ist beim Flughafen und die letzte ist beim Hotel: $S[0] = 0$ und $S[M - 1] = L$.

Jeder Bus fährt mit Höchstgeschwindigkeit, außer er holt einen langsameren Bus ein, der vor ihm auf der Straße fährt. In diesem Fall müssen beide Busse mit der Geschwindigkeit des langsameren Busses fahren, bis sie die nächste Sortierstation erreichen. Dort überholen dann die schnelleren Busse die langsameren.

Formal ist die Zeit $t_{i,j}$ (in Sekunden), zu der Bus i bei der Sortierstation j **ankommt**, wie folgt definiert (für alle $0 \leq i \leq N$ und $0 \leq j \leq M$): Sei $t_{i,0} = T[i]$ für jedes $0 \leq i < N$, und sei $t_{N,0} = Y$. Für jedes j so dass $0 < j < M$:

- Die **erwartete Ankunftszeit** (in Sekunden) von Bus i bei Sortierstation j, bezeichnet mit $e_{i,j}$, sei definiert als die Zeit, zu der Bus i bei der Sortierstation j ankommen würde, wenn er seit der Ankunftszeit bei Sortierstation $j - 1$ mit Höchstgeschwindigkeit fahren würde. Es gilt also:
  - $e_{i,j} = t_{i,j-1} + W[i] \cdot (S[j] - S[j - 1])$ für jedes $0 \leq i < N$, und
  - $e_{N,j} = t_{N,j-1} + X \cdot (S[j] - S[j - 1])$.
- Die Ankunftszeit von Bus i bei Sortierstation j ist das *Maximum* der erwarteten Ankunftszeit von Bus i und von jedem anderen Bus, der bei Station $j - 1$ früher als Bus i ankommt. Formal ausgedrückt, sei $t_{i,j}$ das Maximum von $e_{i,j}$ und jedem $e_{k,j}$, für das $0 \leq k \leq N$ und $t_{k,j-1} < t_{i,j-1}$.

Die IOI-Veranstalter möchten nun die Fahrt des Reservebusses planen (Bus N). Deine Aufgabe ist es, Q Fragen der Veranstalter zu beantworten, und zwar dieser Art: Wann würde der Reservebus beim Hotel ankommen, wenn er den Flughafen zur Zeit Y (in Sekunden) verlässt?

Implementierungsdetails

Implementiere die folgenden Funktionen:

```
void init(int L, int N, int64[] T, int[] W, int X, int M, int[] S)
```

- L: die Länge der Straße.
- N: die Anzahl der *regulären* Busse (ohne Reservebus).
- T: ein Array der Länge N, das die Zeiten beschreibt, zu denen die regulären Busse den Flughafen verlassen.
- W: ein Array der Länge N, das die (Höchst-)Geschwindigkeiten der regulären Busse beschreibt.
- X: die (Höchst-)Geschwindigkeit des Reservebusses.
- M: die Anzahl der Sortierstationen.
- S: ein Array der Länge M, das die Distanzen der Sortierstationen vom Flughafen beschreibt.
- Diese Funktion wird genau einmal für jeden Testfall aufgerufen, vor allen Aufrufen der folgenden Funktion `arrival_time`.

```
int64 arrival_time(int64 Y)
```

- Y: die Zeit, zu welcher der Reservebus (Bus N) den Flughafen verlassen soll.
- Diese Funktion soll die Zeit zurückgeben, zu welcher der Reservebus das Hotel erreicht.
- Diese Funktion wird genau Q Mal aufgerufen.

Beispiel

Wir betrachten einige Funktionsaufrufe, beginnend mit:

```
init(6, 4, [20, 10, 40, 0], [5, 20, 20, 30], 10, 4, [0, 1, 3, 6])
```

Die folgende Tabelle zeigt die erwarteten und echten Ankunftszeiten der regulären Busse (ohne den Reservebus 4) an den Sortierstationen:

i		$t_{i,0}$		$e_{i,1}$	$t_{i,1}$		$e_{i,2}$	$t_{i,2}$		$e_{i,3}$	$t_{i,3}$
0		20		25	30		40	40		55	55
1		10		30	30		70	70		130	130
2		40		60	60		100	100		160	180
3		0		30	30		90	90		180	180

Die Ankunftszeiten an Station 0 sind die geplanten Abfahrtszeiten der Busse vom Flughafen. Es gilt also $t_{i,0} = T[i]$ für $0 \leq i \leq 3$.

Die erwarteten und echten Ankunftszeiten an Sortierstation 1 werden wie folgt berechnet:

- Die erwarteten Ankunftszeiten an Station 1:
  - Bus 0: $e_{0,1} = t_{0,0} + W[0] \cdot (S[1] - S[0]) = 20 + 5 \cdot 1 = 25$.
  - Bus 1: $e_{1,1} = t_{1,0} + W[1] \cdot (S[1] - S[0]) = 10 + 20 \cdot 1 = 30$.
  - Bus 2: $e_{2,1} = t_{2,0} + W[2] \cdot (S[1] - S[0]) = 40 + 20 \cdot 1 = 60$.
  - Bus 3: $e_{3,1} = t_{3,0} + W[3] \cdot (S[1] - S[0]) = 0 + 30 \cdot 1 = 30$.
- Die echten Ankunftszeiten an Station 1:
  - Die Busse 1 und 3 kommen vor Bus 0 an Station 0 an, also ist $t_{0,1} = \max([e_{0,1}, e_{1,1}, e_{3,1}]) = 30$.
  - Bus 3 kommt vor Bus 1 an Station 0 an, also ist $t_{1,1} = \max([e_{1,1}, e_{3,1}]) = 30$.
  - Die Busse 0, 1 und 3 kommen vor Bus 2 an Station 0 an, also ist $t_{2,1} = \max([e_{0,1}, e_{1,1}, e_{2,1}, e_{3,1}]) = 60$.
  - Es kommt kein Bus vor Bus 3 an Station 0 an, also ist $t_{3,1} = \max([e_{3,1}]) = 30$.

```
arrival_time(0)
```

Bus 4 braucht 10 Sekunden für 1 Kilometer. Er soll nun nach 0 Sekunden am Flughafen abfahren. Für diesen Fall zeigt die folgende Tabelle die Ankunftszeiten aller Busse. Der einzige Unterschied bei den erwarteten und echten Ankunftszeiten der regulären Busse ist unterstrichen.

i		$t_{i,0}$		$e_{i,1}$	$t_{i,1}$		$e_{i,2}$	$t_{i,2}$		$e_{i,3}$	$t_{i,3}$
0		20		25	30		40	40		55	$\underline{60}$
1		10		30	30		70	70		130	130
2		40		60	60		100	100		160	180
3		0		30	30		90	90		180	180
4		0		10	10		30	30		60	60

Der Reservebus 4 kommt also nach 60 Sekunden am Hotel an. Folglich soll die Funktion 60 zurückgeben.

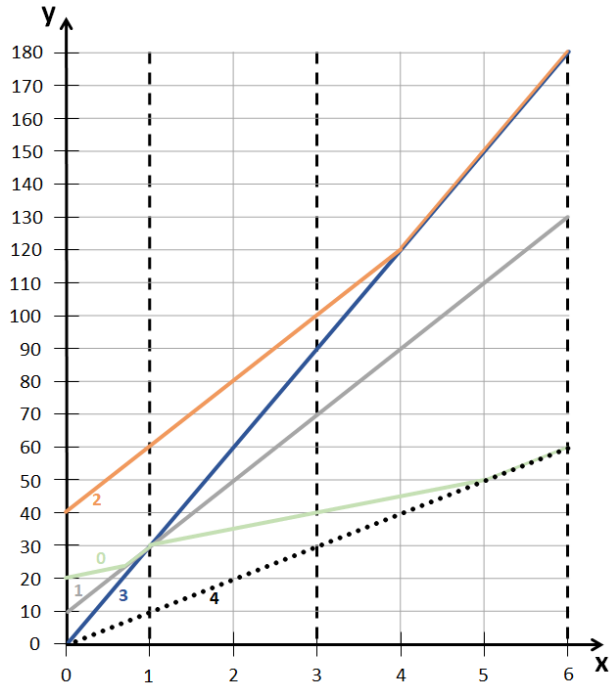
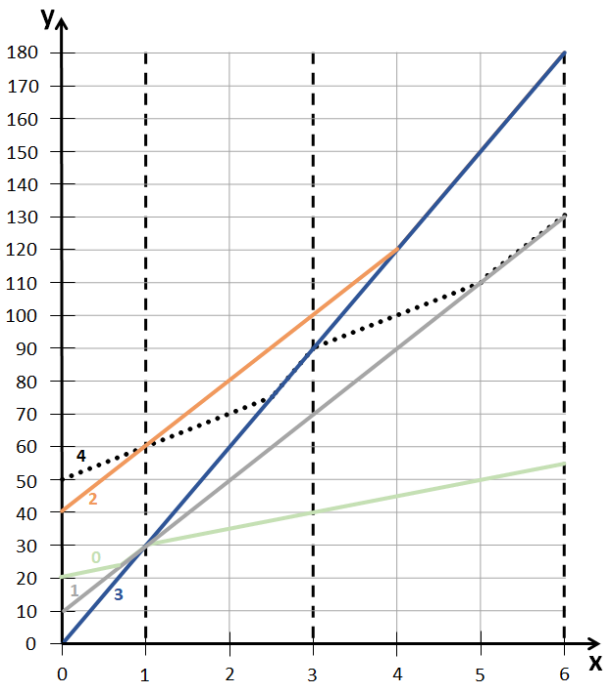
```
arrival_time(50)
```

Jetzt soll Bus 4 nach 50 Sekunden abfahren. In diesem Fall gibt es keine Veränderungen der Ankunftszeiten der regulären Busse im Vergleich zur ersten Tabelle. Die Ankunftszeiten werden in folgender Tabelle dargestellt.

i	$t_{i,0}$	$e_{i,1}$	$t_{i,1}$	$e_{i,2}$	$t_{i,2}$	$e_{i,3}$	$t_{i,3}$
0	20	25	30	40	40	55	55
1	10	30	30	70	70	130	130
2	40	60	60	100	100	160	180
3	0	30	30	90	90	180	180
4	50	60	60	80	90	120	130

Bus 4 überholt den langsameren Bus 2 bei Sortierstation 1, da sie zur gleichen Zeit dort ankommen. Dann wird Bus 4 zwischen den Stationen 1 und 2 von Bus 3 gebremst, weshalb Bus 4 erst nach 90 Sekunden an Station 2 ankommt und nicht nach 80 Sekunden. Nachdem Bus 4 Station 2 verlässt, wird er von Bus 1 gebremst, bis beide im Hotel ankommen. Bus 4 kommt also nach 130 Sekunden am Hotel an. Folglich soll die Funktion 130 zurückgeben.

Die Zeiten, die die einzelnen Busse für die möglichen Entfernungen vom Flughafen brauchen, lassen sich als Diagramm darstellen. Die X-Achse gibt die Entfernung (in Kilometern) vom Flughafen an, und die Y-Achse gibt die Zeit (in Sekunden) an. Vertikale gestrichelte Linien zeigen die Positionen der Sortierstationen. Die durchgezogenen Linien (mit den Nummern der Busse) stellen die vier regulären Busse dar. Die gepunktete schwarze Linie stellt den Reservebus dar.

arrival_time(0)	arrival_time(50)
	

Beschränkungen

- $1 \leq L \leq 10^9$
- $1 \leq N \leq 1\,000$
- $0 \leq T[i] \leq 10^{18}$ $(0 \leq i < N)$
- $1 \leq W[i] \leq 10^9$ $(0 \leq i < N)$
- $1 \leq X \leq 10^9$
- $2 \leq M \leq 1\,000$
- $0 = S[0] < S[1] < \cdots < S[M-1] = L$
- $1 \leq Q \leq 10^6$
- $0 \leq Y \leq 10^{18}$

Teilaufgaben

1. (9 Punkte) $N = 1, Q \leq 1\,000$
2. (10 Punkte) $M = 2, Q \leq 1\,000$
3. (20 Punkte) $N, M, Q \leq 100$
4. (26 Punkte) $Q \leq 5\,000$
5. (35 Punkte) Keine weiteren Beschränkungen.

Beispielgrader

Der Beispielgrader liest die Eingabe im folgenden Format:

- Zeile 1: $L\ N\ X\ M\ Q$
- Zeile 2: $T[0]\ T[1]\ \ldots\ T[N-1]$
- Zeile 3: $W[0]\ W[1]\ \ldots\ W[N-1]$
- Zeile 4: $S[0]\ S[1]\ \ldots\ S[M-1]$
- Zeile $5+k$ ($0 \leq k < Q$): Y der k-ten Frage

Der Beispielgrader gibt deine Antworten im folgenden Format aus:

- Zeile $1+k$ ($0 \leq k < Q$): der Rückgabewert von `arrival_time` für die k-te Frage