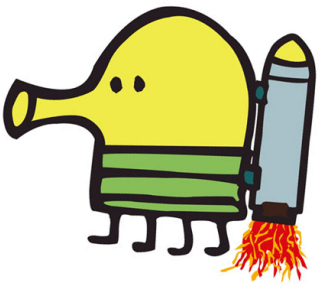
Doodle Jump

Description et tâche



Dans le jeu *Doodle Jump,* un personnage initialement sur le sol (à 0 mètres) doit bondir de plateforme en plateforme pour atteindre la ligne d'arrivée située à m mètres du sol. Sur certaines de ces plateformes se trouve un jetpack avec une quantité de carburant b_i. Quand le personnage arrive à la hauteur a_i d'un jetpack, il peut l'utiliser ou l'ignorer. Si le personnage choisit de l'utiliser, le jetpack propulse le personnage sans effort vers le haut d'un mètre par unité de carburant. Seulement lorsque le carburant est vide, et non avant, le personnage perd son jetpack et continue à sauter. Le personnage ne peut jamais redescendre, seulement monter et il ne peut utiliser qu'un seul jetpack à la fois. Il est garanti qu'un jetpack ne dépasse jamais la ligne d'arrivée. Le personnage étant paresseux, il aimerait savoir la distance minimale qu'il est obligé de sauter lui-même en optimisant le choix des jetpacks (il veut donc maximiser la distance parcourue en jetpack). Votre tâche consiste à trouver cette distance minimale.

Entrée

La première ligne contient 2 entiers n et m, le premier désignant le nombre de jetpacks et le second la hauteur en mètres de la ligne d'arrivée.

Les n lignes suivantes contiennent 2 entiers a_i et b_i, a_i désignant la hauteur en mètres à laquelle se trouve un jetpack et b_i la quantité de carburant qu'il possède.

Sortie

Le programme affiche un seul entier, la distance minimale, en mètres que le personnage doit parcourir en sautant.

Restrictions

$0 \leq n \leq 10^5$
$0 \leq m \leq 10^9$

pour $1 \leq i \leq n$:
$0 \leq a_i, b_i \leq m$
$a_i + b_i \leq m$

Exemples d'exécution

entrée	sortie
3 6 2 3 1 2 2 3	3
4 10 9 1 1 3 0 4 4 1	4

Dans le premier exemple, le personnage choisit de ne pas utiliser le premier jetpack qu'il croise. Il prend le second qui l'emmène de la hauteur 2 à la hauteur 5 (3 mètres plus haut). Il parcourt le dernier mètre en sautant. Il a donc parcouru 2+1 = 3 mètres.

Dans le second exemple, il utilise tous les jetpacks sauf celui à la hauteur 1.

Solution

Minimiser la distance parcourue en sautant revient à maximiser celle parcourue en jetpack. Tout d'abord on trie en temps `O(NlogN)` les jetpacks tel que si `a[i]<a[j]` alors `i<j`. On utilise ensuite la programmation dynamique. Soit `dp[i]` la distance maximale que le personnage peut voler à partir de la position `a[i]`. Il peut alors soit ignorer le jetpack et sauter jusqu'à la position `a[i+1]`, soit voler avec, puis sauter jusqu'au prochain jetpack pour prendre la prochaine décision ou bien jusqu'à la ligne d'arrivée s'il ne rencontre plus de jetpack. Pour le dernier jetpack, la meilleure stratégie consiste à toujours le prendre. On peut en dériver les formules suivantes:

```
dp[n]=b[n];
dp[i]=max(dp[i+1],dp[k]+b[i]);
```

pour `1<=i<=n-1` avec k l'indice du prochain jetpack que le personnage rencontre (celui-ci peut être trouvé en temps `O(logN)` avec un *binary search* par exemple). On choisira par convention `dp[n+1]=0` pour traiter le cas où le personnage ne rencontre plus de jetpack. L'algorithme entier s'exécute en temps `O(NlogN)`. La réponse finale est donnée par

```
m-dp[1];
```