Scroll Translation

Objective
Give practice with a greedy algorithm.
Give practice with a sweep.
Give additional practice with sorting.

Story
Many artifacts have since been recovered by your “colleagues”. You want to begin determining what important finds your team has recovered. There are several old, deteriorating scrolls that might be critical to finding the biggest score of an ancient civilization in history. You want to determine the exact content of these documents, but you do not have the ability to interpret these ancient writings alone. You, luckily, have a translator for the ancient documents. However, your translator speaks and writes in a modern vernacular associated with a local populace that you never bothered to learn. Luckily, you have a second translator that can take the local tongue and turn it into a format with which you are familiar.

Each document has some time requirement for translation by both translators. The times required by both translators can vary from document to document and translators can take different times on the same document. To ensure that no errors are made in the process the two translators will not work on the same document at the same time. Additionally, if not already obvious, your second translator must wait until the first translator is done before they can get to work. You want to determine the earliest time at which all the documents can be read (by you). You can change the order in which the documents are translated to ensure the total time taken will be as small as possible.

Problem
Given a list of required times for translation by two different translators determine the soonest time in which all the documents can be completely translated.

Input
Input will begin with a line containing 1 integer, n ($1 \leq n \leq 500{,}000$), representing the number of scrolls to translate. The following n lines will each contain 2 integers, the i-th of which are a_i and b_i ($1 \leq a_i, b_i \leq 1{,}000{,}000{,}000$), which represent the time required by the first and second translator respectively to translate the i-th document.

Output
Output should contain 1 line containing a single integer representing the minimum total required time to translate all the documents completely using both translators.

Sample Input	**Sample Output**

`4`
`5 7`
`7 8`
`8 4`
`4 5`
	`28`

`3`
`10 1`
`2 4`
`5 5`
	`18`

Explanation

Case 1

You can order the scrolls as (4 5), (5 7), (7 8), and (8 4) for an optimal order

If we order the scrolls in the above order we can see the following behavior.

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Trans. 1	Scroll (**4**,5)				Scroll (**5**,7)					Scroll (**7**,8)						
Trans. 2	UNUSED				Scroll (4,**5**)					Scroll (5,**7**)						

Time	17	18	19	20	21	22	23	24	25	26	27	28
Trans. 1	Scroll (**8**,4)								UNUSED			
Trans. 2	Scroll (7,**8**)								Scroll (8,**4**)			

The first 4 time units the first scroll (4,5) is translated by the first translator and the second translator does nothing. The next 5 time units the first scroll is translated by the SECOND translator, and the first translator translates the next scroll (5,7). In the next 7 time units the first translator translates the third scroll (7,8), while the second translator translates the second. In the next 8 time units the first translator translates the last scroll (8,4), while the second one translates the third. In the last 4 time units the first translator does nothing, while the second translates the last scroll.

The total time is 4 + 5 + 7 + 8 + 4 = 28.

Case 2
An optimal order is (2 4), (5 5), and (10 1).

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Trans. 1	(**2**,4)		(**5**,5)					(**10**,1)										!U
Trans. 2	!USED		(2,**4**)				!U	(5,**5**)					!USED					(10,**1**)

The last box is only wider because of the order pair being so large -_-

Note that the second translator cannot work for 1 time unit after he finishes the first scroll, because the second scroll was not finished yet.

Hints

Sorting: Since this is the greedy assignment, it is strongly encouraged that you only try one possible arrangement by sorting the scrolls with respect to each other by some comparison method.

One Object Per Scroll: In some greedies it is ideal to make two events or sortable objects per given item. In this problem you only need one.

The Sweep: Once you have the proper ordering you need to step through the ordered objects and compute the total time to translate the documents. Consider storing how much time the second translator still needs to work before he is done. Every time the first translator finishes a document, decrement the time for the second translator by the time required by the first prior to adding the time extra the second translator needs to work for this partially finished document. Make sure that after decrementing the second translator's time is non-negative.

The Actual Greedy: With respect to final sweep we can observe that we want the second translator to have a lot of extra time so that way he stays busy. The first greedy observation is that documents with more second translator time than first translator time needs to be handled before documents that have more translator time than second translator time.

Grouping: There will be two to three groups (or types) of scrolls.

The first group I recommend using is when the first translation time is **less than** the second translation time. As mentioned in the hint above this type of scroll will be better to have earlier than the scrolls for the other group.

The second (and arguably optional group) is the group where the first and second translation time are **equal**. These are better than having a longer first translation time, but worse than having a shorter first translation time.

The third group is composed of the scrolls where the first translation time is **greater than** the second translation time. These should strictly follow the scrolls from the other two groups to ensure that the second translator has built up extra time.

When Breaking Ties: When building up as much extra time for the second translator, it is a good idea to choose quick to translate documents for the first translator to build up time for the second translator. In other words the first group should have shorter first translation times first.

When handling the documents that require less second translation time (group 3), it is better to take the documents that require the longer secondary translation time first.

Grading Criteria

- Read/Write from/to standard input/output
    - 10 points
- Good comments, whitespace, and variable names
    - 15 points
- Implement code to determine the total translation time based on an order (sweep)
    - 10 points
- Implement code to sort a list of documents based on the translation times (comparable interface)
    - 10 points
- Implement a translation time storage method (a class)
    - 5 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using javac.

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a greedy sweep. ***Without this programs will earn at most 50 points!***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 5 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.