RISK
IDENT

Scala Coding Challenge

The following is a coding challenge which should be completed in Scala.

The task shouldn't take more than 4 hours, and leaves you a certain amount of freedom in how you go about solving it. This is deliberate. Your code will serve as the basis for a technical discussion during the interview, where we explore your solution and dig a bit deeper into what assumptions you have, and why you made the choices you made. We are writing functional code, so please keep in mind that somebody will read it. Try to write your solution with production-ready quality based on your previous experience.

Please provide your solution as sbt project in tar.gz and do not push it public in Github.

Good luck!

Minimum Average Waiting Time

Problem Statement

Tieu owns a pizza restaurant and he manages it in his own way. While in a normal restaurant, a customer is served by following the first-come, first-served rule, Tieu simply minimizes the average waiting time of his customers. So he gets to decide who is served first, regardless of how sooner or later a person comes.

Different kinds of pizzas take different amounts of time to cook. Also, once he starts cooking a pizza, he cannot cook another pizza until the first pizza is completely cooked. Let's say we have three customers who come at time t=0, t=1, & t=2 respectively, and the time needed to cook their pizzas is 3, 9, & 6 respectively. If Tieu applies first-come, first-served rule, then the waiting time of three customers is 3, 11, & 16 respectively. The average waiting time in this case is (3 + 11 + 16) / 3 = 10. This is not an optimized solution. After serving the first customer at time t=3, Tieu can choose to serve the third customer. In that case, the waiting time will be 3, 7, & 17 respectively. Hence the average waiting time is (3 + 7 + 17) / 3 = 9.

Help Tieu achieve the minimum average waiting time. For the sake of simplicity, just find the integer part of the minimum average waiting time.

Input Format

- The first line contains an integer N, which is the number of customers.
- In the next N lines, the i^{th} line contains two space separated numbers T_i and L_i. T_i is the time when i^{th} customer order a pizza, and L_i is the time required to cook that pizza.

Output Format

- Display the integer part of the minimum average waiting time.
- Make your solution callable via following function:

```
object OrderProcessor {

  /**
   * @param in: An InputStream, which contains the following
input:
   * A line containing a single number: The number of guests
G,
   * Followed by G lines containing two numbers Oi and Di
separated by space.
   * There may be a trailing newline.
   * Oi ist the ordering time for Gi, Di is the time it takes
to bake Gi's pizza.
   * 0 <= G  <= 100000
   * 0 <= Oi <= 1000000000
   * 1 <= Di <= 1000000000
   *
   * @param out: A PrintStream, which process writes the
following output to:
   * A single line containing the integer part of the average
waiting time if the input is valid.
   * A single line starting with the words "Syntax error" and
an optional description otherwise.
   * There may be a trailing newline.
   */
  def process(in: InputStream, out: PrintStream): Unit = {

  }
}
```

Constraints

- $1 \leq N \leq 10^5$
- $0 \leq T_i \leq 10^9$
- $1 \leq L_i \leq 10^9$

Note

- The waiting time is calculated as the difference between the time a customer orders a pizza (the time at which they enter the shop) and the time it is served.
- The cook does not know about future orders.

Sample Input #00

3
0 3
1 9
2 6

Sample Output #00

9

Sample Input #01

```
3
0 3
1 9
2 5
```

Sample Output #01

```
8
```

Explanation #01

Let's call the person ordering at time = 0 as *A*, time = 1 as *B* and time = 2 as *C*. By delivering pizza for *A*, *C* and *B* we get the minimum average wait time to be

(3 + 6 + 16)/3 = 25/3 = 8.33

the integer part is **8** and hence the answer.