3 Advanced Problem: Network packet processing simulation

We strongly recommend you start solving advanced problems only when you are done with the basic problems (for some advanced problems, algorithms are not covered in the video lectures and require additional ideas to be solved; for some other advanced problems, algorithms are covered in the lectures, but implementing them is a more challenging task than for other problems).

Problem Introduction

In this problem you will implement a program to simulate the processing of network packets.

Problem Description

Task. You are given a series of incoming network packets, and your task is to simulate their processing. Packets arrive in some order. For each packet number i, you know the time when it arrived A_i and the time it takes the processor to process it P_i (both in milliseconds). There is only one processor, and it processes the incoming packets in the order of their arrival. If the processor started to process some packet, it doesn't interrupt or stop until it finishes the processing of this packet, and the processing of packet i takes exactly P_i milliseconds.

The computer processing the packets has a network buffer of fixed size S. When packets arrive, they are stored in the buffer before being processed. However, if the buffer is full when a packet arrives (there are S packets which have arrived before this packet, and the computer hasn't finished processing any of them), it is dropped and won't be processed at all. If several packets arrive at the same time, they are first all stored in the buffer (some of them may be dropped because of that — those which are described later in the input). The computer processes the packets in the order of their arrival, and it starts processing the next available packet from the buffer as soon as it finishes processing the previous one. If at some point the computer is not busy, and there are no packets in the buffer, the computer just waits for the next packet to arrive. Note that a packet leaves the buffer and frees the space in the buffer as soon as the computer finishes processing it.

Input Format. The first line of the input contains the size S of the buffer and the number n of incoming network packets. Each of the next n lines contains two numbers. i-th line contains the time of arrival A_i and the processing time P_i (both in milliseconds) of the i-th packet. It is guaranteed that the sequence of arrival times is non-decreasing (however, it can contain the exact same times of arrival in milliseconds — in this case the packet which is earlier in the input is considered to have arrived earlier).

Constraints. All the numbers in the input are integers. $1 \leq S \leq 10^5$; $1 \leq n \leq 10^5$; $0 \leq A_i \leq 10^6$; $0 \leq P_i \leq 10^3$; $A_i \leq A_{i+1}$ for $1 \leq i \leq n-1$.

Output Format. For each packet output either the moment of time (in milliseconds) when the processor began processing it or -1 if the packet was dropped (output the answers for the packets in the same order as the packets are given in the input).

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 6 sec, `Python`: 8 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 6 sec, `Ruby`: 6 sec, `Scala`: 6 sec.

Memory Limit. 512MB.

Sample 1.

Input:

```
1 0
```

Output:

```

```

Explanation:
If there are no packets, you shouldn't output anything.

Sample 2.

Input:

```
1 1
0 0
```

Output:

```
0
```

Explanation:
The only packet arrived at time 0, and computer started processing it immediately.

Sample 3.

Input:

```
1 2
0 1
0 1
```

Output:

```
0
-1
```

Explanation:
The first packet arrived at time 0, the second packet also arrived at time 0, but was dropped, because the network buffer has size 1 and it was full with the first packet already. The first packet started processing at time 0, and the second packet was not processed at all.

Sample 4.

Input:

```
1 2
0 1
1 1
```

Output:

```
0
1
```

Explanation:
The first packet arrived at time 0, the computer started processing it immediately and finished at time 1. The second packet arrived at time 1, and the computer started processing it immediately.

Starter Files

The starter solutions for C++, Java and Python3 in this problem read the input, pass the requests for processing of packets one-by-one and output the results. They declare a class that implements network buffer simulator. The class is partially implemented, and your task is to implement the rest of it. If you use other languages, you need to implement the solution from scratch.

What to Do

To solve this problem, you can use a list or a queue (in this case the queue should allow accessing its last element, and such queue is usually called a deque). You can use the corresponding built-in data structure in your language of choice.

One possible solution is to store in the list or queue `finish_time` the times when the computer will finish processing the packets which are currently stored in the network buffer, in increasing order. When a new packet arrives, you will first need to pop from the front of `finish_time` all the packets which are already processed by the time new packet arrives. Then you try to add the finish time for the new packet in `finish_time`. If the buffer is full (there are already S finish times in `finish_time`), the packet is dropped. Otherwise, its processing finish time is added to `finish_time`.

If `finish_time` is empty when a new packet arrives, computer will start processing the new packet immediately as soon as it arrives. Otherwise, computer will start processing the new packet as soon as it finishes to process the last of the packets currently in `finish_time` (here is when you need to access the last element of `finish_time` to determine when the computer will start to process the new packet). You will also need to compute the processing finish time by adding P_i to the processing start time and push it to the back of `finish_time`.

You need to remember to output the processing start time for each packet instead of the processing finish time which you store in `finish_time`.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.