share, $p(j) - p(i)$? (If there is no way to make money during the n days, we should conclude this instead.)

In the solved exercise, we showed how to find the optimal pair of days i and j in time $O(n \log n)$. But, in fact, it's possible to do better than this. Show how to find the optimal numbers i and j in time $O(n)$.

8. The residents of the underground city of Zion defend themselves through a combination of kung fu, heavy artillery, and efficient algorithms. Recently they have become interested in automated methods that can help fend off attacks by swarms of robots.

Here's what one of these robot attacks looks like.

- A swarm of robots arrives over the course of n seconds; in the i^{th} second, x_i robots arrive. Based on remote sensing data, you know this sequence $x_1, x_2, \ldots, x_n$ in advance.
- You have at your disposal an *electromagnetic pulse* (EMP), which can destroy some of the robots as they arrive; the EMP's power depends on how long it's been allowed to charge up. To make this precise, there is a function $f(\cdot)$ so that if j seconds have passed since the EMP was last used, then it is capable of destroying up to $f(j)$ robots.
- So specifically, if it is used in the k^{th} second, and it has been j seconds since it was previously used, then it will destroy $\min(x_k, f(j))$ robots. (After this use, it will be completely drained.)
- We will also assume that the EMP starts off completely drained, so if it is used for the first time in the j^{th} second, then it is capable of destroying up to $f(j)$ robots.

The problem. Given the data on robot arrivals $x_1, x_2, \ldots, x_n$, and given the recharging function $f(\cdot)$, choose the points in time at which you're going to activate the EMP so as to destroy as many robots as possible.

Example. Suppose $n = 4$, and the values of x_i and $f(i)$ are given by the following table.

i	**1**	**2**	**3**	**4**
x_i	1	10	10	1
$f(i)$	1	2	4	8

The best solution would be to activate the EMP in the 3^{rd} and the 4^{th} seconds. In the 3^{rd} second, the EMP has gotten to charge for 3 seconds, and so it destroys $\min(10, 4) = 4$ robots; In the 4^{th} second, the EMP has only gotten to charge for 1 second since its last use, and it destroys $\min(1, 1) = 1$ robot. This is a total of 5.

(a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

```
Schedule-EMP(x_1, ..., x_n)
  Let j be the smallest number for which f(j) ≥ x_n
    (If no such j exists then set j = n)
  Activate the EMP in the n^th second
  If n − j ≥ 1 then
    Continue recursively on the input x_1, ..., x_{n−j}
      (i.e., invoke Schedule-EMP(x_1, ..., x_{n−j}))
```

In your example, say what the correct answer is and also what the algorithm above finds.

(b) Give an efficient algorithm that takes the data on robot arrivals $x_1, x_2, \ldots, x_n$, and the recharging function $f(\cdot)$, and returns the maximum number of robots that can be destroyed by a sequence of EMP activations.

9. You're helping to run a high-performance computing system capable of processing several terabytes of data per day. For each of n days, you're presented with a quantity of data; on day i, you're presented with x_i terabytes. For each terabyte you process, you receive a fixed revenue, but any unprocessed data becomes unavailable at the end of the day (i.e., you can't work on it in any future day).

You can't always process everything each day because you're constrained by the capabilities of your computing system, which can only process a fixed number of terabytes in a given day. In fact, it's running some one-of-a-kind software that, while very sophisticated, is not totally reliable, and so the amount of data you can process goes down with each day that passes since the most recent reboot of the system. On the first day after a reboot, you can process s_1 terabytes, on the second day after a reboot, you can process s_2 terabytes, and so on, up to s_n; we assume $s_1 > s_2 > s_3 > \cdots > s_n > 0$. (Of course, on day i you can only process up to x_i terabytes, regardless of how fast your system is.) To get the system back to peak performance, you can choose to reboot it; but on any day you choose to reboot the system, you can't process any data at all.

The problem. Given the amounts of available data $x_1, x_2, \ldots, x_n$ for the next n days, and given the profile of your system as expressed by $s_1, s_2, \ldots, s_n$ (and starting from a freshly rebooted system on day 1), choose