- See previous practice problem sets for instructions.

1. Our neighbourhood machine shop has a buffing machine that they use to do the finishing touches on metal strips. A schematic of this machine is shown in Figure 1. The metal strip enters from the right side, gets processed by the machine, and exits from the left.

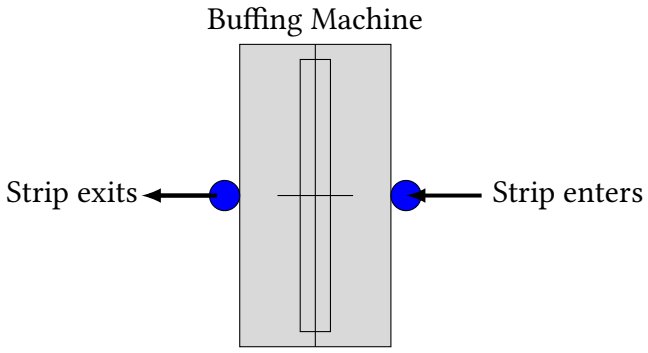

Figure 1: The buffing machine at work.

There is a maximum length M of continuous metal strip that this machine can process at one go. This length M is determined by various factors such as the length of the output tray and the heat and debris produced during the processing. Once the machine has processed length M worth of continuous metal strip the processed strip has to be cut out and removed, and the machine has to be stopped, cleaned, and cooled down before it can start processing again. Processing beyond length M at one go will result in very expensive recovery operations, so the machine shop has to avoid this at all costs.

The input strip comes with notches at random locations along its length, as shown in Figure 2. The machine has to cut the strip only at these notches, not at any other point along the strip.

Figure 2: Notches on the strip.

Once a metal strip is processed by the buffing machine, it has to be welded back together to its original length before being returned to the customer. To reduce the total cost of processing, the metal shop would like to minimize the number of times that each strip is cut by the buffing machine. Your task is to design a greedy algorithm that achieves *the minimum number of cuts*.

More formally: Let L be the length of an input strip, and let $x_1, x_2, \ldots, x_n$ be the distances—from the left end-point of the strip—at which the notches are located. We refer to each notch by its distance from the left end-point of the strip. We say that a subset N of these notches is *feasible* if (i) the leftmost notch—say x_i—in N satisfies $x_i \leq M$, (ii) every two consecutive notches x_j, x_k in N satisfy $(x_k - x_j) \leq M$, and the rightmost notch—say x_ℓ—in N satisfies $(L - x_\ell) \leq M$. That is, a set of notches is feasible if and only if the buffing machine can process the entire strip by cutting it up exactly at these notches. We assume that the set of all available notches is feasible, since otherwise there is no way that the

machine can process the entire strip without expensive stoppages. Your task is to design a greedy algorithm that finds a feasible set of notches of the *smallest size*.

I have described three greedy strategies below. In each case, either come up with an example to show that the strategy is not feasible/optimal, or prove that the strategy is optimal.

(a) **Very Greedy Strategy:** Whenever a notch is encountered, cut the strip at that notch.

(b) **Conservative Strategy:** Keep track of the length ℓ of strip that has been processed since the last cut. If no cut has been made so far, then ℓ is the length of the strip from the start of the strip to the current location. When a notch is encountered,

- If $\ell \geq \frac{M}{2}$ then cut the strip at the current notch
- If $\ell < \frac{M}{2}$ then don't cut the strip at the current notch

(c) **Optimistic Strategy:** When a notch is encountered, look ahead and find the distance ℓ of the *next* notch from the point where the last cut was made. If no cut has been made so far, then ℓ is the distance from the start of the strip to the *next* notch. If there is no next notch, then ℓ is the distance till the end of the strip.

- If $\ell > M$ then cut the strip at the current notch
- If $\ell \leq M$ then don't cut the strip at the current notch

2. To promote saving among poor households, the Government plans to issue a series of 12 bonds $B_1, B_2, \ldots, B_{12}$ on January 1, 2025. Each of these bonds has an initial issue price of ₹$10,000$/- and a duration of 12 months. The bonds accumulate compound interest on a monthly basis at *monthly* rates of $r_1 < r_2 < \cdots < r_{12}$ respectively where each $r_i > 1$. The maturity amount consisting of the principal and accumulated interest, will be paid out on January 1, 2026.

   To promote monthly savings, the Government (re)opens the bonds for subscription on the first day of each month till December 2025; that is, the bonds can be bought on Jan 1, Feb 1, …Dec 1 2025. To be fair to all investors, the bonds are offered each month *at a price that includes the accumulated interest.* For instance: bond B_6 can be bought at ₹$10,000$/- on January 1, at ₹$10,000 \times r_6^3$ on April 1, and at ₹$10,000 \times r_6^{10}$ on November 1 (among other dates), and will be redeemed at ₹$10,000 \times r_6^{12}$ on January 1, 2026.

   To prevent hoarding, each person is allowed to buy *at most one unit* of each series. So a person—if they have the money—can buy at most one unit of each of the 12 bonds, where the buying is spread out over the 12 dates Jan 1, Feb 1, …Dec 1 2025.

   Suppose you have the money to buy any bond that is available for your purchase, at the prices that they are offered each month. What is a sequence of purchases that will maximize your total return (total redemption proceeds, minus total cost of purchase)? Use an *exchange argument* to prove that this sequence in fact maximizes the total return.