Problem 5: Hit The Fly (15 points) [File Upload]

By Arya Maheshwari

*Read through the **Problem Statement** and **Grading** sections of this problem fully. We've also provided an example of the setup and why one example algorithm would fail for intuition in the **Example** section.*

Problem Statement

You wake up in the middle of the night in Cuyler Hall to a sudden buzzing noise around you. Once again, a fly has gotten into the room. Since you're terrified of getting insect bites, you want to hit the fly as soon as possible to put an end to its buzzing, but you can't see the fly in the dark room. Your goal is to design an algorithm to guarantee that you will eventually hit the fly.

Your room is a circle and and you know the fly is buzzing along the perimeter, so we will model your room as a 1-dimensional circular array of size N (indexed by locations $0, \cdots, N-1$). The fly starts at some initial index (say i) which we view as its position at time $t = 0$, when you wake up to the buzzing. This starting position is unknown to you. Luckily, you happen to be an expert on fly flight dynamics and know that the fly will be moving at some constant speed through your array-room, i.e. say it moves k indices forward in the array every time step – but you also don't know what this constant speed is. More formally, the position p_t of the fly at time t is given by

$$p_t = (p_{t-1} + k) \mod N \text{ for } t \geq 1 \text{ and } p_0 = i$$

where both i and k are unknown integers. This setup is illustrated in the diagram in Figure 1. Note that the fly will wrap around back to index 0 when it tries to move past index $N-1$ in the circular array (since we're modeling your room, which is a circle).

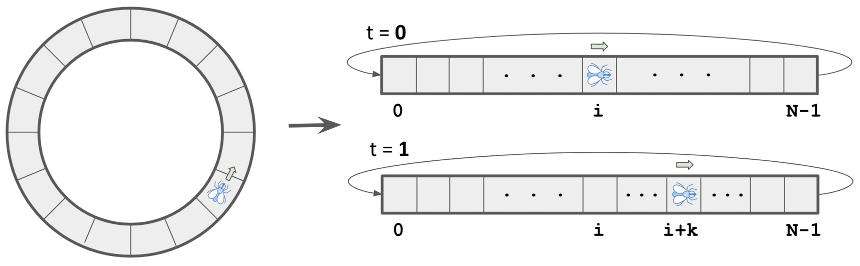

Figure 1: Diagram of fly movement. Left: top-down room view. Right: how to interpret this as a circular array.

Here's what you can do: at each time step $t = 1, 2, \cdots$, you can select an index $x_t \in \{0, \cdots, N-1\}$ in the array to HIT. Then there are two cases: if **(1)** the fly position p_t is equal to your HIT query x_t, then you successfully hit the fly at time t and your algorithm terminates. Otherwise **(2)** you continue to the next time step $t+1$, for which you again pick an index to HIT and the fly position will update (based on the previously provided formula). Thus notice your algorithm simply provides an integer in $\{0, \cdots, N-1\}$ to HIT at each time step $t = 1, 2, \cdots$ until termination, and all you are told is whether you hit the fly or not.

Task summary: Given the room size N, design an algorithm that is guaranteed to eventually HIT the fly in the fewest number of time steps possible, where the starting index i and speed k are unknown. In other words, it should be guaranteed to terminate by some finite time step M, and you want M to be as small as possible. See details on solution requirements and grading below, and see the Example for an example setup and algorithm (that fails).

Grading

Your solution must include **(i)** a description of your algorithm, which in particular should clearly state which index x_t you are hitting at all time steps t that your algorithm will run for. You must also include **(ii)** a rigorous proof that explains why your algorithm is *correct*, that is, why your algorithm is guaranteed to eventually hit the fly. If you've never written a proof before or need some quick refreshers, feel free to drop by office hours to get some pointers!

Finally, your solution must *also* **(iii)** state and prove the worst-case asymptotic runtime ("big O" notation) of your algorithm in terms of N — i.e., how many time steps your algorithm will run for in the *worst case*. Recall that the initial position i and the speed k are unknown to you, so these should not be included in your stated runtime (instead, these being unknown might factor in to your "worst-case" considerations).

Important: Full credit will *only* be awarded for algorithms that are sub-quadratic in N (i.e., the worst-case runtime of the algorithm must be $o(N^2)$, which means strictly lower asymptotically than N^2, rather than $O(N^2)$). Partial credit will be awarded for algorithms with valid proofs with runtimes that do not meet this condition. No credit will be awarded for solutions that lack any of the following: an algorithm description, statement of the worst-case runtime, valid proof of correctness, and a valid proof of the runtime.

Example on next page.

Example

Let's try out the following test example: suppose $N = 6$, $i = 4$, and $k = 2$. Then we have $p_0 = i = 4$, $p_1 = (p_0 + 2) \mod 6 = 0$ since the fly wraps around, $p_2 = (p_1 + 2) \mod 6 = 2$, $p_3 = (p_2 + 2) \mod 6 = 4$, and so on.

Suppose your algorithm is the following: "at each time step t, always `HIT` index $x_t = 1$". This algorithm fails the task because we can show that it is *not* guaranteed to always eventually hit the fly; in particular, we can see that it will never hit the fly for the above test example. This is because the fly's path will cycle through $4 \to 0 \to 2 \to 4 \to \cdots$, which will never land at index 1.