**Find a duplicate, Space Edition™ BEAST MODE**

In [Find a duplicate, Space Edition™](https://www.interviewcake.com/question/cpp/find-duplicate-optimize-for-space), we were given a vector of integers where:

1. the integers are in the range 1..n1..*n*
2. the vector has a length of n+1*n*+1

These properties mean the vector *must have at least 1 duplicate*. Our challenge was to find a duplicate number without modifying the input and optimizing for *space*. We used a divide and conquer approach, iteratively cutting the vector in half to find a duplicate integer in O(nlg⁡n)*O*(*n*lg*n*) time and O(1)*O*(1) space (sort of a modified binary search).

But we can actually do *better*. **We can find a duplicate integer in O(n)*O*(*n*)time while keeping our space cost at O(1)*O*(1)**.

This is a tricky one to derive (unless you have a strong background in graph theory), so we'll get you started:

**Imagine each item in the vector as a node in a linked list**. In any linked list, ↴ each node has a **value** and a **"next"** pointer. In *this* case:

* The **value** is the *integer* from the vector.
* The **"next"** pointer points to the **value-eth** node in the list (numbered starting from 1). For example, if our value was *3*, the "next" node would be the *third* node.

Here’s a full example:

**Notice we're using "positions" and not "indices."** For this problem, we'll use the word "position" to mean something *like* "index," but different: indices start at 0, while positions start at 1. More rigorously: position = index + 1.

Using this, **find a duplicate integer in O(n)*O*(*n*) time while keeping our space cost at O(1)*O*(1)**. Just like before, **don't modify the input.**

**Drawing pictures will help a lot with this one.** Grab some paper and pencil (or a whiteboard, if you have one).