



Master algorithms together on [Binary Search](#)! Create a room, invite your friends, and race to finish the problems.



Daily Coding Problem

[Blog](#)

Daily Coding Problem #15

Problem

This problem was asked by Facebook.

Given a stream of elements too large to store in memory, pick a random element from the stream with uniform probability.

Solution

Naively, we could process the stream and store all the elements we encounter in a list, find its size, and pick a random element from $[0, \text{size} - 1]$. The problem with this approach is that it would take $O(N)$ space for a large N .

Instead, let's attempt to solve using loop invariants. On the i th iteration of our loop to pick a random element, let's assume we already picked an element uniformly from $[0, i - 1]$. In order to maintain the loop invariant, we would need to pick the

ith element as the new random element at $1 / (i + 1)$ chance. For the base case where $i = 0$, let's say the random element is the first one. Then we know it works because

- For $i \geq 0$, before the loop began, any element K in $[0, i - 1]$ had $1 / i$ chance of being chosen as the random element. We want K to have $1 / (i + 1)$ chance of being chosen after the iteration. This is the case since the chance of having being chosen already but not getting swapped with the i th element is $1 / i (1 - (1 / (i + 1)))$ which is $1 / i \cdot i / (i + 1)$ or $1 / (i + 1)$

Let's see how the code would look:

```
import random

def pick(big_stream):
    random_element = None

    for i, e in enumerate(big_stream):
        if random.randint(1, i + 1) == 1:
            random_element = e
    return random_element
```

Since we are only storing a single variable, this only takes up constant space!

By the way, this is called [reservoir sampling](#)!

