Observations about lookup speed

- We know that array lookups for a given index are O(1)
- Idea: we can store elements in an array, at their own index
- A = [1, 2, 3, 4, 5, 6, ..., n]
- Then, we can check if an element is there by checking that index in O(1) time
- Downside: space complexity will need to be O(max(A) - min(A)) to store elements uniquely, and will need to resize array whenever larger element is added.
    - Example: A = [1, 1,000,000] will require 1,000,000 slots to store 2 items

Hashing

- Idea: put elements into a fixed-size array by taking the remainder
  - A = [1, 5, 6, 3, 16, 20], with size of hash table = 6
  - This gives hash table H = [6, 1, 20, 3, 16, 5]
- Okay in the above example, but what about A = [6, 12, 1, 13, 2, 14]? Now have multiple elements that hash to same thing: 6 mod 6 = 12 mod 6 = 0.
- Solution: use a list to keep track of all elements that hash to same index, and instead just store a pointer to the list in H if needed:
  - H = [[6, 12], [1, 13], [2, 14], -1, -1, -1] (fill unused slots with -1)