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)

Hash function effectiveness

- Downside of simple dividing/remainder hash function: can be bad if data has lots of elements that have the same remainder mod (size of H):
  - A = [1, 1, 1, 1, 1, 1, 1, 1]
  - H = [[1, 1, 1, 1, 1, 1, 1, 1], -1, -1, -1, -1, -1]
  - This puts us back at the O(n) time to find an element in a list (not good)
- Solution: use more complex hash function or methodology to distribute items more evenly. Good hash functions will expect a constant number of elements (so, O(1) lookup time) in each bucket at most, not n elements.
- Result: **expected** O(1) lookup time.