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.

Resizing hash tables

- Fine so far, but will eventually be bad
- If n = 100 and size of hash table is 5, then will be (at best) 20 elements per index = n/5. Not good - as n increases, number of elements per bucket will be only a constant factor times n, so will eventually be O(n) lookup again.
- Solution: resize hash table when there are too many elements in the table.
- Load factor: number of elements in table / number of table slots.
  - Typically want to resize hash table when load factor is around 0.75, but can vary by implementation.