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.

Resizing hash tables (cont.)

- How can we resize efficiently? Do we need to rehash every single element? Would take O(n) time (bad)!
- Answer: doesn't matter because we resize so infrequently.
- Insertion will be **amortized** O(1) time:
  - Only resize once the load factor is 0.75
  - When we resize, make backing array twice as large
  - So, if we insert n elements, will only need to resize the array something like log(n) times - but each of them will only require $n/2^k$ rehashes, the sum of which is O(n): $\sum_{k=0}^{n} \frac{n}{2^k} = n \sum_{k=0}^{n} \frac{1}{2^k} = 2n$
- So, the total runtime impact of rehashes is O(2log(n)) = O(log(n)). However, we are already adding n elements, so the total amount of runtime for inserting n elements is O(log(n)) + O(n) = O(n).