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).

Set ADT

- ADT: abstract data type. Basically, some expectations we can have for a data structure, as to what operations it supports
- Set: Mathematical meaning of set, says whether something is present or not (no duplicates)
- Set ADT:
  - Insert(element) ⇒ (amortized) O(1)
  - Remove(element) ⇒ (amortized? depends on implementation) O(1)
  - Contains(element) ⇒ (expected) O(1)