Open Addressing

Deletion requires special handling.

- can re-insert all elements following the deleted element
  - if the load factor is low enough, this should only be a small number of elements
  - limited to sequential probing

delete 24

delete 10

delete 35

delete 18

0	1	2	3	4	5	6
35	11	-	10	18	24	5

Open Addressing

Deletion requires special handling.

- can mark empty slot as “deleted” – find continues on, insert can fill
  - drawback: probe sequence lengths are based on the largest the collection has been, not the current size
  - solution: can periodically re-hash everything to clean up

delete 24

delete 10

delete 35

delete 18

0	1	2	3	4	5	6
35	11	-	10	18	24	5

Hashtables

If done properly, hashtables provide O(1) expected time for find, insert, remove – once h(k) has been computed.

- “done properly” means load factor isn't too high and is kept bounded, and there is good distribution of hash values

Computing h(k) can take time.

- e.g. for strings, computing h(k) = O(|k|) … which reduces to O(1) if |k| is bounded, but must be considered as O(|k|) otherwise

Worst-case behavior is O(n) for find and remove, unless separate chaining + a fancier bucket implementation is used (which has memory overhead).

- worst case occurs when key distribution is poor and load factor is high

Hashtables

What about other operations?

- initialization
  - O(N) – size of the array used for the hashtable
- traversal
  - in most cases O(n+N) for separate chaining – must examine each index of table as well as all elements
    - can be worse e.g. worst case dynamic perfect hashing
  - O(N) for open addressing
- find next larger/smaller key, find min/max key
  - full traversal is required because h(k) does not preserve original ordering of keys