Hash Tables (Cont'd)

Carlos Moreno
cmoreno@uwaterloo.ca
EIT-4103

https://ece.uwaterloo.ca/~cmoreno/ece250

These slides, the course material, and course web site are based on work by Douglas W. Harder

Hash Tables

Standard reminder to set phones to silent/vibrate mode, please!

Hash Tables

- Today's class:
  - Investigate the other important technique to deal with collisions, namely, *open addressing* or *probing*.
    - If a bin is taken, probe other bins (following a given pattern) until we find one that is empty.
  - We'll deal with two variations (two patterns of selecting the sequence of bins to probe):
    - Linear probing
    - Double hashing

Hash Tables

- It's really quite easy — let's look at linear probing:
- If a bin is taken, check the next one (in loop, so that if the next one is also taken, you continue to the next one, and so on)
  - When an empty bin is found, place the element there

Hash Tables

- It's really quite easy — let's look at linear probing:
- If a bin is taken, check the next one (in loop, so that if the next one is also taken, you continue to the next one, and so on)
  - When an empty bin is found, place the element there
  - It's that easy!! So, really, that's it — we're done with today's lesson!!!

Hash Tables

- Oh, wait ... we have to cover double hashing as well....

Hash Tables

- Oh, wait ... we have to cover double hashing as well....
- And OH, WAIT!! There's the half-a-ton of related little details that we have to go over ...

Hash Tables

- Oh, wait ... we have to cover double hashing as well....
- And OH, WAIT!! There's the half-a-ton of related little details that we have to go over ...
  - But kidding aside, that really *is* the whole story with the notion of linear probing; let's look at how it works and figure out the little details that get in the way when using the technique.

Hash Tables

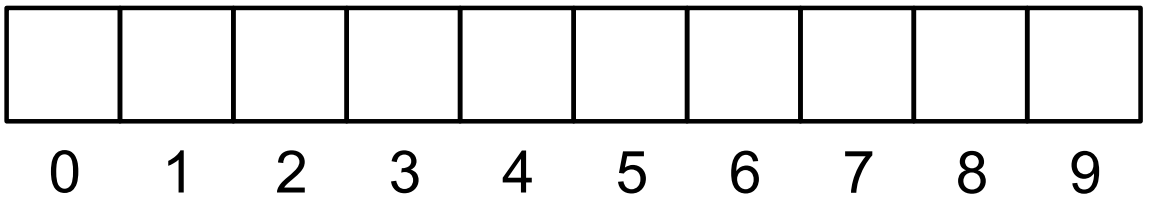
- So, let's look at an example — as usual, our examples (intended for human brains to process, and not for CPUs) will use hash table size 10, and the hash will be the value modulo 10 (i.e., the last digit)

Hash Tables

- The example:
- Insert the numbers

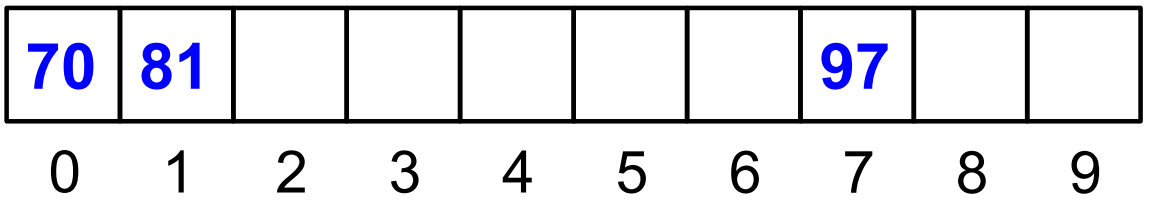
  81, 70, 97, 60, 51, 38, 89, 68, 24

  into the initially empty hash table:



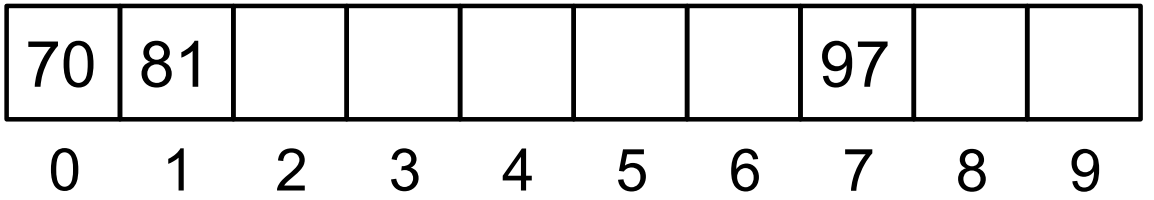
Hash Tables

- The first three are easy — no collision so far, so each element goes in its corresponding bin:



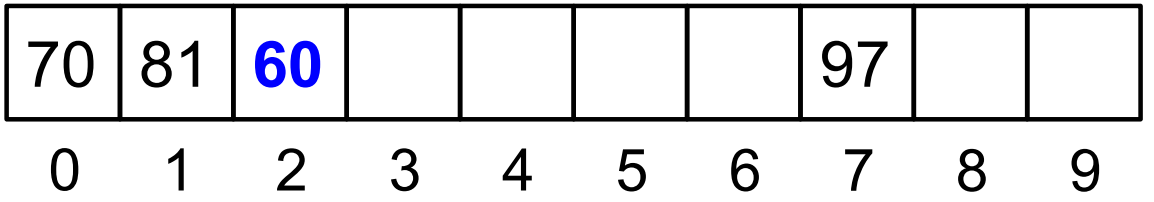
Hash Tables

- Inserting 60 causes a collision in bin 0, therefore, we check:
  - Bin 1 (also full), and
  - Bin 2 (empty)



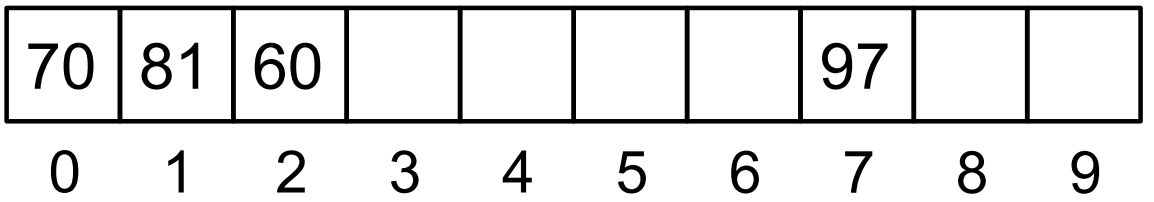
Hash Tables

- Inserting 60 causes a collision in bin 0, therefore, we check:
  - Bin 1 (also full), and
  - Bin 2 (empty)



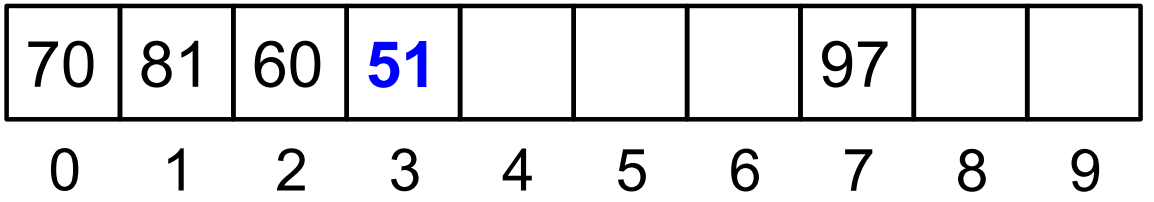
Hash Tables

- Inserting 51 also causes a collision, this time, in bin 1, therefore, we check:
  - Bin 2 (also full), and
  - Bin 3 (empty)



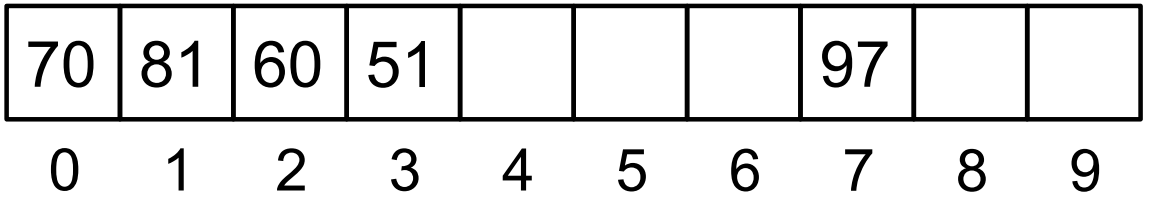
Hash Tables

- Inserting 51 also causes a collision, this time, in bin 1, therefore, we check:
  - Bin 2 (also full), and
  - Bin 3 (empty)



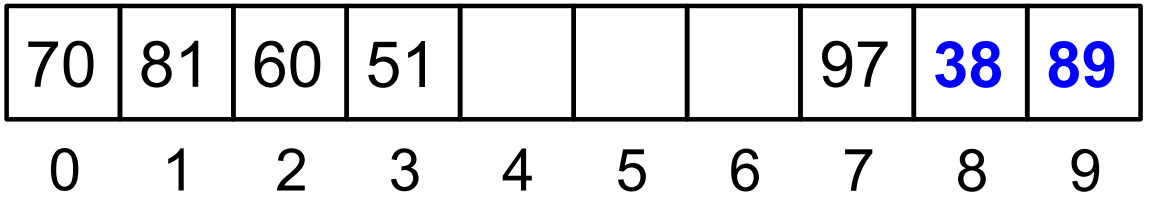
Hash Tables

- 38 and 89 can be placed into bins 8 and 9 respectively without collisions



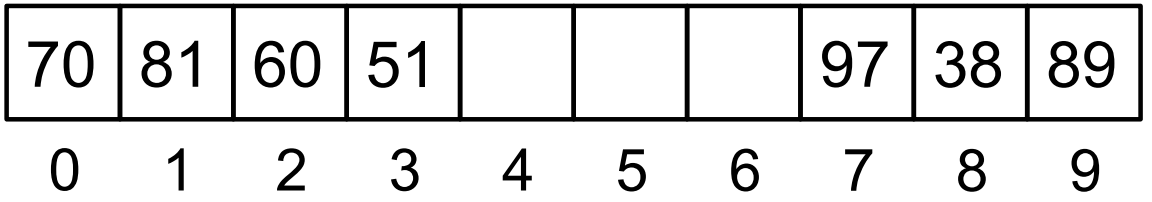
Hash Tables

- 38 and 89 can be placed into bins 8 and 9 respectively without collisions



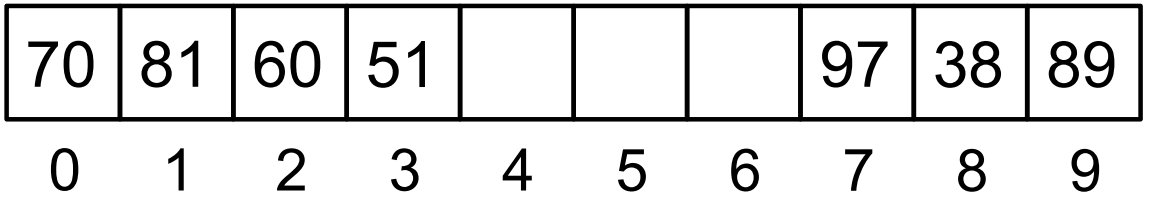
Hash Tables

- 68 causes a collision, and it looks like we're in pretty bad shape, since we run out of “next bins” to probe for available space...



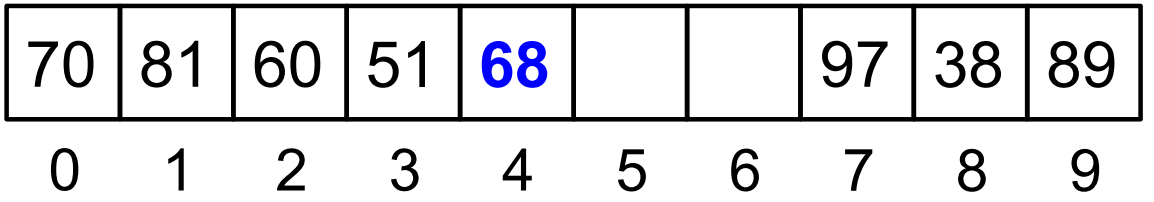
Hash Tables

- 68 causes a collision, and it looks like we're in pretty bad shape, since we run out of “next bins” to probe for available space...
- Any ideas?



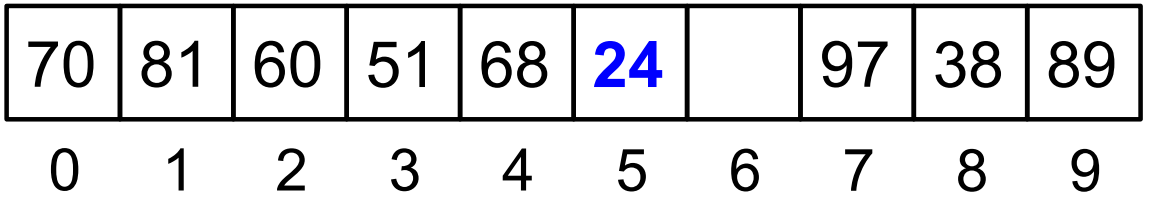
Hash Tables

- Of course — we can always use the underlying array similar to the way we use a *circular* array!
  - Check bins 9, 0, 1, 2, 3, and finally 4 which is empty
  - Insert 68 into bin 4



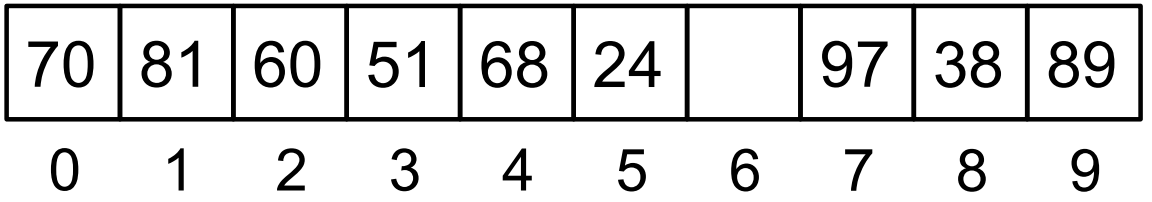
Hash Tables

- Inserting 24 causes a collision in bin 4, however the next bin is empty



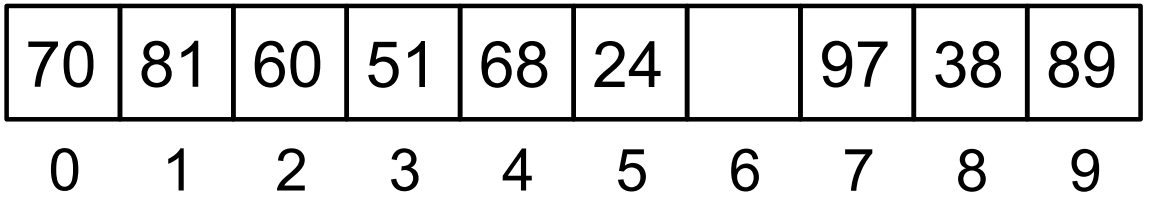
Hash Tables

- An observation:
  - If a hash table using linear probing is too crowded (i.e., a large fraction of the array is taken), we will spend a lot of time probing (back to this in a few minutes)



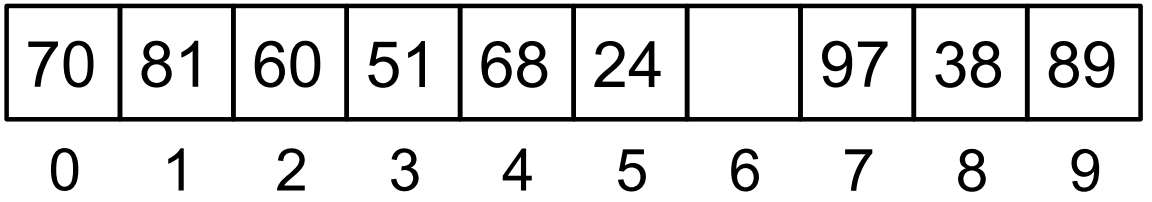
Hash Tables

- Another problem: (maybe?)
  - How do we look up values, if they maybe nowhere near where they're supposed to go? (e.g., what is 68 doing in bin 4??)



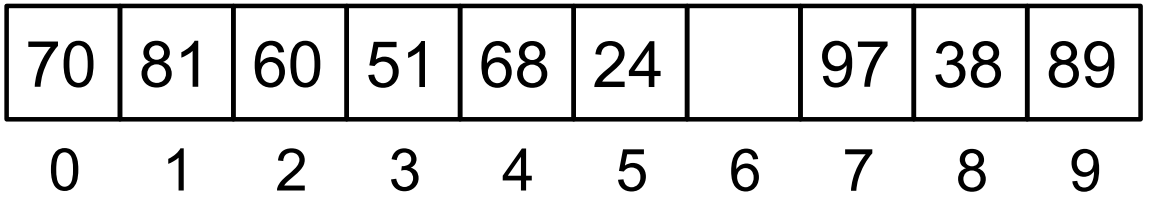
Hash Tables

- Actually, it should be pretty clear that this is not a problem — exactly as we determined that bin 4 is where it should go, we will be able to determine (following the same procedure) that bin 4 is where it is!



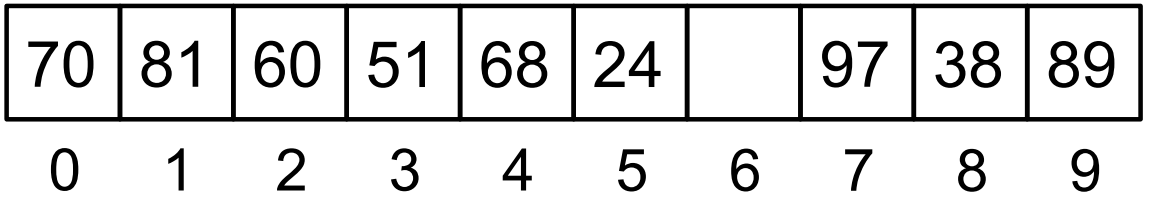
Hash Tables

- Compute the hash of the value that we're looking up
- Check the content of the bin given by that hash
  - If the value is not there, continue searching forward until:
    - The value is found
    - An empty bin is found (meaning that the value is not present)



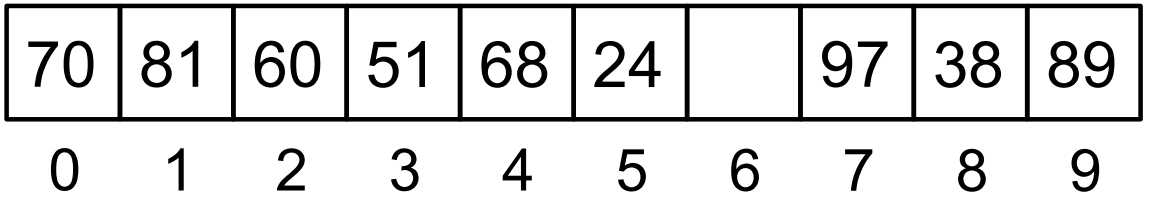
Hash Tables

- Searching for 68
  - Examine bins 8, 9, 0, 1, 2, 3, and 4, finding 68 in bin 4
- Searching for 23
  - Search bins 3, 4, 5, and 6
  - The last bin is empty; therefore 23 is not in the table



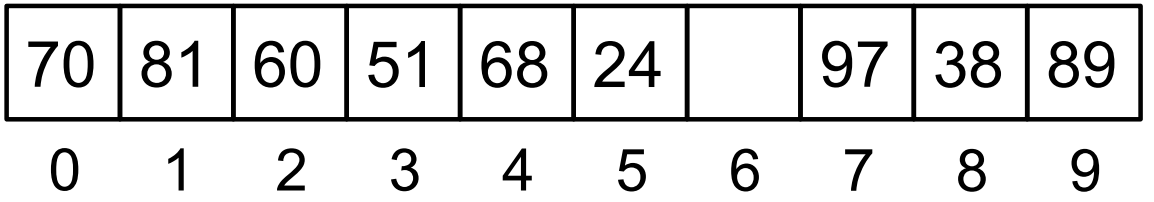
Hash Tables

- Another operation that we'd like:
  - How do we remove an element from the hash table?



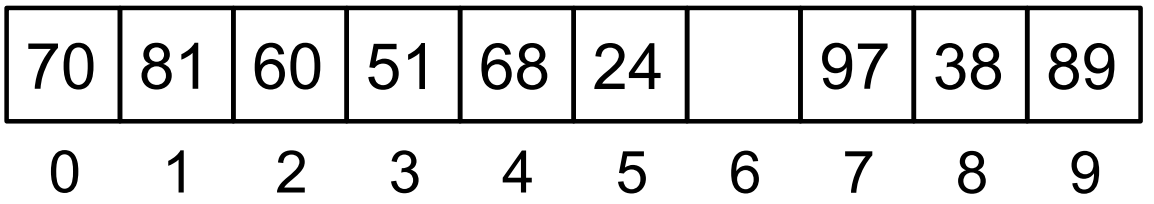
Hash Tables

- Another operation that we'd like:
  - How do we remove an element from the hash table?
  - Easy enough, right? The exact same way that we add them, or find them, we can remove them!



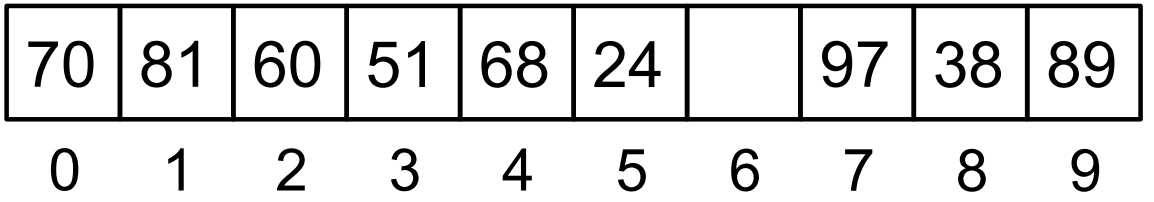
Hash Tables

- Another operation that we'd like:
  - Errrm... Are we sure?



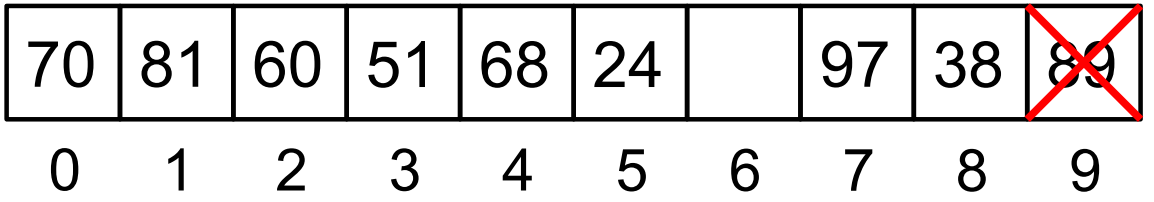
Hash Tables

- Another operation that we'd like:
  - Errrm... Are we sure?
  - Let's try one — let's remove 89...



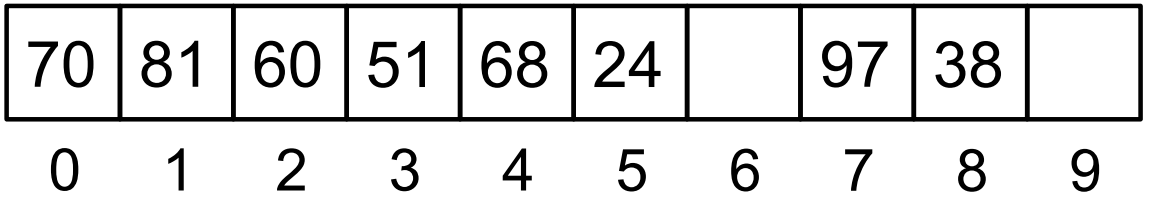
Hash Tables

- Another operation that we'd like:
  - Errrm... Are we sure?
  - Let's try one — let's remove 89... We locate its bin, 9, and erase it (search forward if needed):



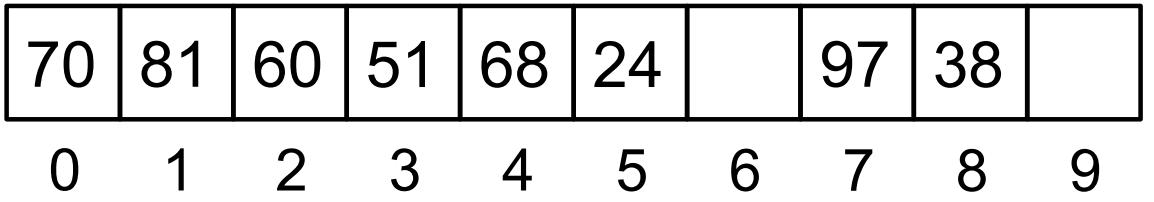
Hash Tables

- Another operation that we'd like:
  - Worked like a charm, right?



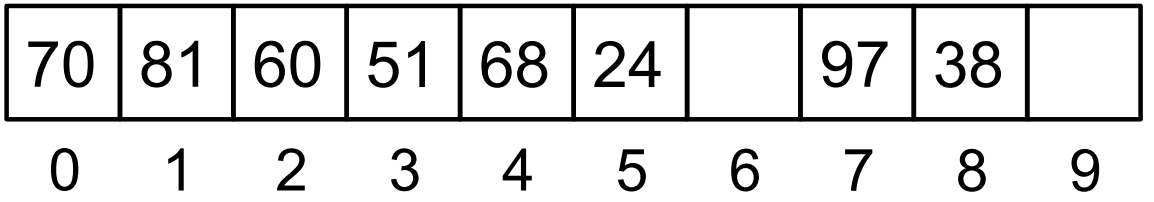
Hash Tables

- Another operation that we'd like:
  - Worked like a charm, right?
    - Really?



Hash Tables

- Another operation that we'd like:
  - Worked like a charm, right?
    - Really?
    - Hint: where's 68?



Hash Tables

- The positions of elements depend on chunks of occupied bins (with no holes).
- Opening holes on those chunks is likely to cause trouble (in this case, elements that are in the hash table can no longer be found).

Hash Tables

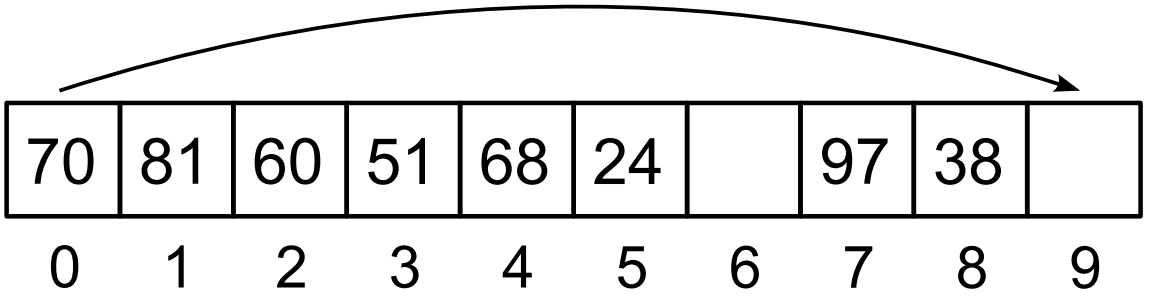
- So, what do we do?

Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?

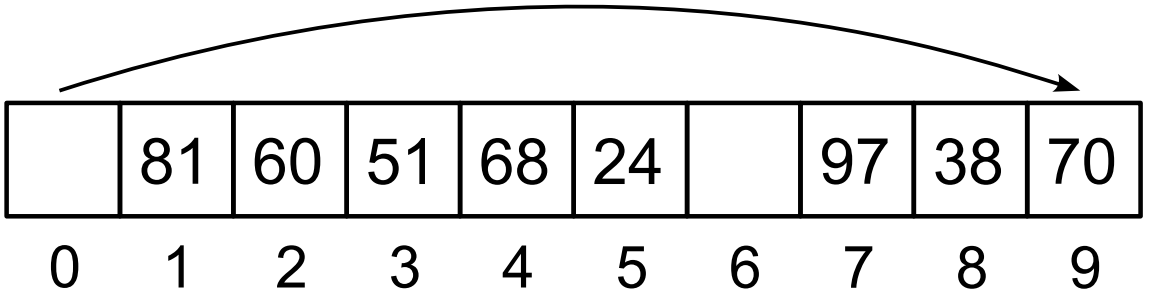
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



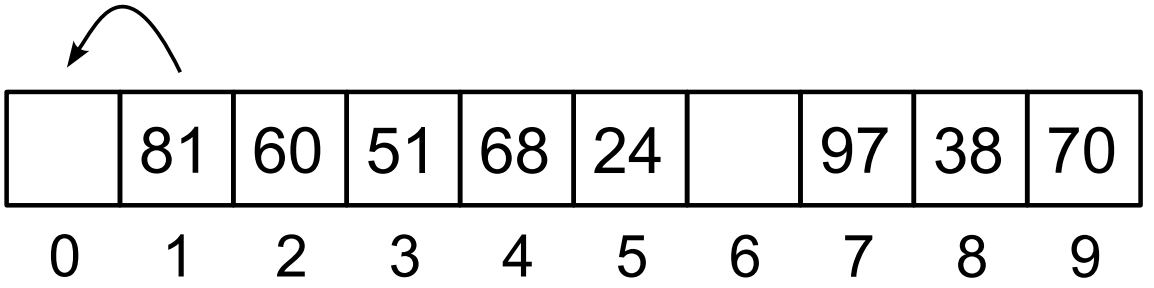
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



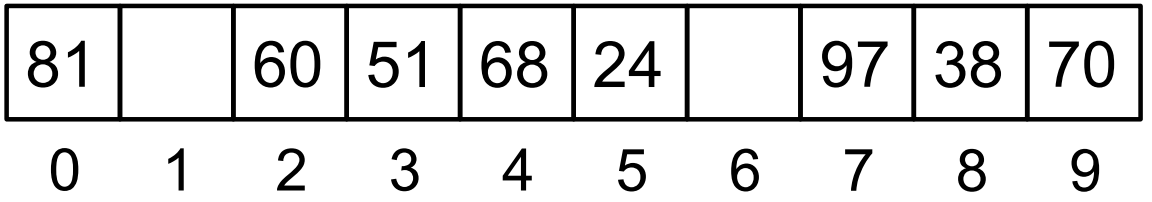
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



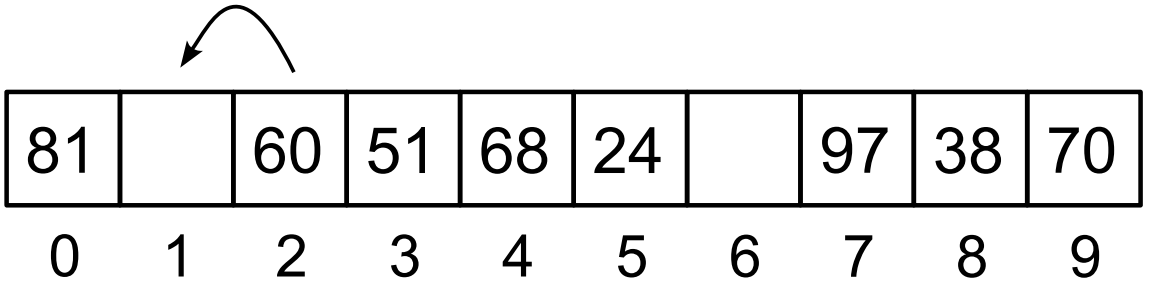
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



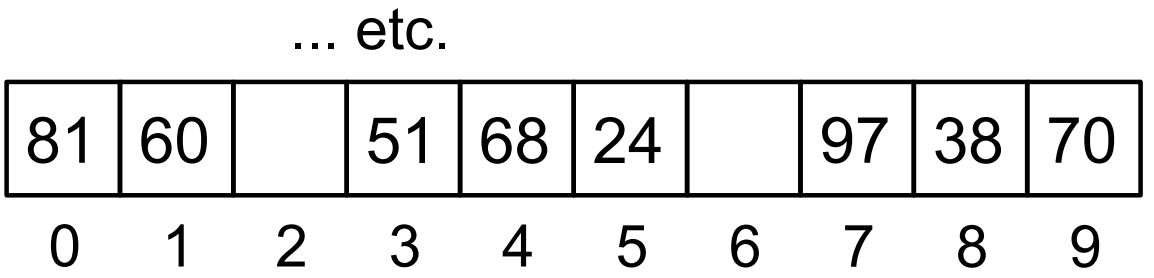
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



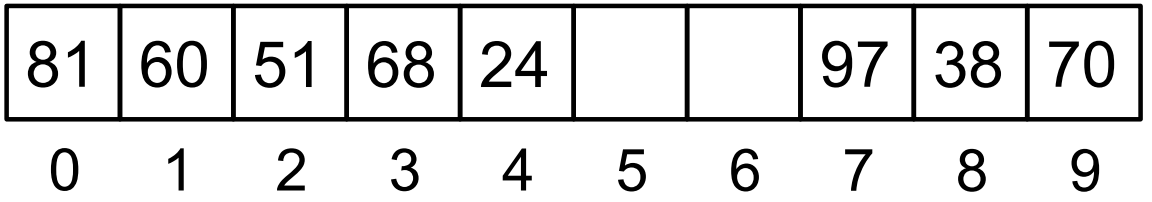
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



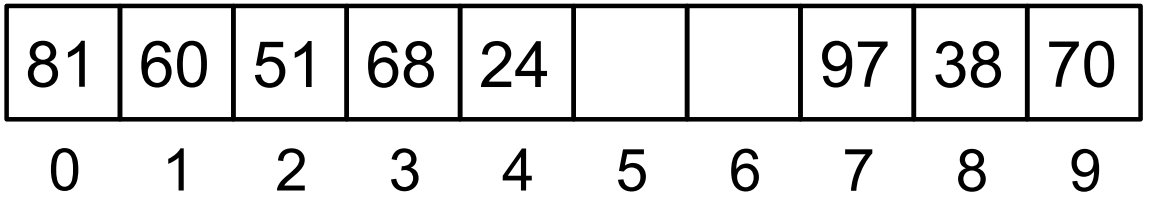
Hash Tables

- So, what do we do?
- How about we shift the elements that follow?
  - In this example, the table is almost full, so it looks like a linear time operation, but in practice, it should be just a few elements, right?



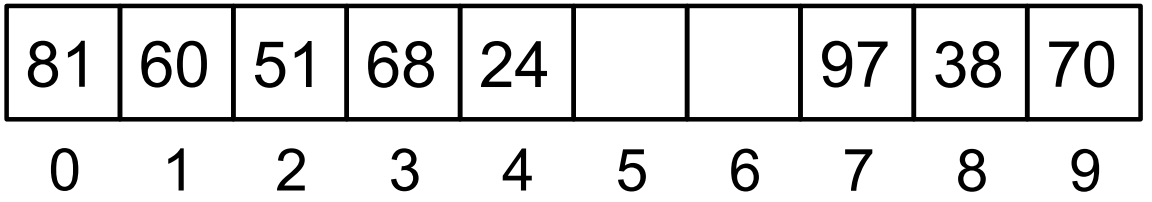
Hash Tables

- Perfect! Now 68 can be found again...



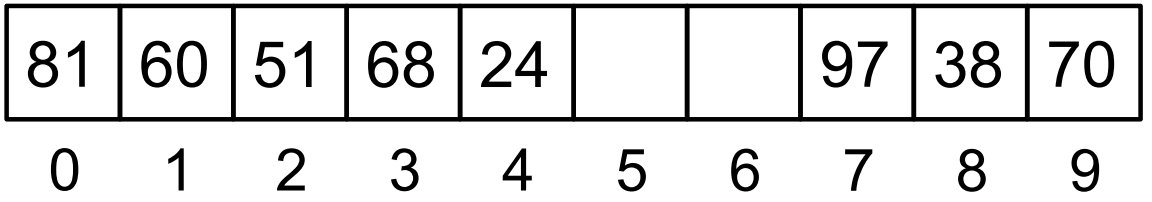
Hash Tables

- Perfect! Now 68 can be found again...
- Errm... Where's 70??



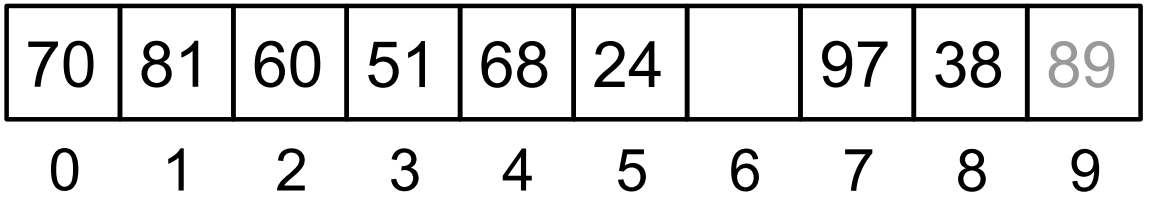
Hash Tables

- Perfect! Now 68 can be found again...
- Errm... Where's 70??
  - In fact, any element that was in the bin corresponding to its hash would now become impossible to find!



Hash Tables

- So, we undo and forget about this idea of shifting the elements that follow...
- Then... what do we do??



Hash Tables

- Class activity (hopefully 5 to 10 minutes ± ϵ):
  - Team up (groups of two to four people, maybe?) and brainstorm to try to come up with a solution
    - There are several possible solutions, not all of them good and efficient, but still valid! Hopefully some of you guys will find some of the efficient ones?

70	81	60	51	68	24		97	38	89
0	1	2	3	4	5	6	7	8	9

Hash Tables

- We'll look at the solutions in class

- I estimate that we'll have run out of time at this point, so we'll (most likely) look at double hashing next class.

Summary

- During today's class, we discussed:
  - Linear probing as a mechanism to deal with collisions.
  - Difficulties associated with this technique.
  - Discussed techniques to efficiently delete elements from the hash table.