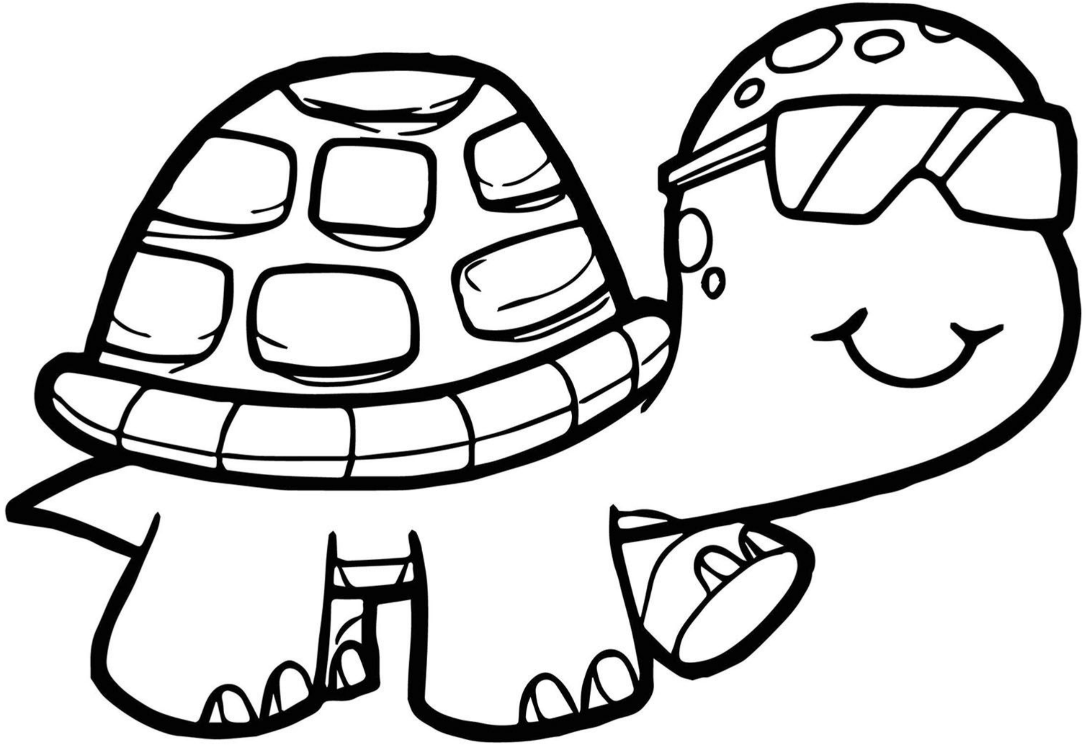
Coloring AND a riddle… now we’re talking

I have keys but no locks.
I have a space but no room.
You can enter, but you can’t go outside.
What am I?



LLRBs & Hashing

Discussion 08

Announcements

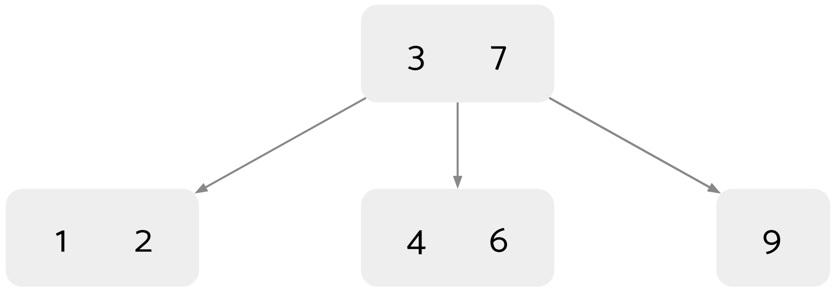
Week 8

- ❑ Weekly Survey 7 - due this Monday 03/08
- ❑ Lab 8 - due this Friday 03/12
- ❑ Project 2 Checkpoint - due this Friday 3/12
- ❑ HW 2 - due next Monday 3/15

Content Review

B-Trees

B-Trees are trees that serve a similar function to binary trees while ensuring a bushy structure. In this class, we'll often use B-Tree interchangeably with 2-3 Trees.

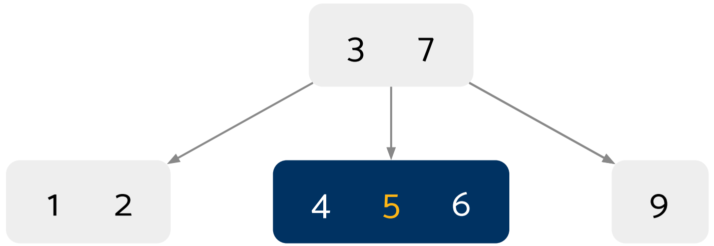


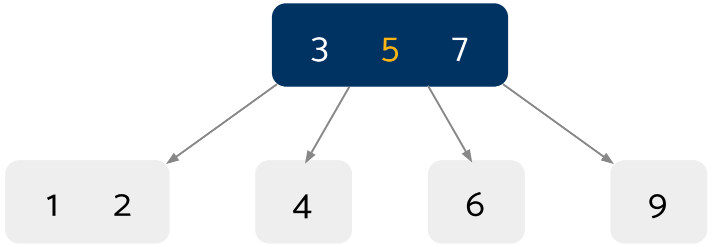
Each node can have up to 2 items and 3 children (there are variations where these values are higher known as 2-3-4 trees).

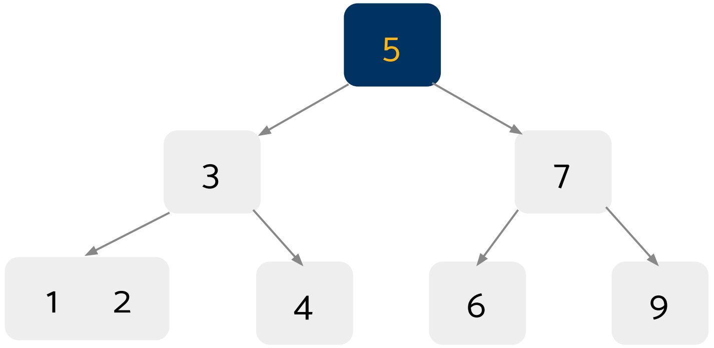
All leaves are the same distance from the root, which makes getting take Θ(log N) time.

B-Trees

When adding to a B-Tree, you first start by adding to a leaf node, and then pushing the excess items up the tree until it follows the rules from the last slide.

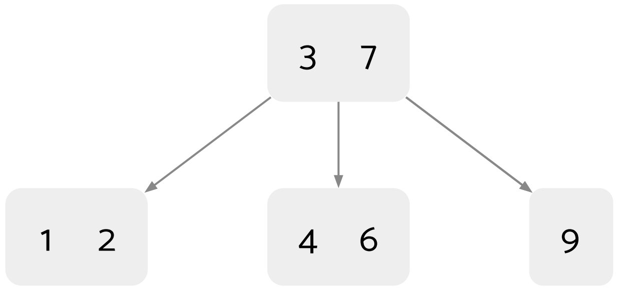


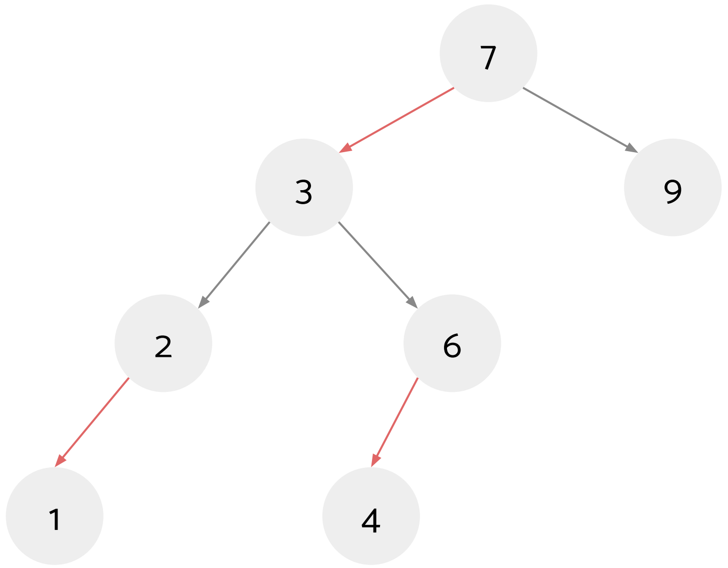




Left Leaning Red Black Trees

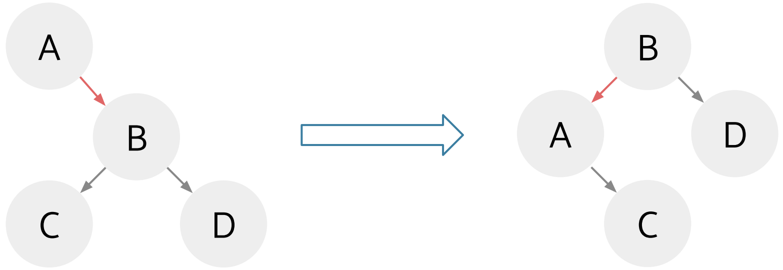
LLRBs are a representation of B-trees that we use because it is easier to work with in code. In an LLRB, each multi-node in a 2-3 tree is represented using a red connection on the left side.



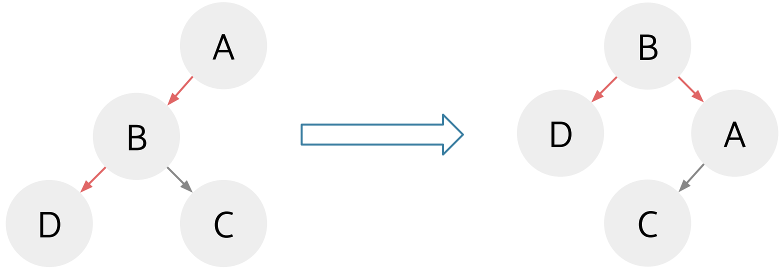


LLRB Balancing Operations

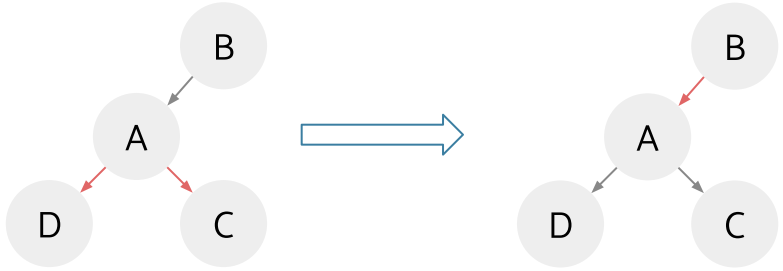
`rotateLeft(A);`



`rotateRight(A);`



`colorFlip(A);`



Hashing

Hash functions are functions that represent an object using an integer. We use them to figure out which bucket of our hashset the item should go in.

Once we have a hash for our object we use mod to find out which bucket it goes into.

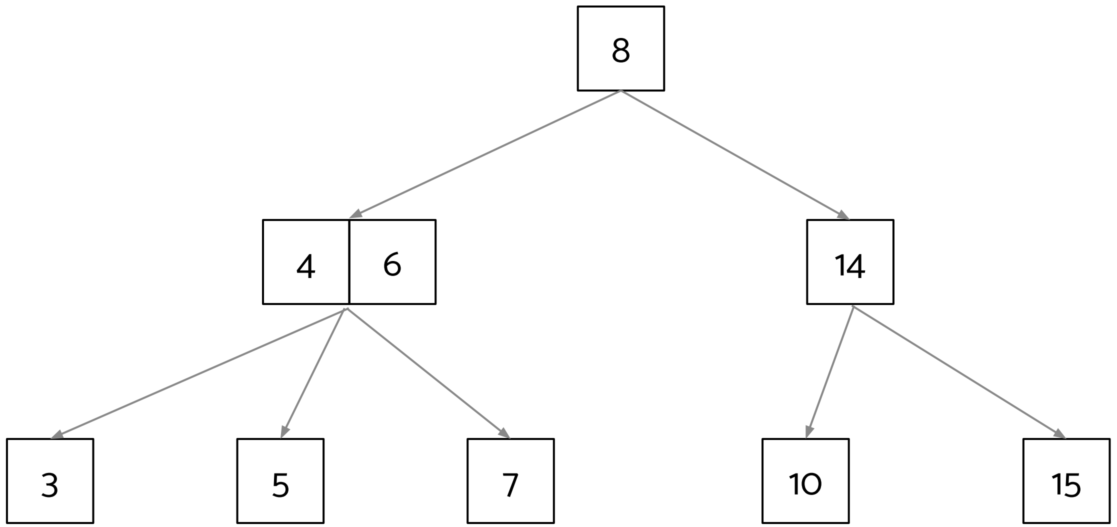
In each bucket, we deal with having lots of items by chaining the items and using .equals to find what we are looking for.

It is important that your `.equals()` function matches the result of comparing hashcodes - if two items are equal, they must also have the same hashcode

Worksheet

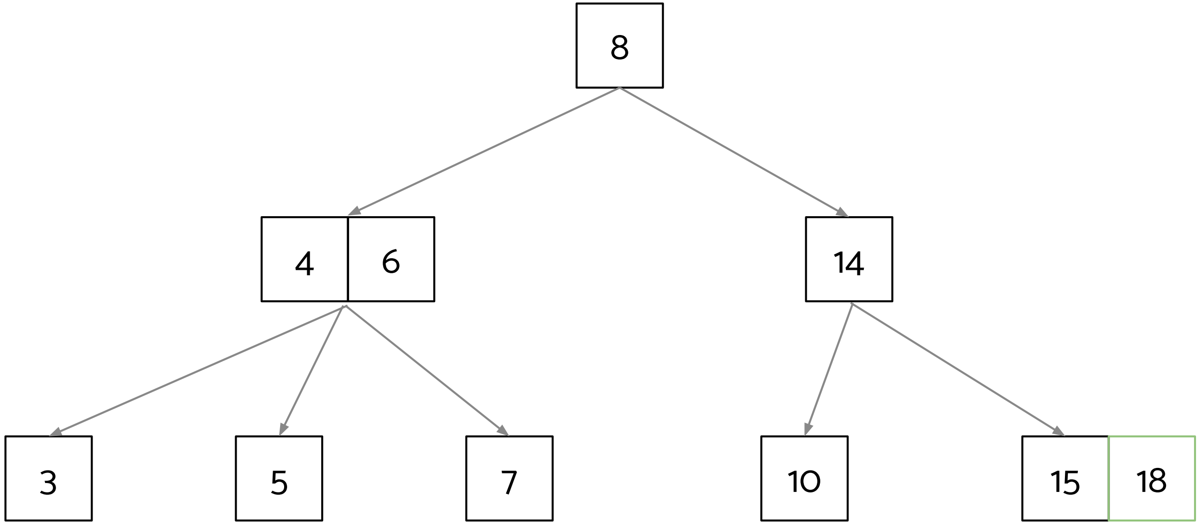
1A 2-3 Trees and LLRBs

Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



1A 2-3 Trees and LLRBs

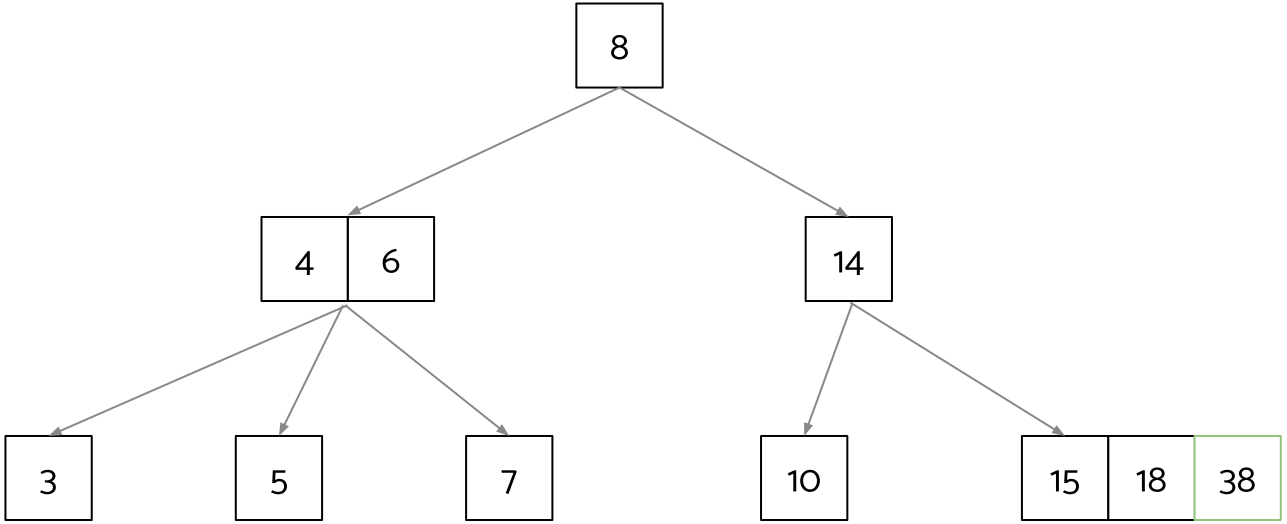
Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 18

1A 2-3 Trees and LLRBs

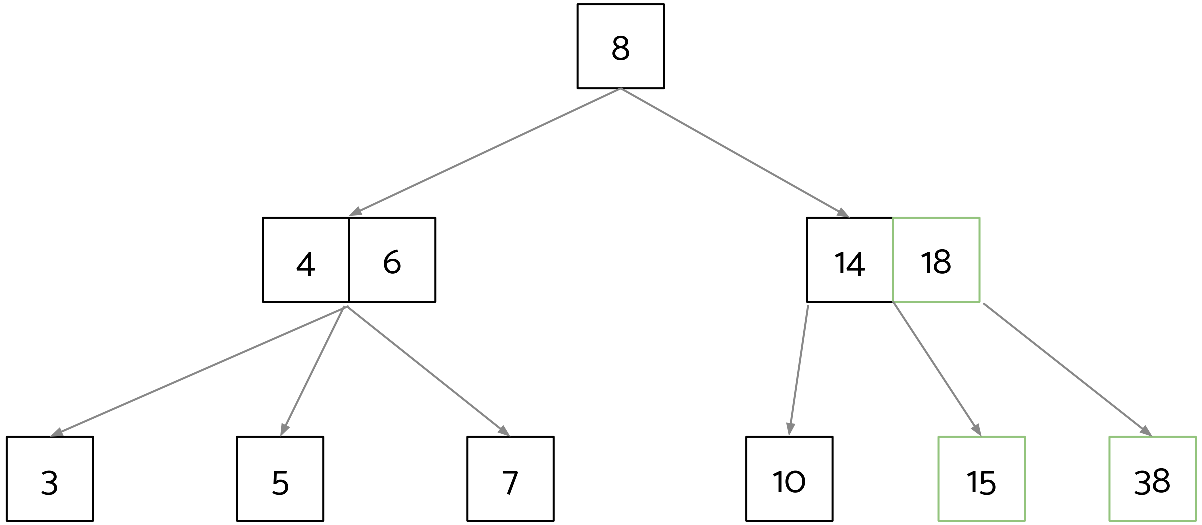
Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 38 (part 1)

1A 2-3 Trees and LLRBs

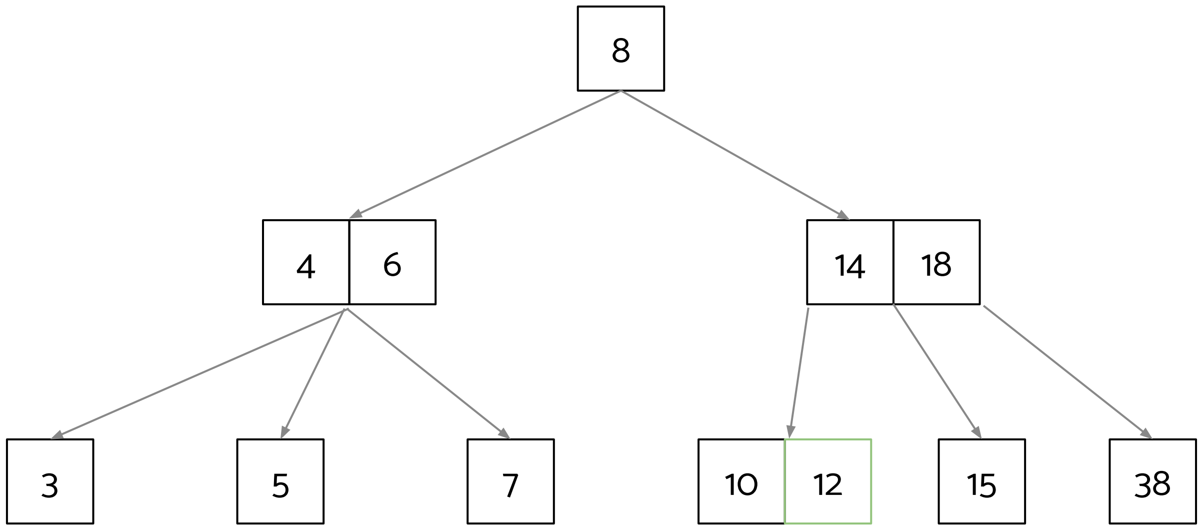
Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 38 (part 2)

1A 2-3 Trees and LLRBs

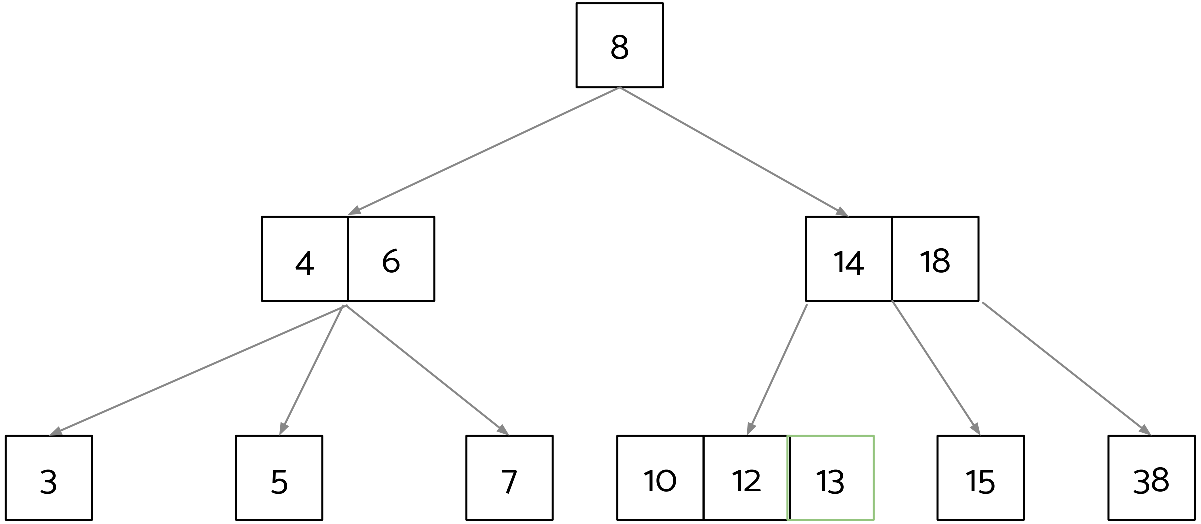
Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 12

1A 2-3 Trees and LLRBs

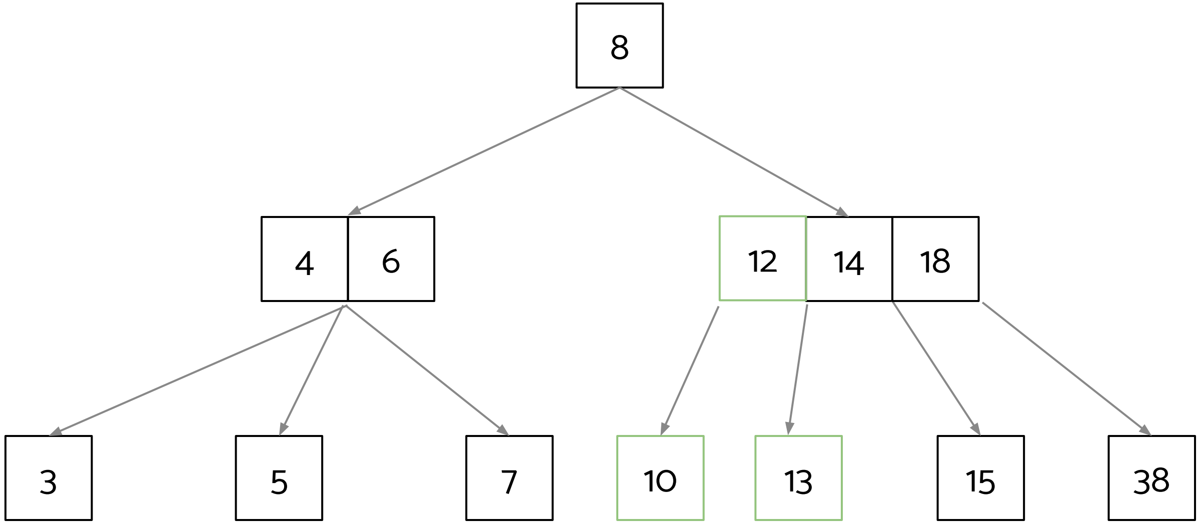
Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 13 (part 1)

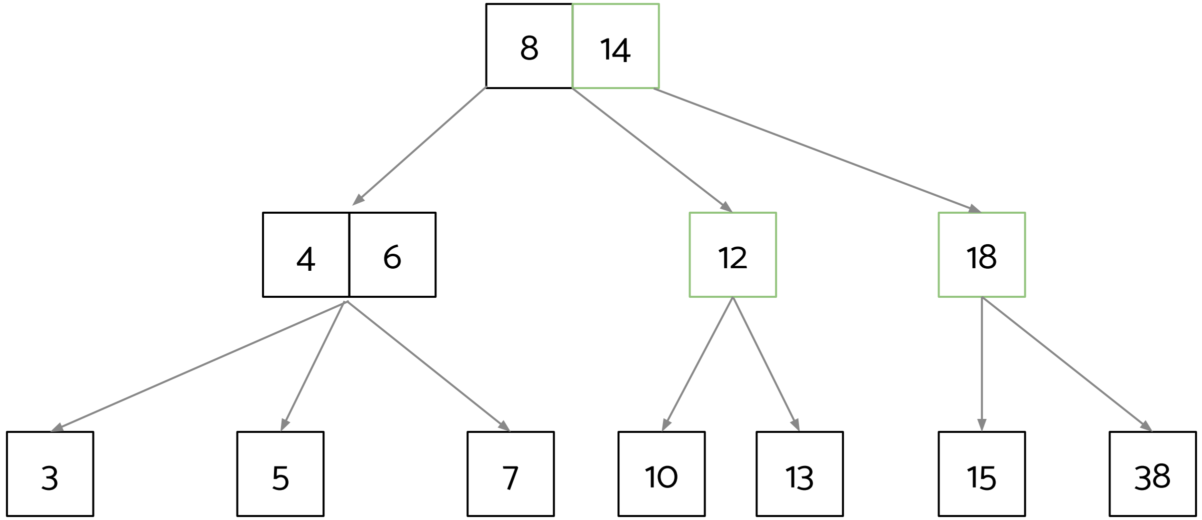
1A 2-3 Trees and LLRBs

Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 13 (part 2)

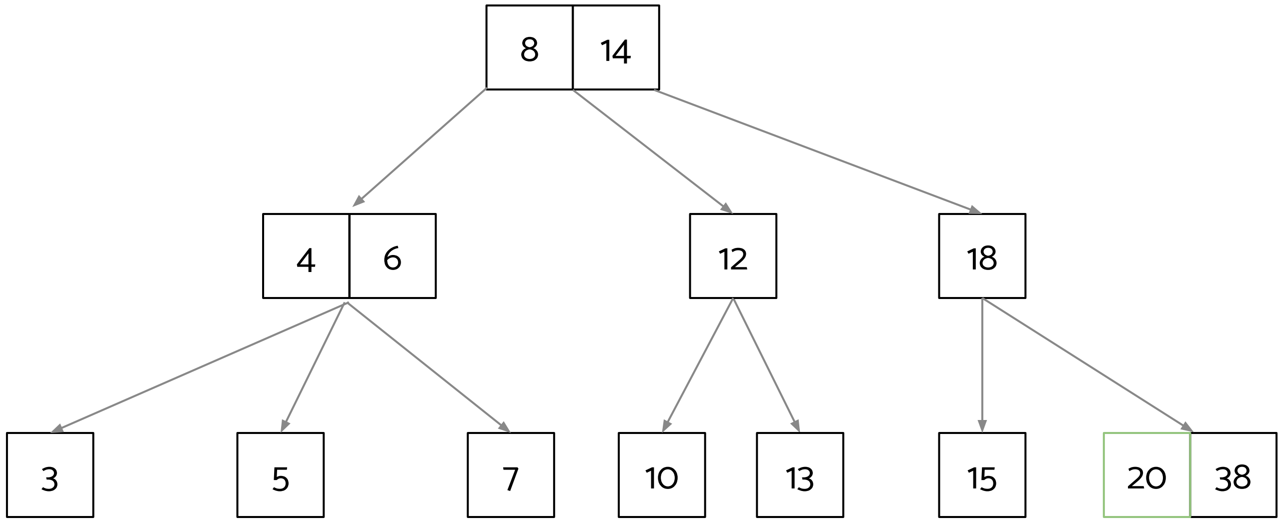
1A 2-3 Trees and LLRBs Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 13 (part 3)

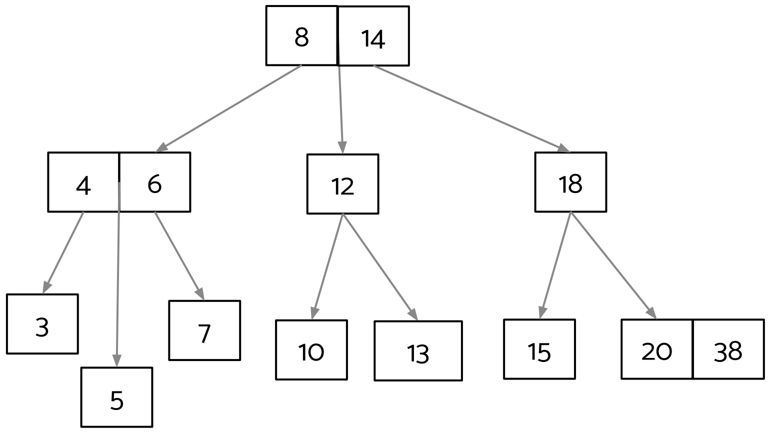
1A 2-3 Trees and LLRBs

Draw what the following 2-3 tree would look like after inserting 18, 38, 12, 13, and 20.



Inserting 20.

1B 2-3 Trees and LLRBs Convert the resulting 2-3 tree to a red-black tree.



1C 2-3 Trees and LLRBs

If a 2-3 tree has depth **H** (that is, the leaves are at distance **H** from the root), what is the maximum number of comparisons done in the corresponding red-black tree to find whether a certain key is present in the tree?

2A Hashing

Are the following hashCodes valid? If they are, what are the advantages or disadvantages?

```
public int hashCode() {
     return -1;
}
```

```
public int hashCode() {
     return intValue() * intValue();
}
```

```
public int hashCode() {
     return super.hashCode();
}
```

2B Hashing

1. When you modify a key that has been inserted into a HashMap will you be able to retrieve that entry again? Explain.

2. When you modify a value that has been inserted into a HashMap will you be able to retrieve that entry again? Explain.

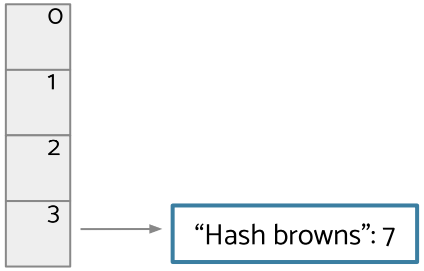
3A A Side of Hash Browns Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



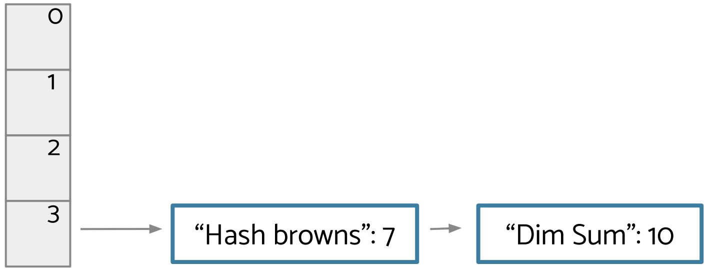
N = 1
M = 4
N/M = 0.25

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns

Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



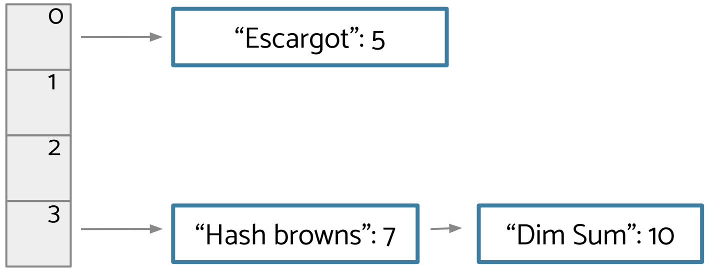
N = 2
M = 4
N/M = 0.5

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns

Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



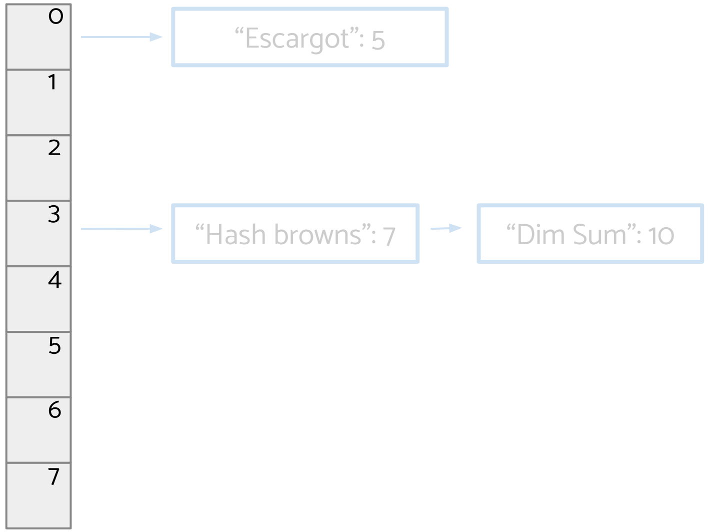
N = 3
M = 4
N/M = 0.75

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns

Draw the HashMap after the following operations.

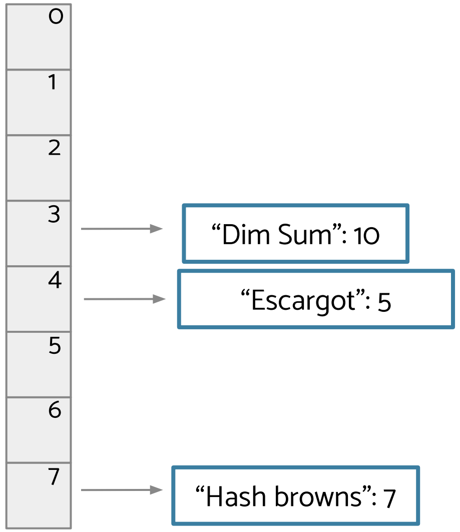
```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns
Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```

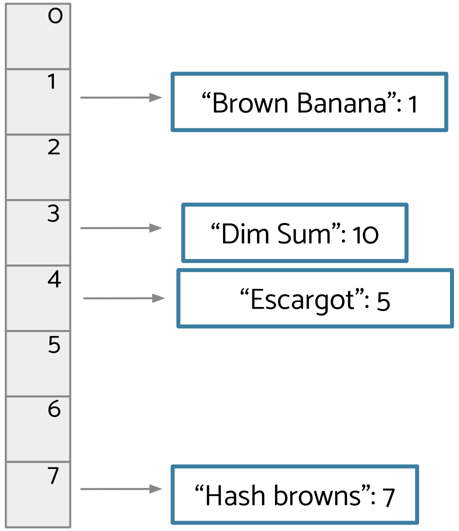


N = 3
M = 8
N/M = 0.375

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



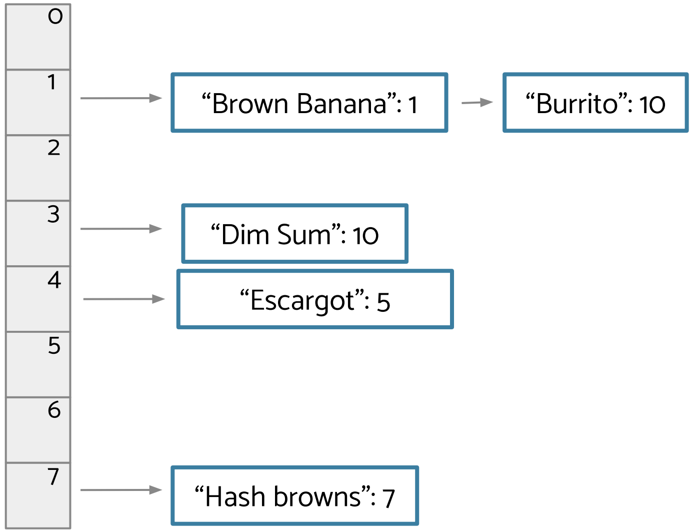
N = 4
M = 8
N/M = 0.5

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns

Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```

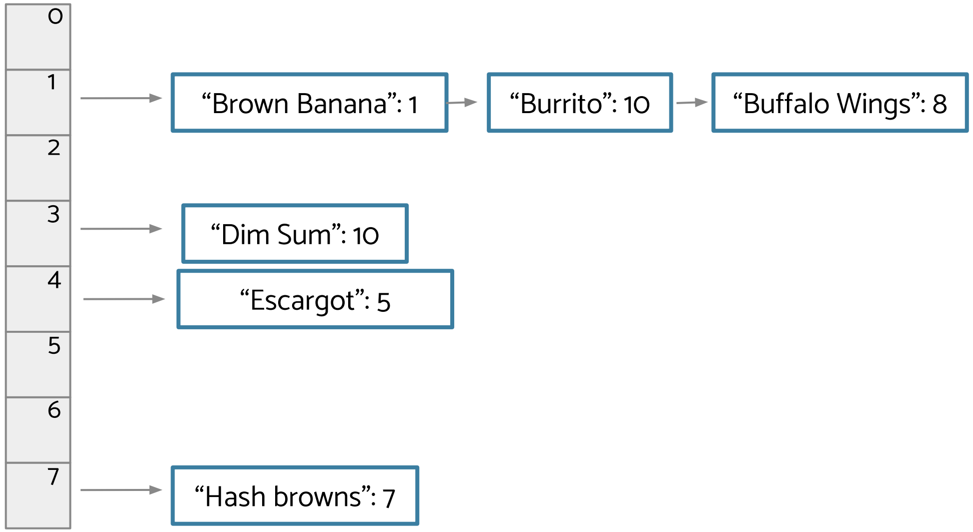


N = 5
M = 8
N/M = 0.625

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns
Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



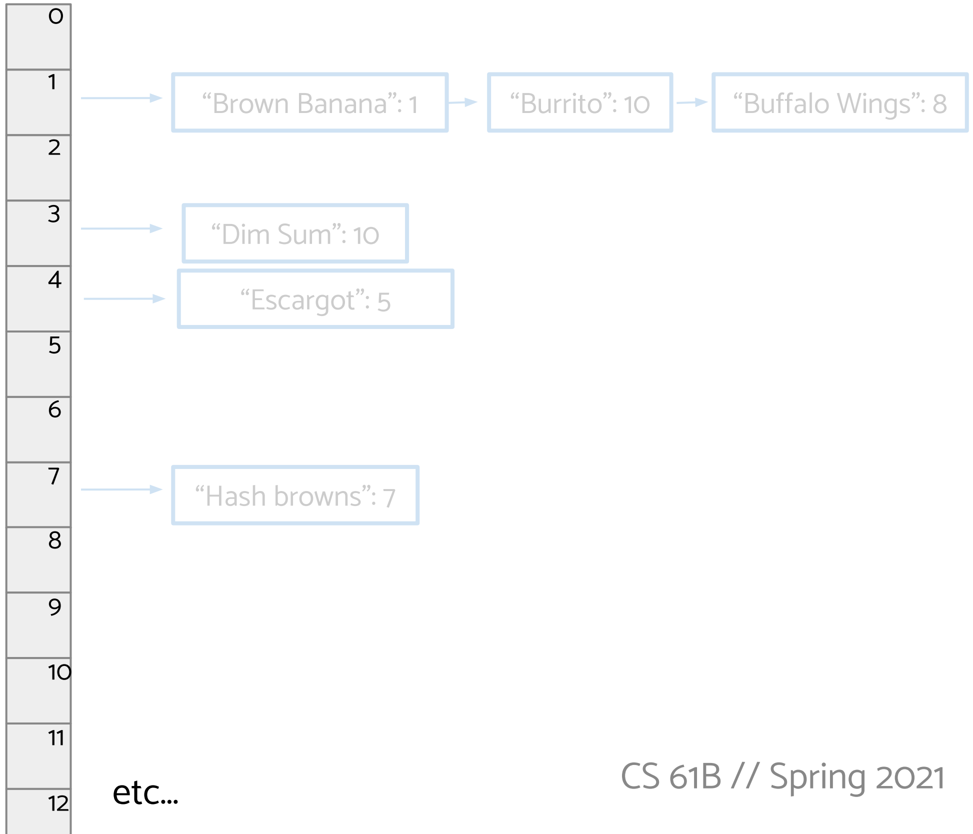
N = 6
M = 8
N/M = 0.75

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns

Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```



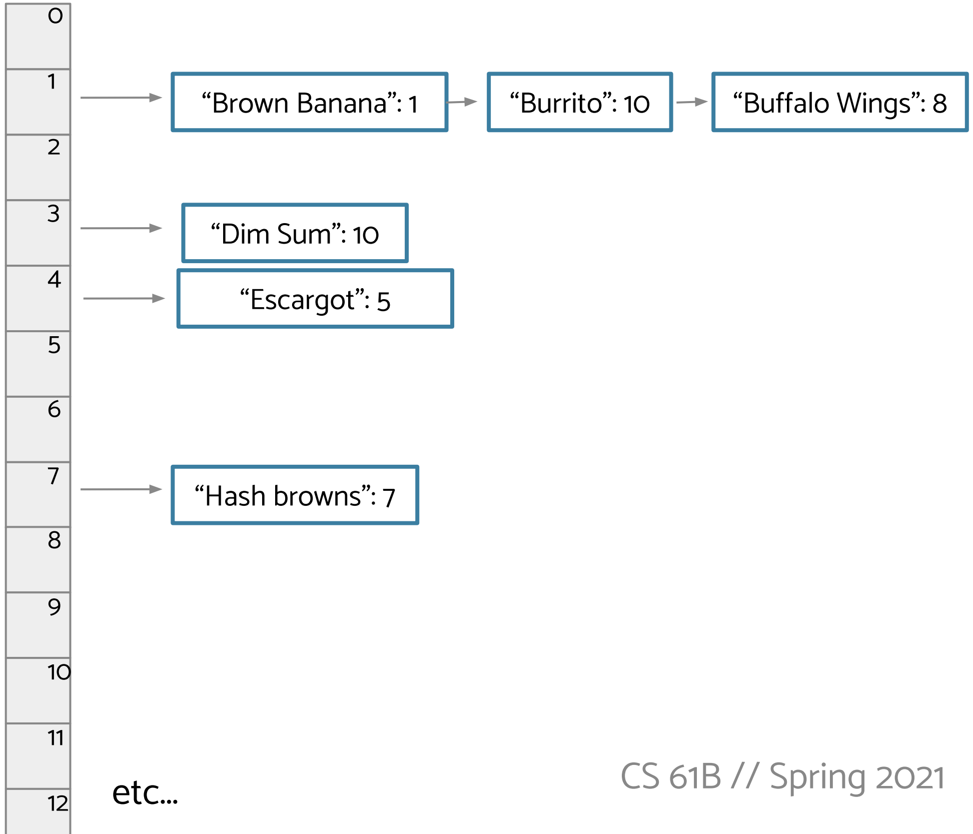
Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

3A A Side of Hash Browns Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```

N = 6
M = 16
N/M = 0.375

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7

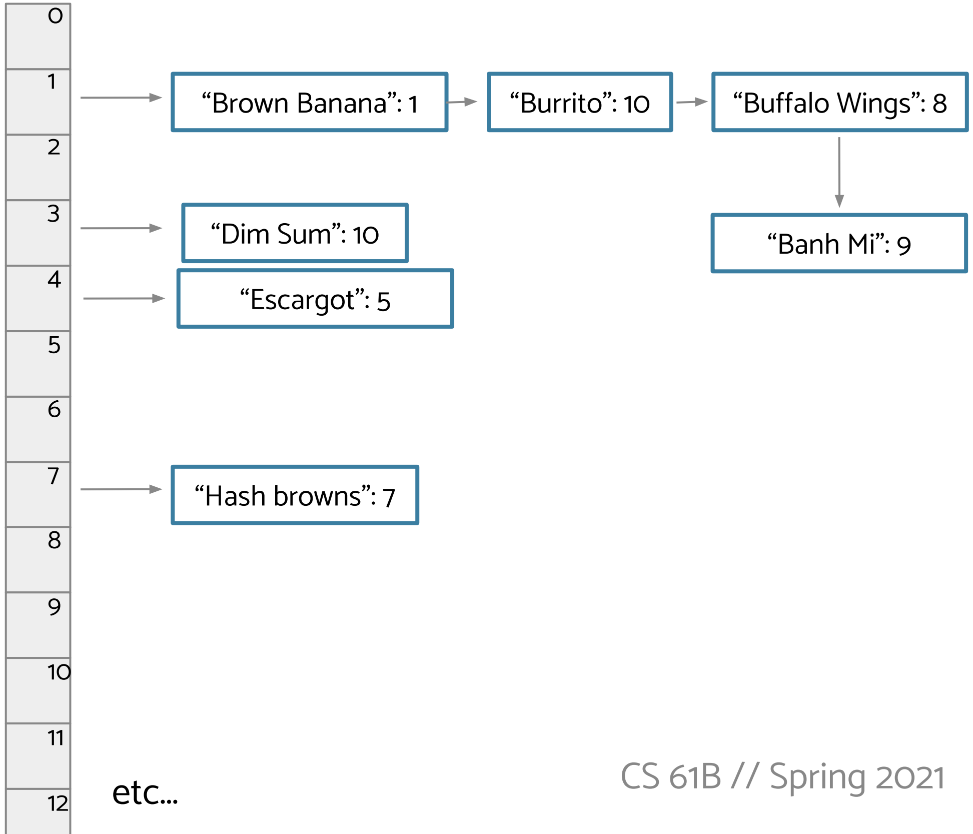


3A A Side of Hash Browns Draw the HashMap after the following operations.

```
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```

N = 7
M = 16
N/M = 0.4375

Helpful: A = 0, B = 1, D = 3, E = 4, H = 7



3B A Side of Hash Browns

Do you see a potential problem here with the behavior of our HashMap?

How could we solve this?