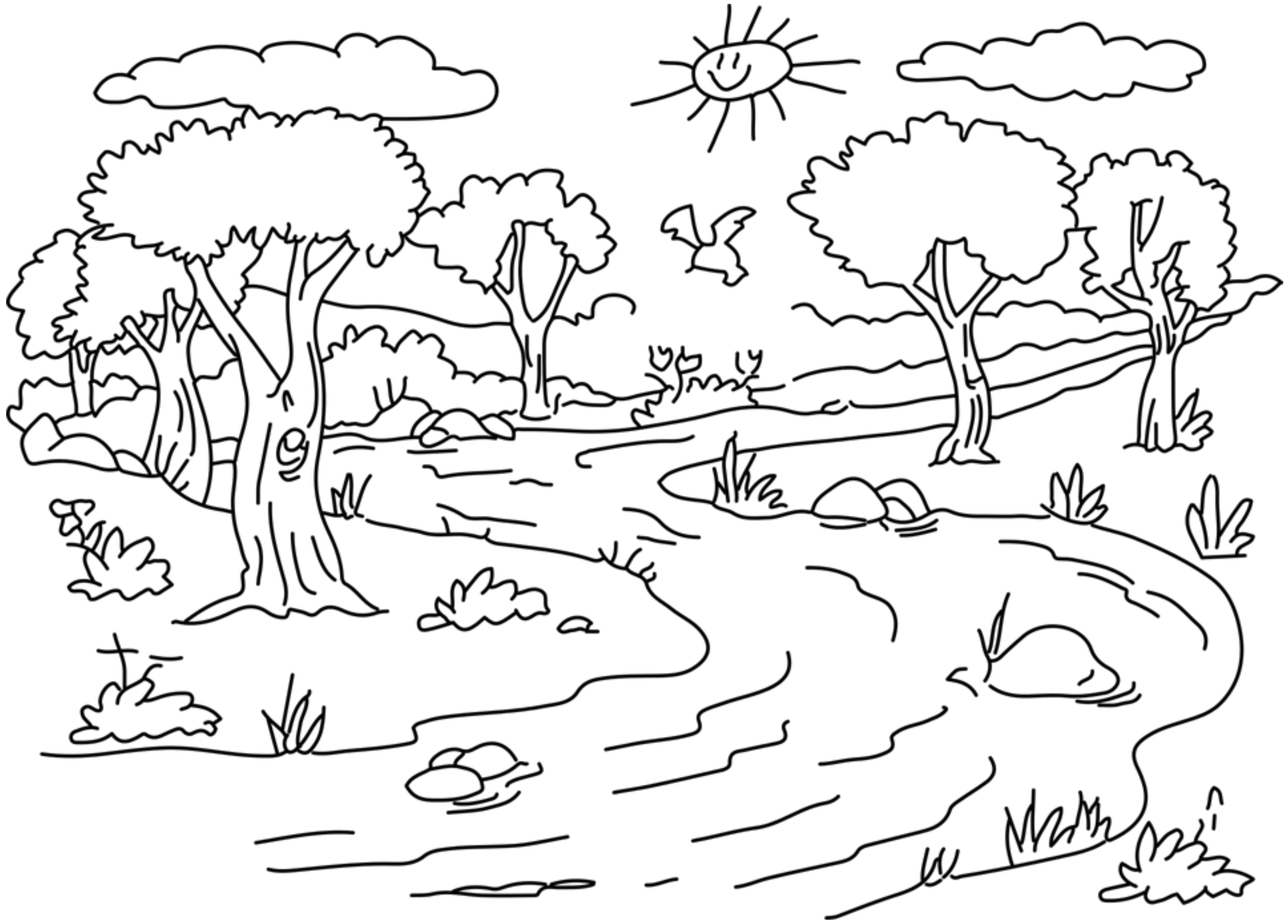


Section starts at Berkeley time...

Till then, let's color using the Zoom annotate feature!

Asymptotics & BSTs

Discussion 07

Announcements

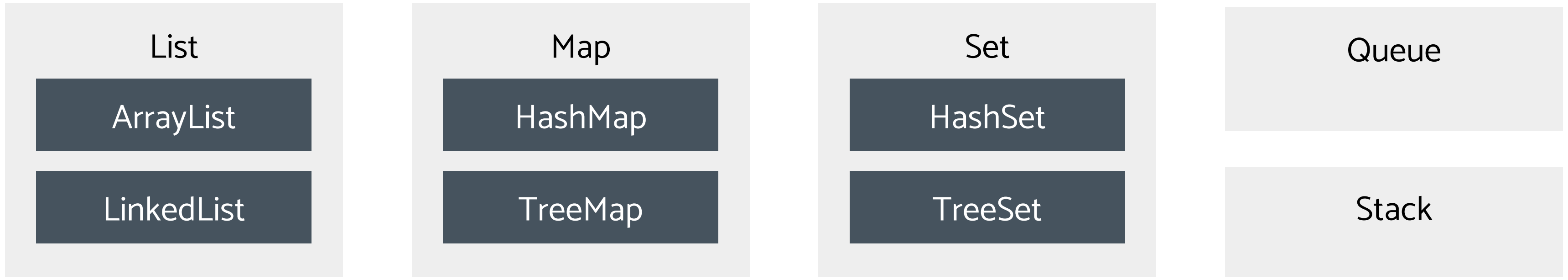
Week 7

- ❑ Weekly Survey 6 - due on Monday 3/1
- ❑ Lab 6 - due on Monday 3/1
- ❑ Lab 7 - due on Friday 3/5
- ❑ Project 2 Checkpoint - due on Friday 3/12
- ❑ HW 2 - due on Monday 3/15
- ❑ Seriously start Proj2

Content Review

Abstract Data Types

Abstract Data Types are data structures where we know *what* they do but not *how*. They are usually represented as interfaces or abstract classes in Java.

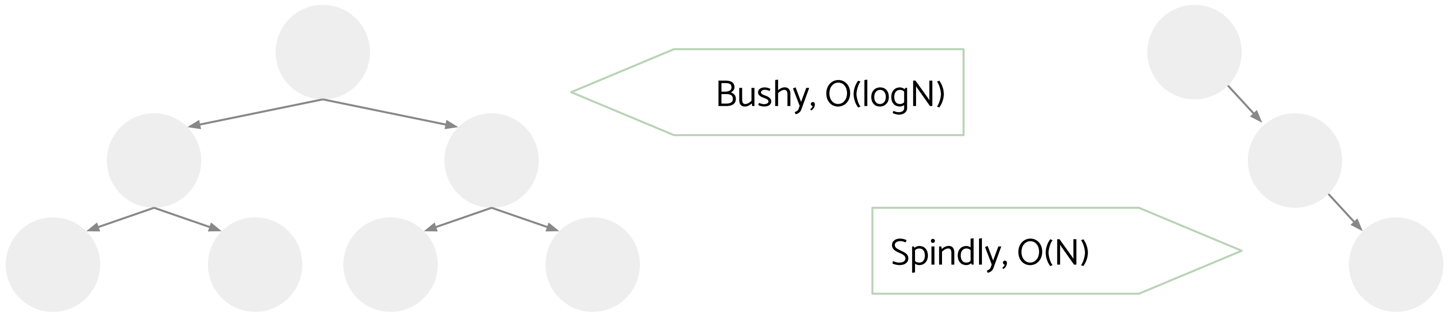


Binary Search Trees

Binary Search Trees are data structures that allow us to quickly access elements in sorted order. They have several important properties:

1. Each node in a BST is a root of a smaller BST
2. Every node to the left of a root has a value “lesser than” that of the root
3. Every node to the right of a root has a value “greater than” that of the root

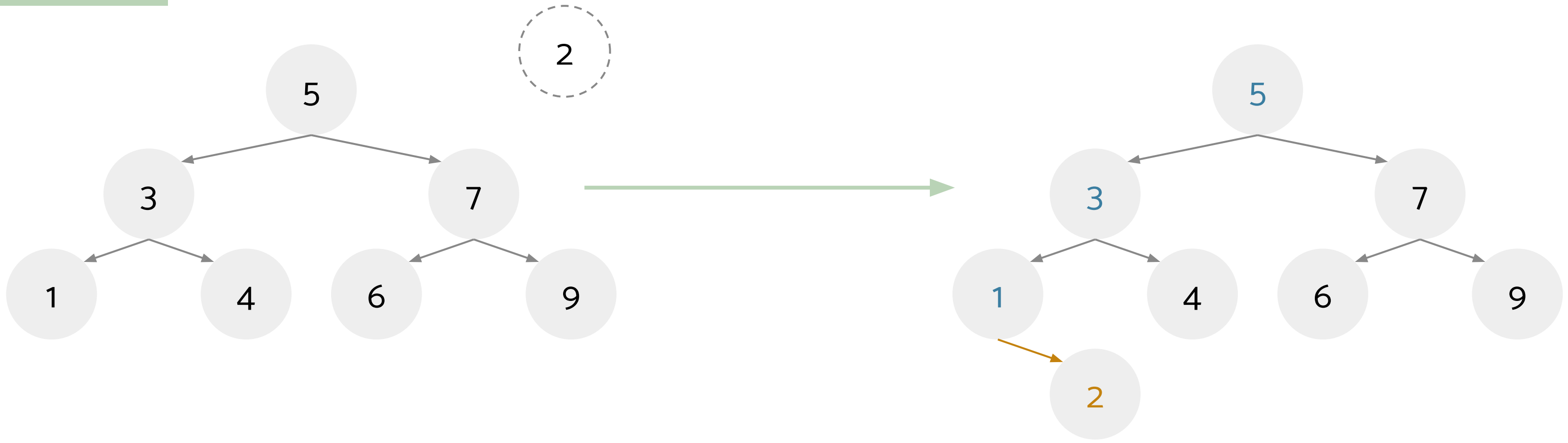
BSTs can be bushy or spindly:



BST Insertion

Items in a BST are always inserted as leaves.

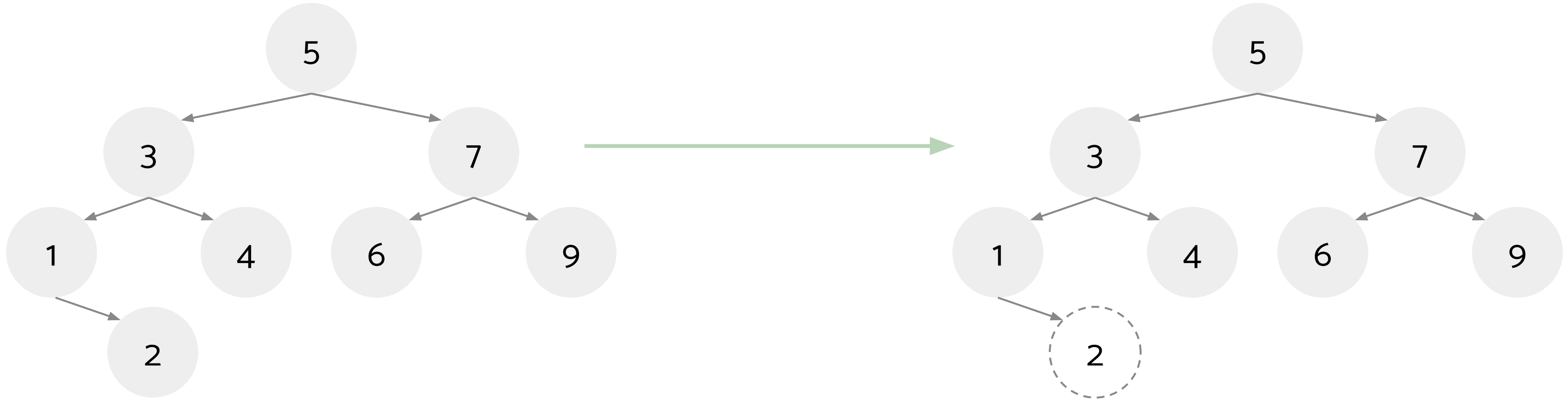
`insert(2)`



BST Deletion

Items in a BST are always deleted via a method called **Hibbard Deletion**. There are several cases to consider:

```
delete(2)
```

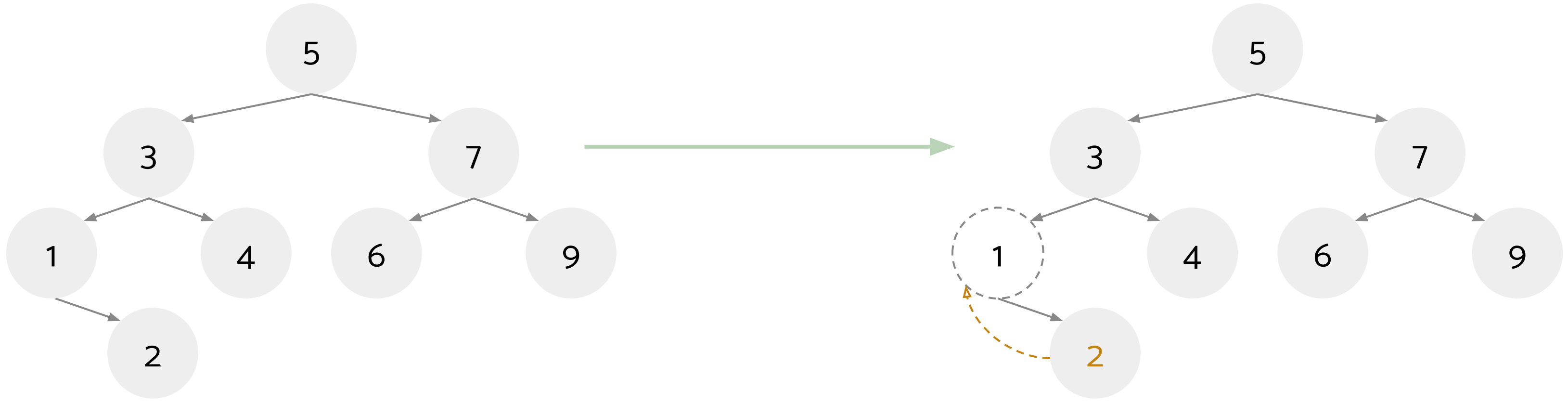


In this case, the node has no children so deletion is an easy process.

BST Deletion

Items in a BST are always deleted via a method called **Hibbard Deletion**. There are several cases to consider:

```
delete(1)
```

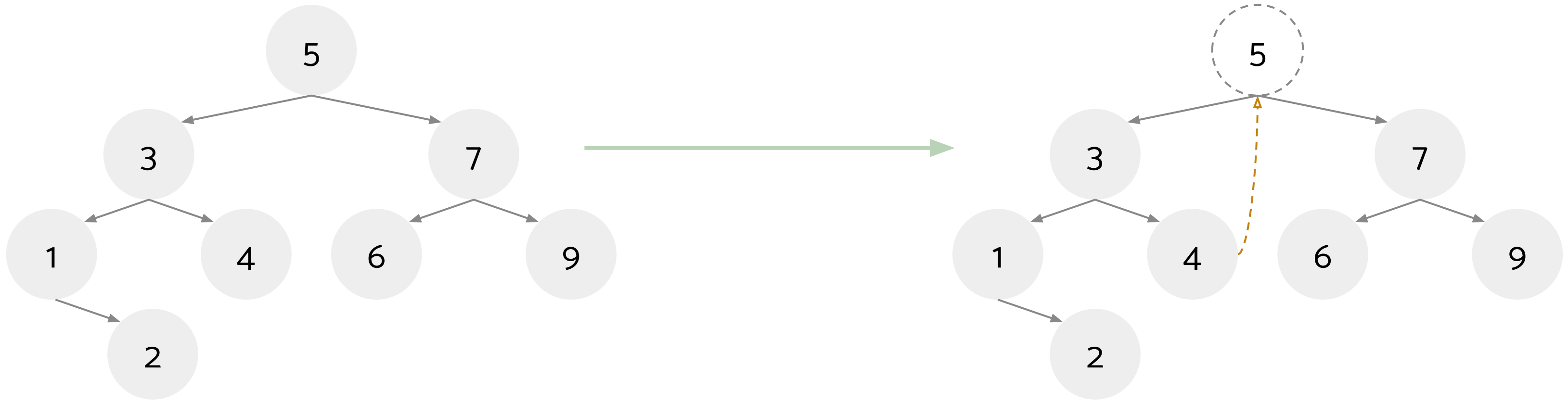


In this case, the node has one child, so it simply replaces the deleted node, and then we act as if the child was deleted in a recursive pattern until we hit a leaf.

BST Deletion

Items in a BST are always deleted via a method called **Hibbard Deletion**. There are several cases to consider:

`delete(5)`

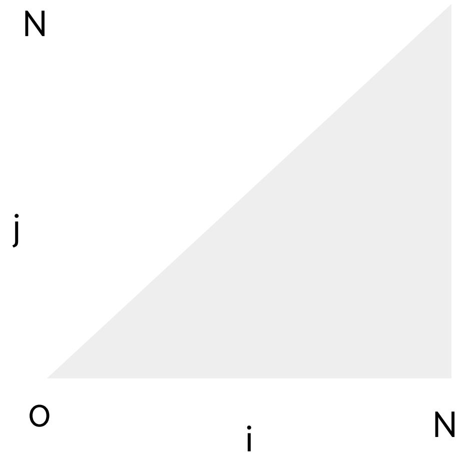


In this case, the node has two children, so we pick either the leftmost node on in the right subtree or the rightmost node in the left subtree.

Asymptotics - Important Points

1. Asymptotic analysis is only valid on very large inputs, and comparisons between runtimes is only useful when comparing inputs of different orders of magnitude.
2. While common themes are helpful, rules like “trees are always 2^N” and “nested for loops are always N^2” can easily lead you astray
3. Doing problems graphically can be helpful if you’re a visual learner:

```
for (int i = 0; i < N; i++) {
     for (int j = 0; j < i; j++) {
          /* Something constant */
     }
}
```



$\frac{1}{2} N^2 = N^2$

Worksheet

1 ADT Matchmaking Match each task to the correct Abstract Data Type for the job.

1) You want to keep track of all the unique users who have logged on to your system.
2) You are creating a version control system and want to associate each file name with a Blob.
3) We are running a server and want to service clients in the order they arrive.
4) We have a lot of books at our library and we want our website to display them in some sorted order. We have multiple copies of some books and we want each listing to be separate.

a) List

b) Map

c) Set

d) Queue

2 I Am Speed

Give the worst case and best case running time in Θ(·) notation in terms of M and N. Assume that `comeOn()` is in Θ(1) and returns a boolean.

```
for (int i = 0; i < N; i += 1) {
     for (int j = 1; j < M; ) {
          if (comeOn()) {
               j += 1;
          } else {
               j *= 2;
          }
     }
}
```

3a Re-cursed with asymptotics

What is the runtime in terms of n?

```
public static int curse(int n) {
    if (n <= 0) {
        return 0;
    } else {
        return n + curse(n - 1);
    }
}
```

3b Re-cursed with asymptotics

What is the runtime in terms of N, the number of nodes in the tree?

```
public static BST find(BST T, Key sk) {
   if (T == null)
      return null;
   if (sk.compareTo(T.key) == 0)
      return T;
   else if (sk.compareTo(T.key) < 0)
      return find(T.left, sk);
   else
      return find(T.right, sk);
}
```

3c Re-cursed with asymptotics

Can you find a runtime bound for the code below? We can assume the System.arraycopy method takes Θ(N) time, where N is the size of the input array.

```
public static void silly(int[] arr) {
    if (arr.length <= 1) {
        System.out.println("You won!");
        return;
    }
    int newLen = arr.length / 2
    int[] firstHalf = new int[newLen];
    int[] secondHalf = new int[newLen];
    System.arraycopy(arr, 0, firstHalf, 0, newLen);
    System.arraycopy(arr, newLen, secondHalf, 0, newLen);
    silly(firstHalf);
    silly(secondHalf);
}
```

4 Have You Ever Went Fast? *Extra*

```
public static boolean findSum(int[] A, int x) {
      for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < A.length; j++) {
                  if (A[i] + A[j] == x) {
                        return true;
                  }
            }
      }
      return false;
}
```

```
public static boolean findSum(int[] A, int x) { // Faster












}
```