ADTs, Asymptotics II, BSTs

Discussion 06

CS61B Spring 2024

Announcements

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
	2/26 Lab 5 Due Homework 2 Due				3/1 Lab 6 Due	
			3/6 Project 2A Due		3/8 Lab 7 Due	

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.

List	Map	Set	Queue
ArrayList	HashMap	HashSet	
LinkedList	TreeMap	TreeSet	Stack

List
- Ordered collection
- Allows duplicates

Map
- Associates a key with a value
- No duplicate keys

Set
- Unordered collection
- No duplicates

Queue / Stack
- Queue = FIFO
- Stack = LIFO

Asymptotics Advice

- Asymptotic analysis is only valid on very large inputs, and comparisons between runtimes is only useful when comparing inputs of different orders of magnitude.
- Use Θ where you can, but won't always have tight bound (usually default to O)
- **Reminder: total work done = sum of all work per iteration or recursive call**
- While common themes are helpful, rules like "nested for loops are always N^2" can easily lead you astray (pay close attention to stopping conditions and how variables update)
- Drop lower-order terms (ie. $n^3 + 10000n^2 - 5000000 \rightarrow \Theta(n^3)$)

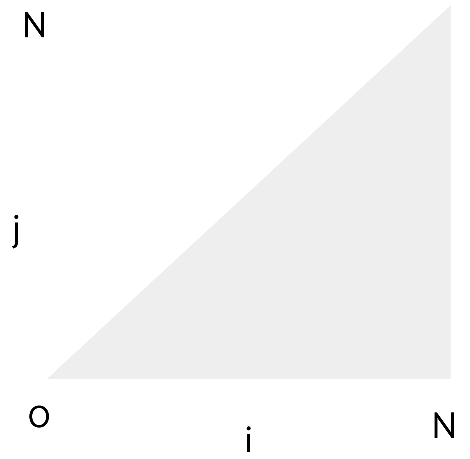
Asymptotics Advice

- For recursive problems, it's helpful to draw out the tree/structure of method calls
- Things to consider in your drawing and calculations of total work:
  - Height of tree: how many levels will it take for you to reach the base case?
  - Branching factor: how many times does the function call itself in the body of the function?
  - Work per node: how much actual work is done per function call?
- Life hack pattern matching when calculating total work where f(N) is some function of N
  - $1 + 2 + 3 + 4 + 5 + \ldots + f(N) = [f(N)]^2$
  - $1 + 2 + 4 + 8 + 16 + \ldots + f(N) = f(N)$
    - Rule applies with any geometric factor between terms, like $1 + 3 + 9 + \ldots + f(N)$

Asymptotics Advice

- Doing problems graphically can be helpful if you're a visual learner (plot variable values and calculate area formula):

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



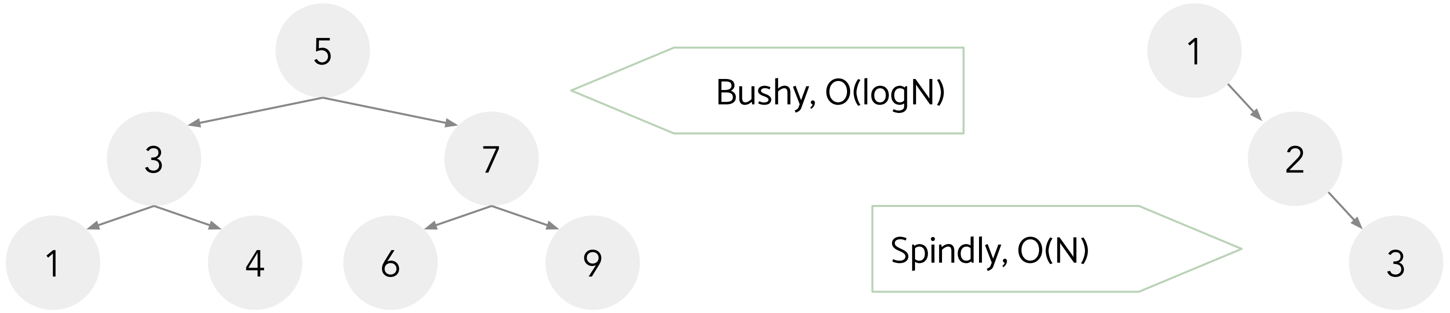
$\frac{1}{2} N^2 = N^2$

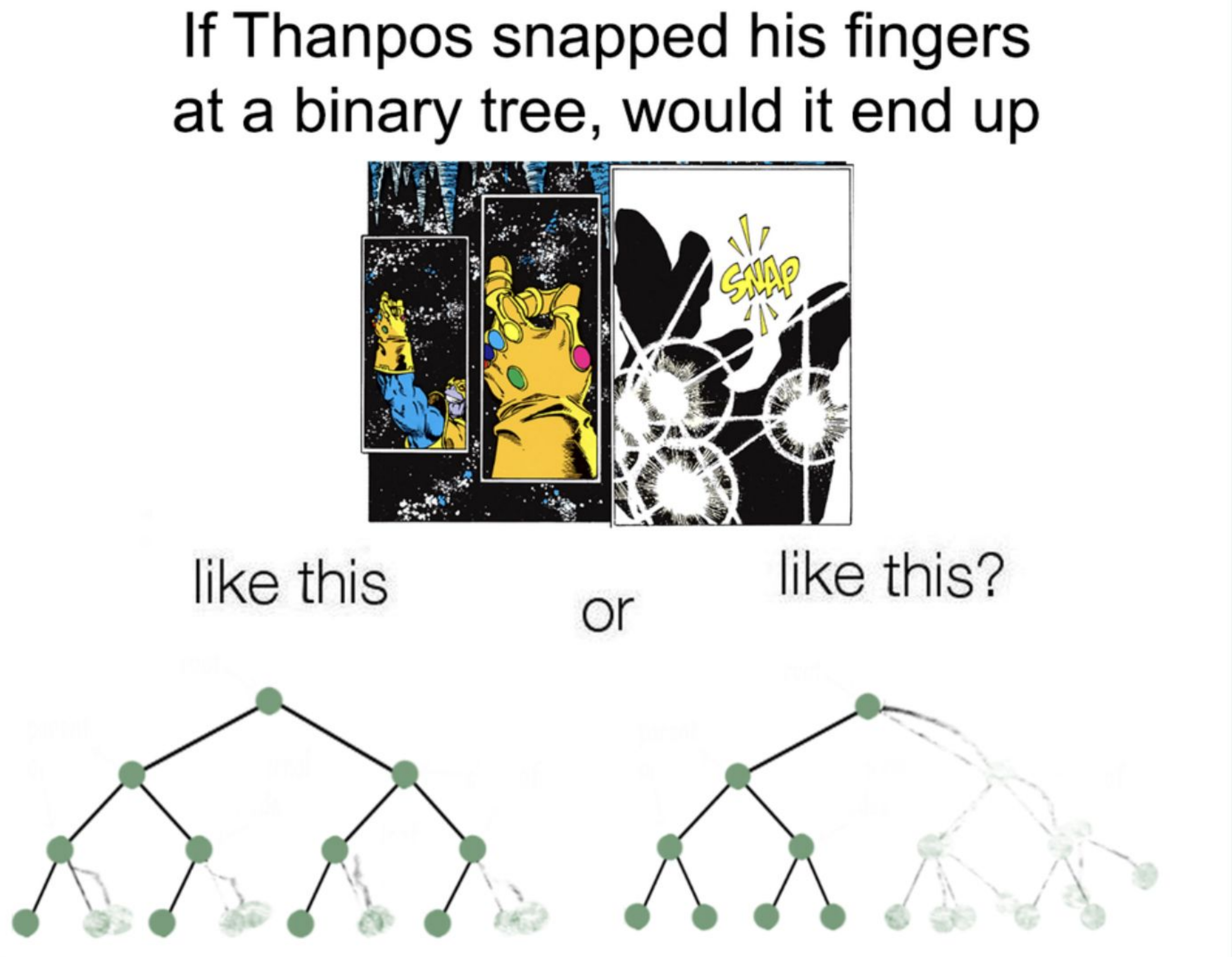
Binary Search Trees

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:

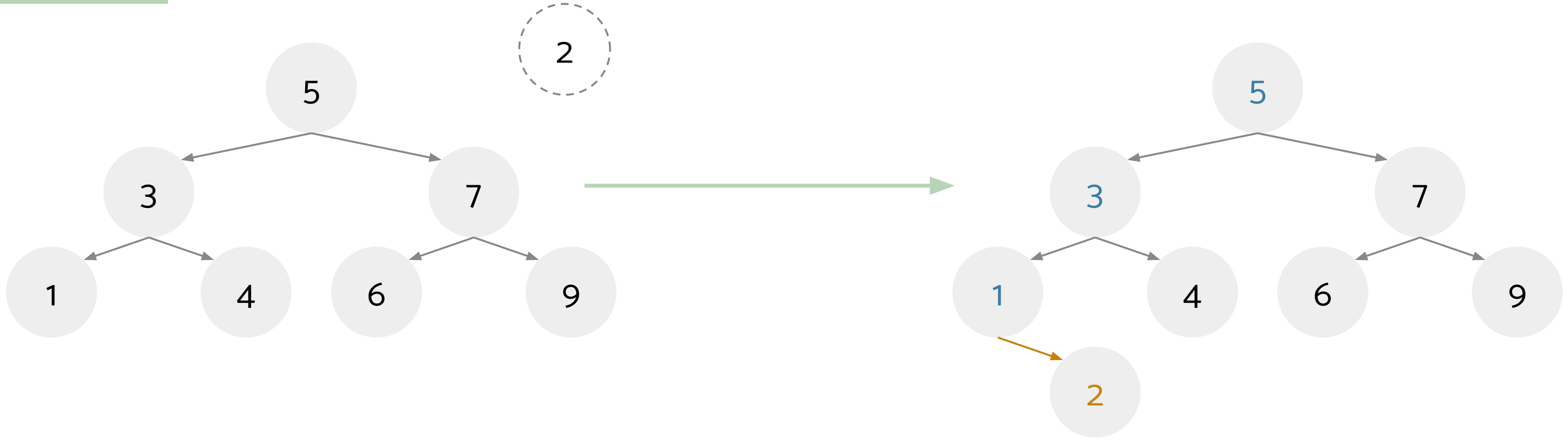



If Thanpos snapped his fingers at a binary tree, would it end up
SNAP
like this
or
like this?

BST Insertion

Items in a BST are always inserted as leaves.

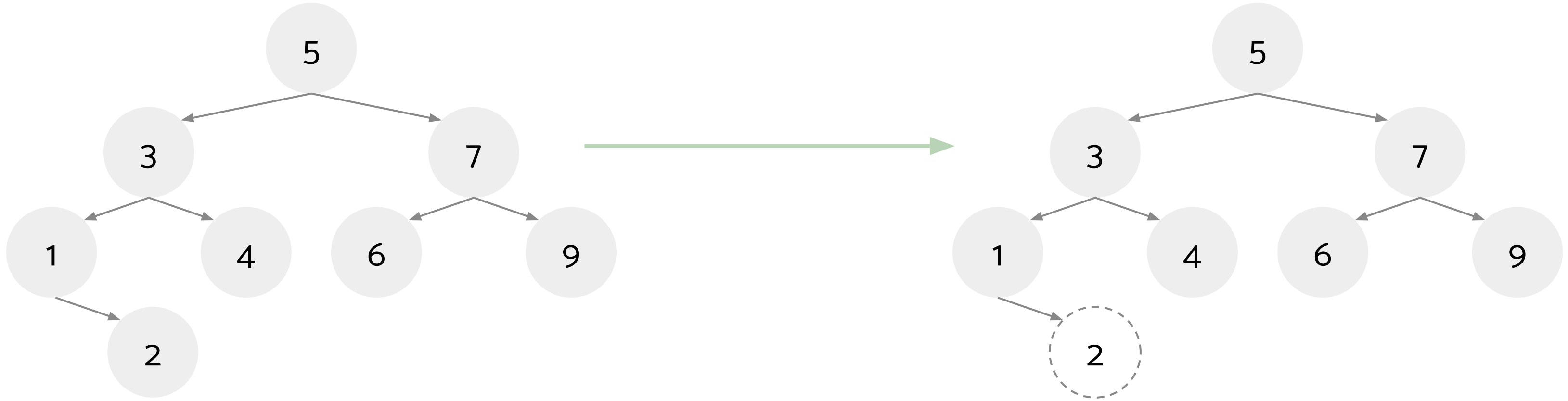
`insert(2)`



BST Deletion

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

`delete(2)`

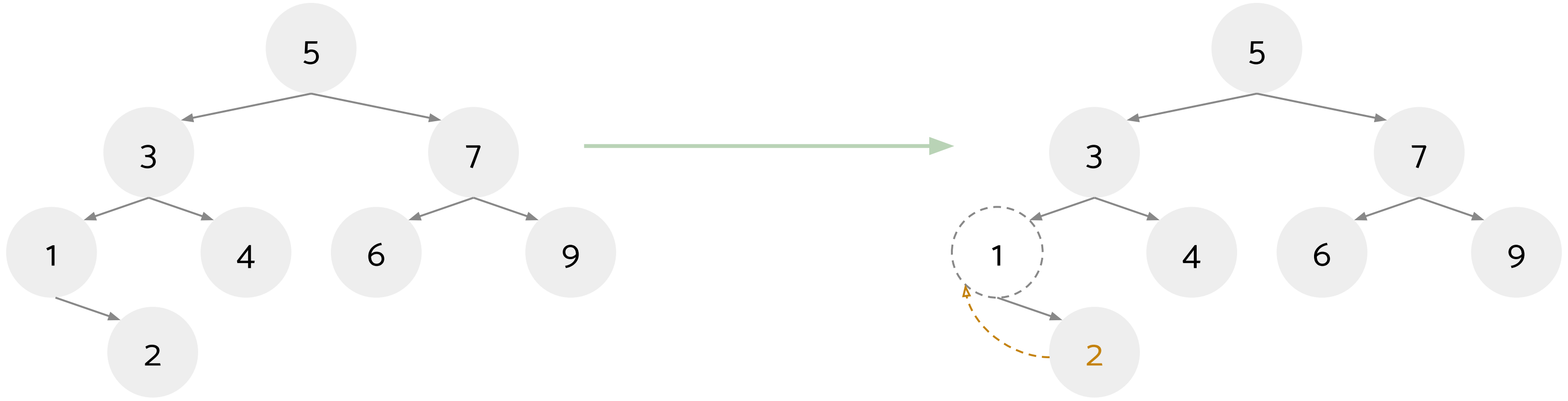


In this case, the node has no children so deletion is an easy process.

BST Deletion

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

`delete(1)`

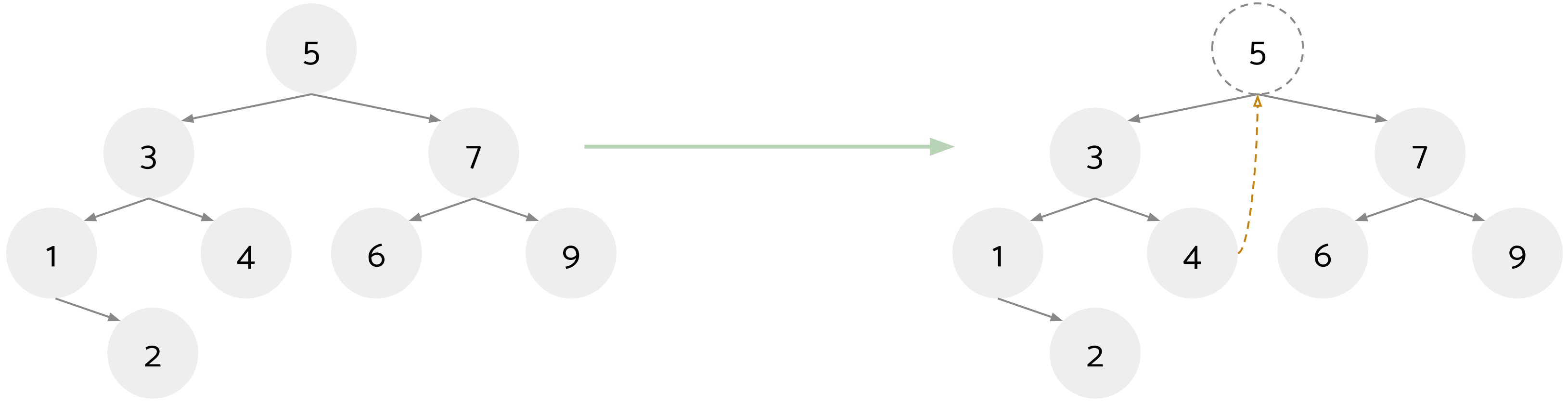


In this case, the node has one child, so it simply replaces the deleted node, and then we act as if the child was deleted in a recursive pattern until we hit a leaf.

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 in the right subtree or the rightmost node in the left subtree.

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 grading a pile of exams and want to grade starting from the top of the pile.

4) We are running a server and want to service clients in the order they arrive.

5) 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.

Options:
- List
- Map
- Queue
- Set
- Stack

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 grading a pile of exams and want to grade starting from the top of the pile.

4) We are running a server and want to service clients in the order they arrive.

5) 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.

Set

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 grading a pile of exams and want to grade starting from the top of the pile.

4) We are running a server and want to service clients in the order they arrive.

5) 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.

Set

Map

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 grading a pile of exams and want to grade starting from the top of the pile.
4) We are running a server and want to service clients in the order they arrive.
5) 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.

Set

Map

Stack

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 grading a pile of exams and want to grade starting from the top of the pile.
4) We are running a server and want to service clients in the order they arrive.
5) 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.

Set

Map

Stack

Queue

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.	Set
2) You are creating a version control system and want to associate each file name with a Blob.	Map
3) We are grading a pile of exams and want to grade starting from the top of the pile.	Stack
4) We are running a server and want to service clients in the order they arrive.	Queue
5) 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.	List

2A I Am Speed **Fill in the blank(s) so that the function has the desired runtime. .**

```
// Desired Runtime: Θ(N)

public static void f1(int N) {

    for (int i = 1; i < N; ____________){

        System.out.println("hi Dom");

    }

}
```

i	1	?	?	…	< N
work per i	1	1	1	…	1

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(N)

public static void f1(int N) {

    for (int i = 1; i < N; i += 1){

        System.out.println("hi Dom");

    }

}
```

i	1	2	3	…	N - 1
work per i	1	1	1	…	1

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(logN)

public static void f2(int N) {

    for (int i = 1; i < N; ____________) {

        System.out.println("howdy Ergun");

    }

}
```

i	1	?	?	...	< N
work per i	1	1	1	...	1

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(logN)
public static void f2(int N) {
    for (int i = 1; i < N; i *= 2) {
        System.out.println("howdy Ergun");
    }
}
```

i	1	2	4	…	N/2
work per i	1	1	1	…	1

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(1)

public static void f3(int N) {

    for (int i = 1; ____________; i += 1) {

        System.out.println("hello Anniyat");

    }

}
```

i	1	2	3	…	?
work per i	1	1	1	…	1

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(1)

public static void f3(int N) {

    for (int i = 1; i < 1000; i += 1) {

        System.out.println("hello Anniyat");

    }

}
```

* Note that the solution is actually just i < C, where C is some constant independent of the input N.

i	1	2	3	…	999
work per i	1	1	1	…	1

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(2^N)
// This one is tricky!
// Hint: think about the sum for 1 + 2 + 4 + ... + f(N)
public static void f4(int N) {
    for (int i = 1; ____________; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            System.out.println("what's up Alyssa");
        }
    }
}
```

i	1	2	4	...	?
work per i	1	2	4	...	< i

2A I Am Speed Fill in the blank(s) so that the function has the desired runtime. .

```
// Desired Runtime: Θ(2^N)
// This one is tricky!
// Hint: think about the sum for 1 + 2 + 4 + ... + f(N)
public static void f4(int N) {
    for (int i = 1; i < Math.pow(2, N); i *= 2) {
        for (int j = 0; j < i; j += 1) {
            System.out.println("what's up Alyssa");
        }
    }
}
```

i	1	2	4	…	$2^N - 1$
work per i	1	2	4	…	$2^N - 1$

2B I Am Speed (Extra) Give the worst case and best case running time in Θ(·) notation in terms of M and N. Assume that `kachow()` is in $\Theta(N^2)$ and returns a boolean.

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

2B I Am Speed (Extra) Give the worst case and best case running time in Θ(·) notation in terms of M and N. Assume that `kachow()` is in $\Theta(N^2)$ and returns a boolean.

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

Best: $\Theta(N^3 \log M)$

Worst: $\Theta(N^3 M)$

i	0	1	…	N - 1
Best case work per i	$N^2\log(M)$	$N^2\log(M)$	…	$N^2\log(M)$
Worst case work per i	$N^2(M - 1)$	$N^2(M - 1)$	…	$N^2(M - 1)$

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);
    }
}
```

3a Re-cursed with asymptotics What is the runtime in terms of n?

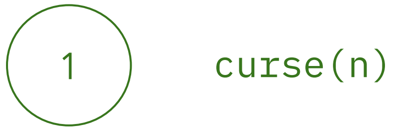
```
public static int curse(int n) {
    if (n <= 0) {
        return 0;
    } else {
        return n + curse(n - 1);
    }
}
```

- height of tree: N
- branching factor: 1
- work per node: 1

3a Re-cursed with asymptotics

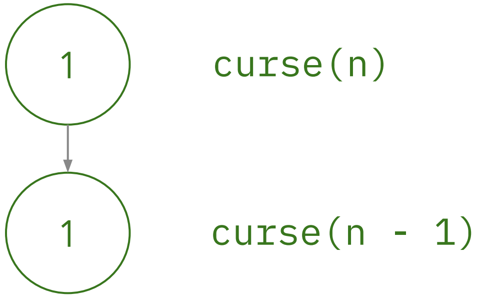
What is the runtime in terms of n?

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



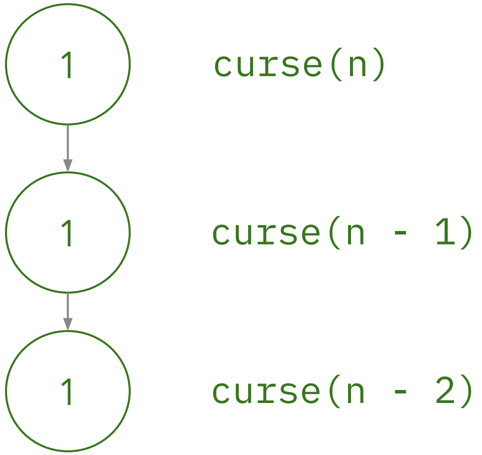
3a Re-cursed with asymptotics What is the runtime in terms of n?

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



3a Re-cursed with asymptotics What is the runtime in terms of n?

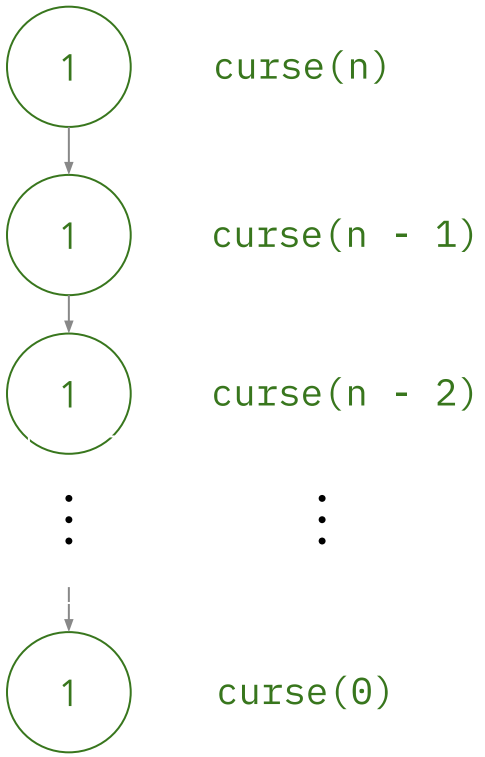
```
public static int curse(int n) {
    if (n <= 0) {
        return 0;
    } else {
        return n + curse(n - 1);
    }
}
```



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);
    }
}
```

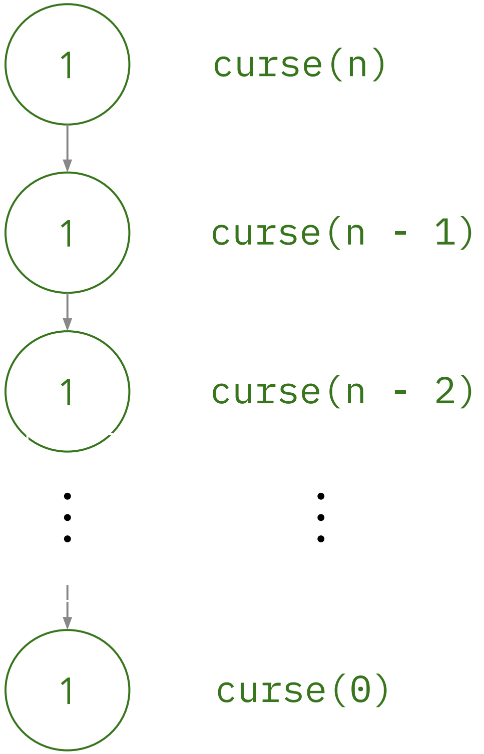


3a Re-cursed with asymptotics What is the runtime in terms of n?

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

Runtime: Θ(N)

- 1 total work per level * N levels = N



3b Re-cursed with asymptotics

Can you find a runtime bound for the code below? 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);
}
```

3b Re-cursed with asymptotics

Can you find a runtime bound for the code below? 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);
}
```

- height of tree: logN
- branching factor: 2
- work per node: N (arr.length)

3b Re-cursed with asymptotics

Can you find a runtime bound for the code below? 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);
}
```

3b Re-cursed with asymptotics

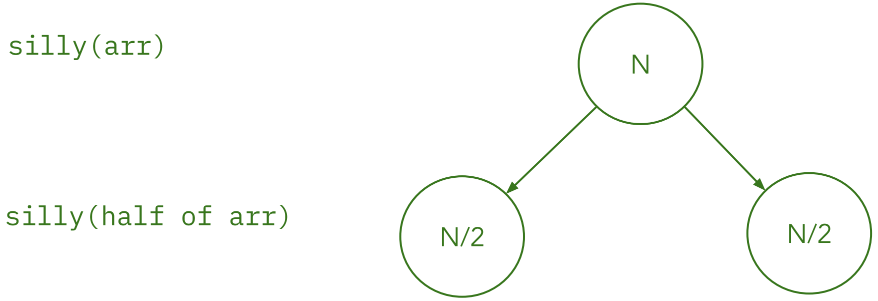
Can you find a runtime bound for the code below? Assume the System.arraycopy method takes Θ(N) time, where N is the size of the input array.

```
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);
}
```



3b Re-cursed with asymptotics

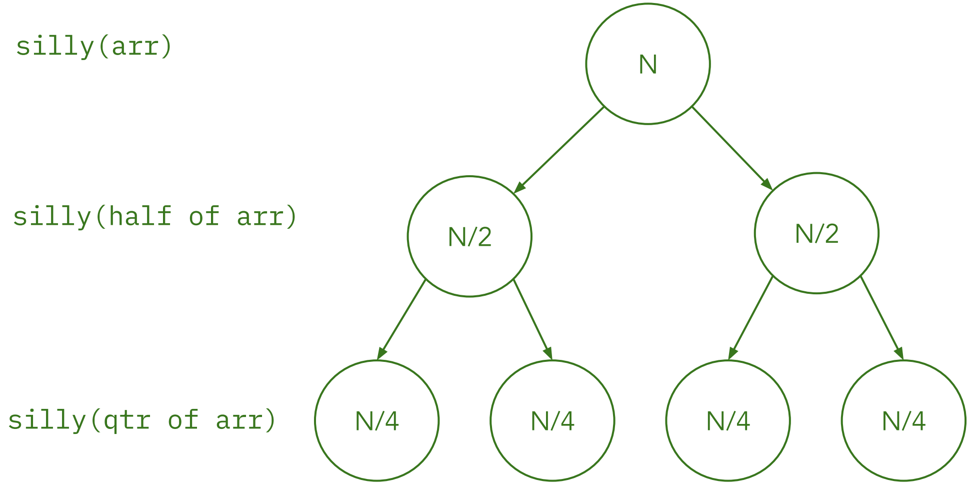
Can you find a runtime bound for the code below? Assume the System.arraycopy method takes Θ(N) time, where N is the size of the input array.

```
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);
}
```



3b Re-cursed with asymptotics

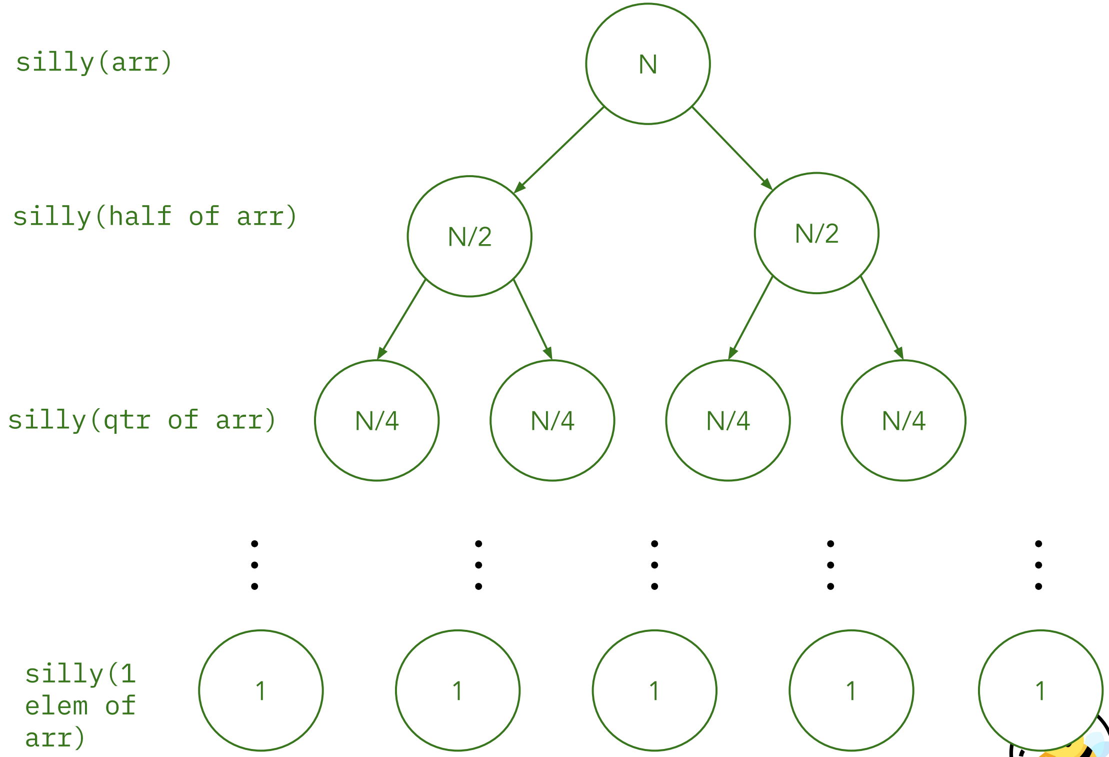
Can you find a runtime bound for the code below? Assume the System.arraycopy method takes Θ(N) time, where N is the size of the input array.

```
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);
}
```



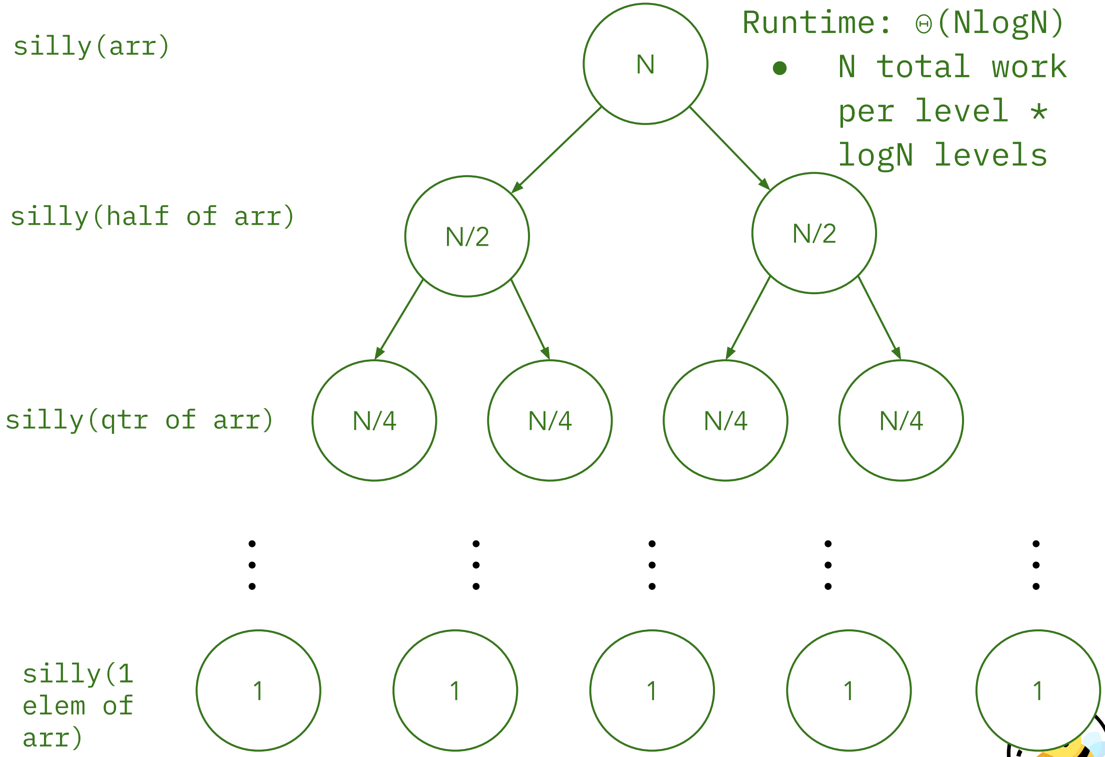
3b Re-cursed with asymptotics Can you find a runtime bound for the code below? Assume the System.arraycopy method takes Θ(N) time, where N is the size of the input array.

```
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);
}
```



Runtime: Θ(NlogN)

- N total work per level * logN levels

3c Re-cursed with asymptotics

Given that `exponentialWork` runs in $\Theta(3^N)$ time with respect to input N, what is the runtime of `ronnie`?

```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```

3c Re-cursed with asymptotics Given that `exponentialWork` runs in $\Theta(3^N)$ time with respect to input N, what is the runtime of `ronnie`?

```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```

- `height of tree: N/2`
- `branching factor: 3`
- `work per node:` 3^N

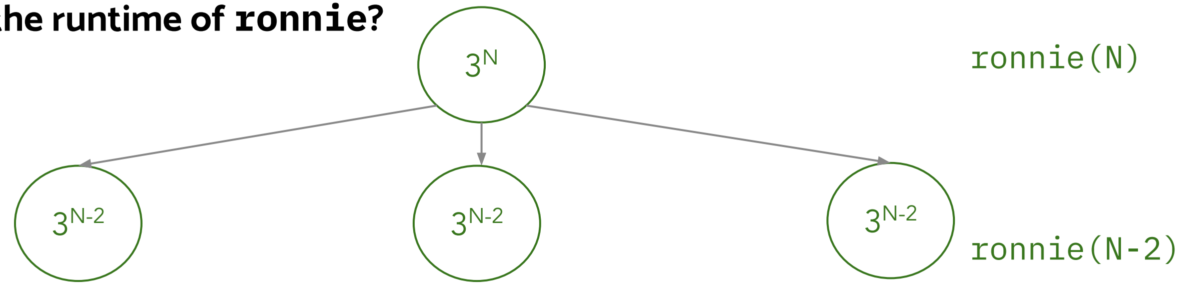
3c Re-cursed with asymptotics

Given that `exponentialWork` runs in $\Theta(3^N)$ time with respect to input N, what is the runtime of `ronnie`?

```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```

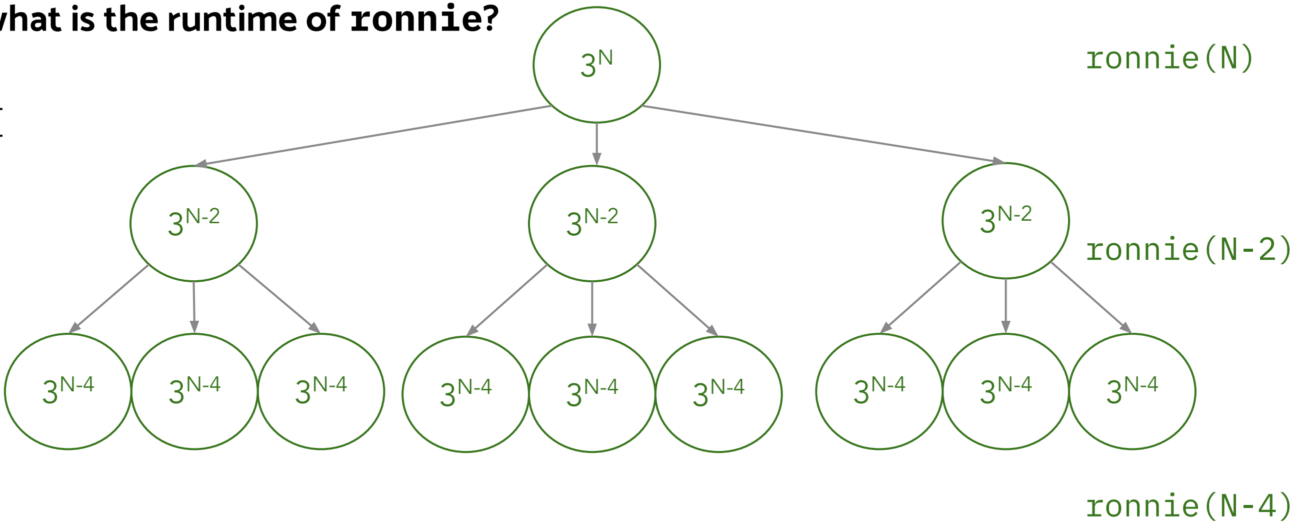
3c Re-cursed with asymptotics Given that `exponentialWork` runs in $\Theta(3^N)$ time with respect to input N, what is the runtime of `ronnie`?

```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```



3c Re-cursed with asymptotics Given that `exponentialWork` runs in Θ(3^N) time with respect to input N, what is the runtime of `ronnie`?

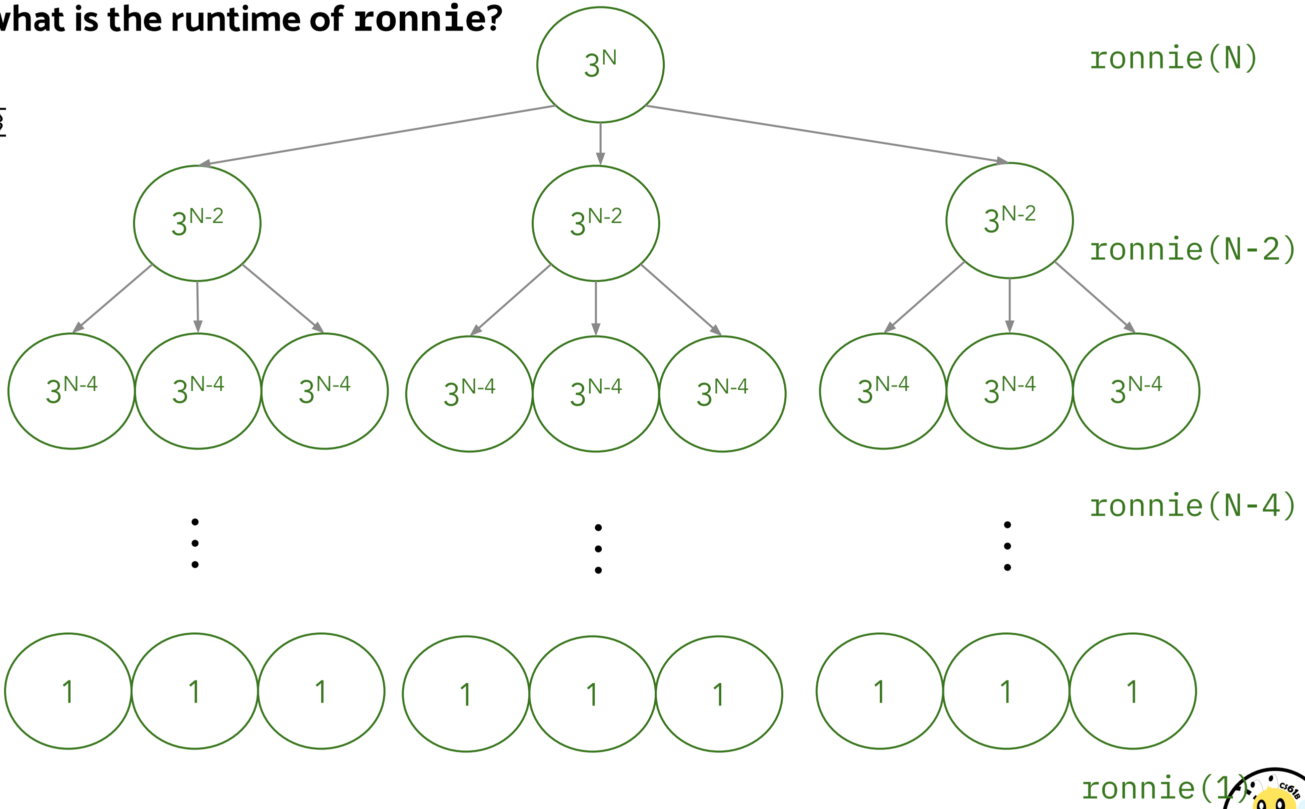
```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```



3c Re-cursed with asymptotics

Given that `exponentialWork` runs in $\Theta(3^N)$ time with respect to input N, what is the runtime of `ronnie`?

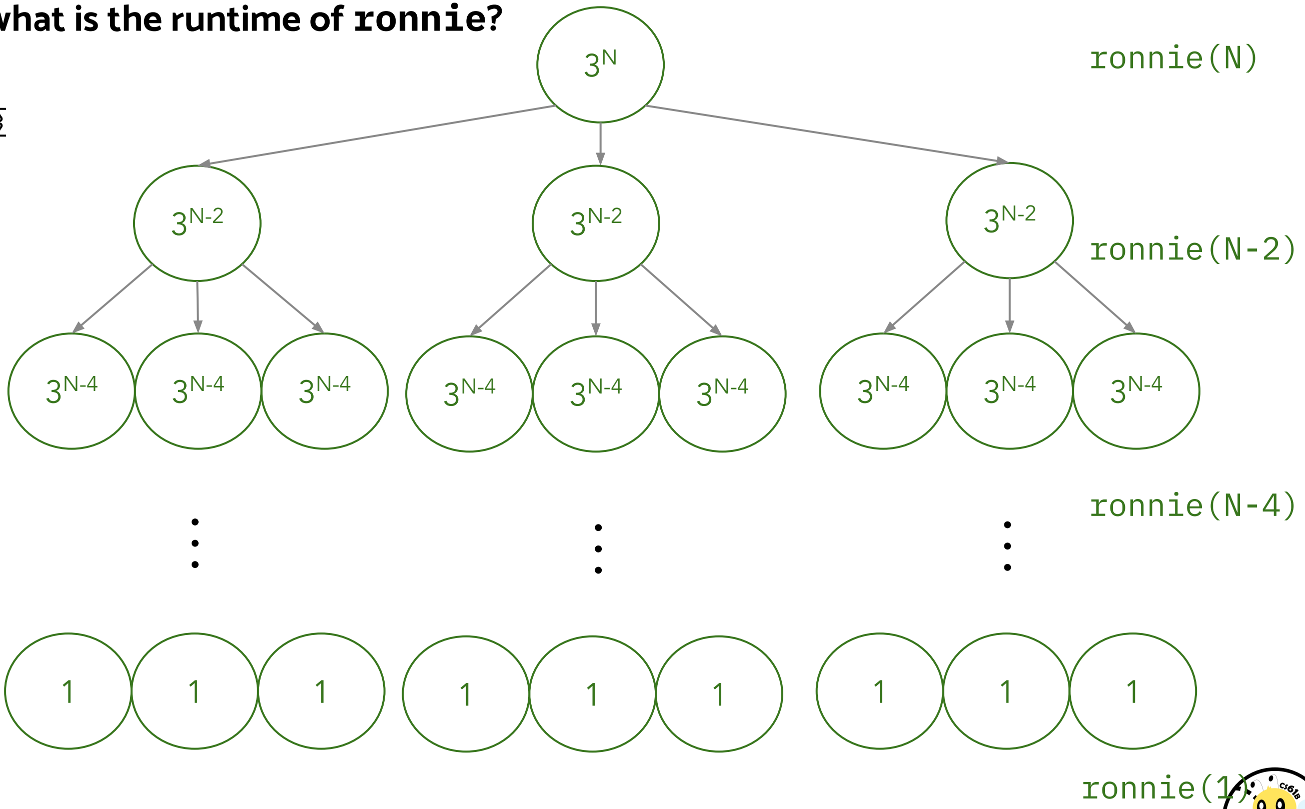
```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```



3c Re-cursed with asymptotics

Given that `exponentialWork` runs in Θ(3^N) time with respect to input N, what is the runtime of `ronnie`?

```
public void ronnie(int N) {
    if (N <= 1) {
        return;
    }
    ronnie(N - 2);
    ronnie(N - 2);
    ronnie(N - 2);
    exponentialWork(N);
}
```



Total work = $3^N + 3*3^{N-2} + 9*3^{N-4} + \ldots + 3^{N/2}*1$

$= 3^N + 3^{N-1} + 3^{N-2} + \ldots + 3^{N/2}$

Runtime: $\Theta(3^N)$

4a BST Asymptotics **What is the runtime for find on a perfectly bushy BST in terms of N, the number of nodes in the tree? Can we generalize the runtime to a tight bound?**

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

4a BST Asymptotics

What is the runtime for find on a perfectly bushy BST in terms of N, the number of nodes in the tree? Can we generalize the runtime to a tight bound?

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

`Runtime: O(logN)`

Cannot generalize to tight bound because lower and upper bound are different

- Lower: Ω(1)
  - `find` was called on the root's key
- Upper: O(logN)
  - `find` was called on a leaf's key

4b BST Asymptotics

In what order should we insert the keys [6, 2, 5, 9, 0, -3] to the BST such that the runtime of a single find operation after all keys are inserted is O(N)? Draw out the resulting BST.

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

4b BST Asymptotics

In what order should we insert the keys [6, 2, 5, 9, 0, -3] to the BST such that the runtime of a single find operation after all keys are inserted is O(N)? Draw out the resulting BST.

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

In either ascending or descending sorted order. Ascending tree: