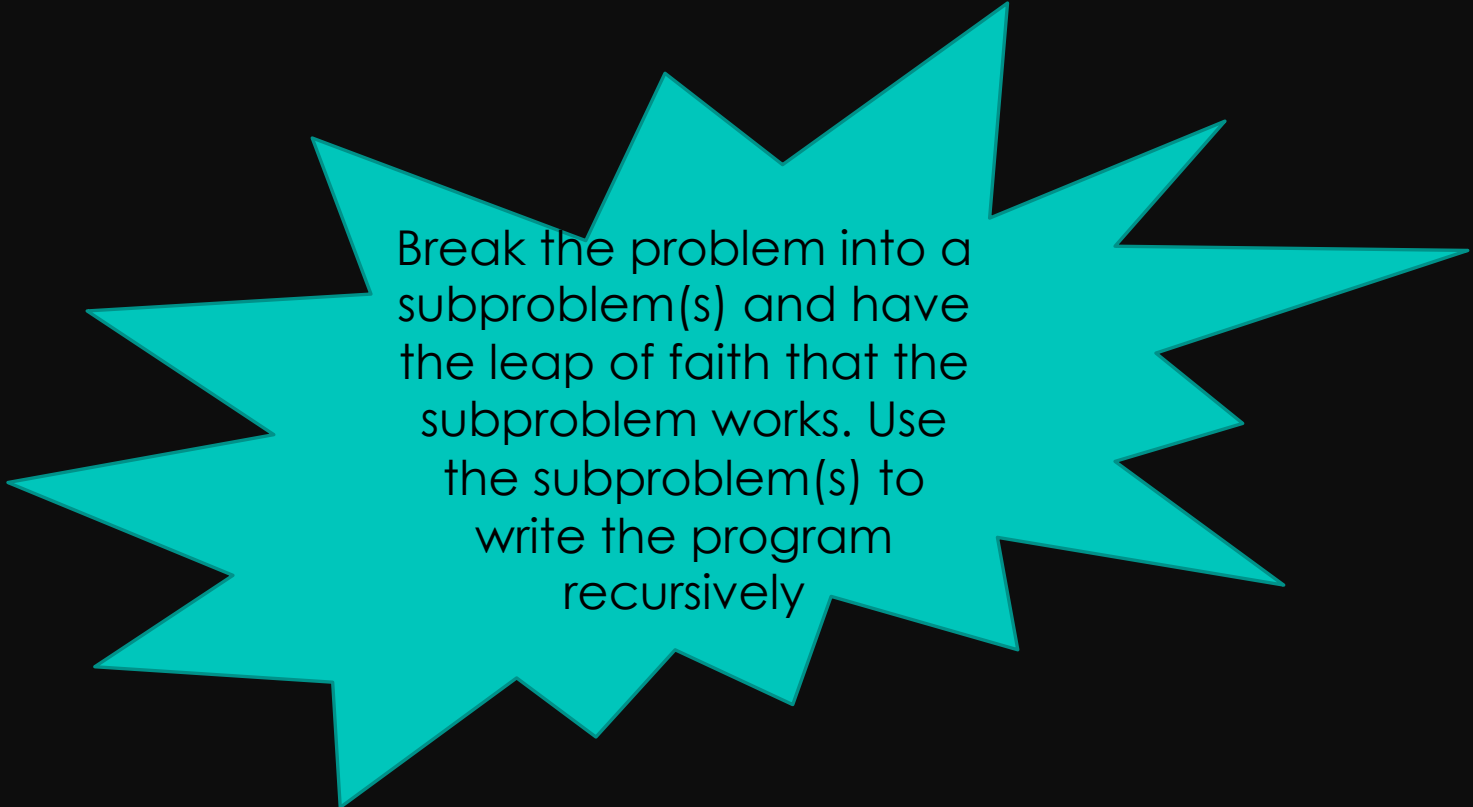


Recursion

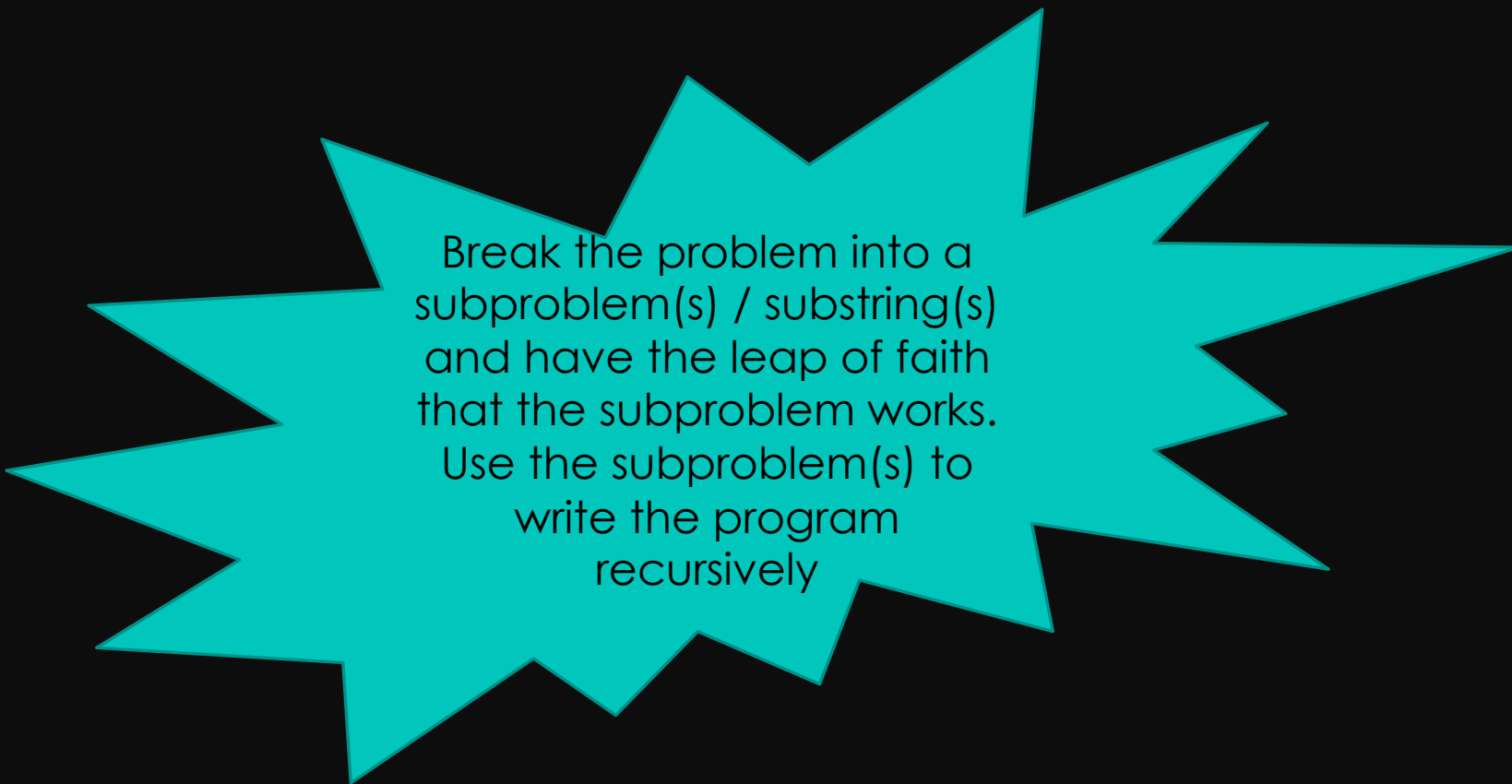
Think Recursively



Break the problem into a subproblem(s) and have the leap of faith that the subproblem works. Use the subproblem(s) to write the program recursively

Think Recursively

Reverse a String="Hello" recursively



Break the problem into a subproblem(s) / substring(s) and have the leap of faith that the subproblem works. Use the subproblem(s) to write the program recursively

Think Recursively

Reverse a String="Hello" recursively

Reverse String: "Hello"

= "o" + Reverse sub-string : ("Hell")

Have the leap of faith that
Reverse sub-string("Hell")
works

OR

Reverse String: "Hello"

= Reverse sub-string : ("ello") + "H"

Have the leap of faith that
reverse sub-string("ello") works

Think Recursively

Reverse a String="Hello" recursively

Reverse String: "Hello"

= "o" + Reverse sub-string : ("Hell")
= "o" + "l" + Reverse sub-string : ("Hel")

Have the leap of faith that
reverse sub-string("Hel")
works

OR

Reverse String: "Hello"

= Reverse sub-string : ("ello") + "H"
= Reverse sub-string : ("llo") + "e" + "H"

Have the leap of faith that
reverse sub-string("llo")
works

Think Recursively

Reverse a String="Hello" recursively

Reverse String: "Hello"

= "o" + Reverse sub-string : ("Hell")
= "o" + "l" + Reverse sub-string : ("Hel")
= "o" + "l" + "l" + Reverse sub-string : ("He")

Have the leap of faith that
Reverse sub-string("He")
works

OR

Reverse String: "Hello"

= Reverse sub-string : ("ello") + "H"
= Reverse sub-string : ("llo") + "e" + "H"
= Reverse sub-string : ("lo") + "l" + "e" + "H"

Have the leap of faith that
Reverse sub-string("lo")
works

Think Recursively

Reverse a String="Hello" recursively

Reverse String: "Hello"

= "o" + Reverse sub-string : ("Hell")
= "o" + "l" + Reverse sub-string : ("Hel")
= "o" + "l" + "l" + Reverse sub-string : ("He")
= "o" + "l" + "l" + "e" + Reverse sub-string : ("H")

Have the leap of faith that
Reverse sub-string("H")
works

OR

Reverse String: "Hello"

= Reverse sub-string : ("ello") + "H"
= Reverse sub-string : ("llo") + "e" + "H"
= Reverse sub-string : ("lo") + "l" + "e" + "H"
= Reverse sub-string : ("o") + "l" + "l" + "e" + "H"

Have the leap of faith that
Reverse sub-string("o") works

Think Recursively

Reverse a String="Hello" recursively

Reverse String: "Hello"

= "o" + Reverse sub-string : ("Hell")
= "o" + "l" + Reverse sub-string : ("Hel")
= "o" + "l" + "l" + Reverse sub-string : ("He")
= "o" + "l" + "l" + "e" + Reverse sub-string : ("H")
= "o" + "l" + "l" + "e" + "H"
= "olleH"

OR

Reverse String: "Hello"

= Reverse sub-string : ("ello") + "H"
= Reverse sub-string : ("llo") + "e" + "H"
= Reverse sub-string : ("lo") + "l" + "e" + "H"
= Reverse sub-string : ("o") + "l" + "l" + "e" + "H"
= "o" + "l" + "l" + "e" + "H"
= "olleH"

Think Recursively

Reverse a String="Hello" recursively

```
public static String reverse(String name){  
    if (name == null) return name;  
    if (name.length() == 1) return name;  
  
    return name.substring(name.length()-1) + reverse(name.substring(0, name.length() - 1));  
}
```

Have the leap of faith that
reverse(substring(0, n-1))
works

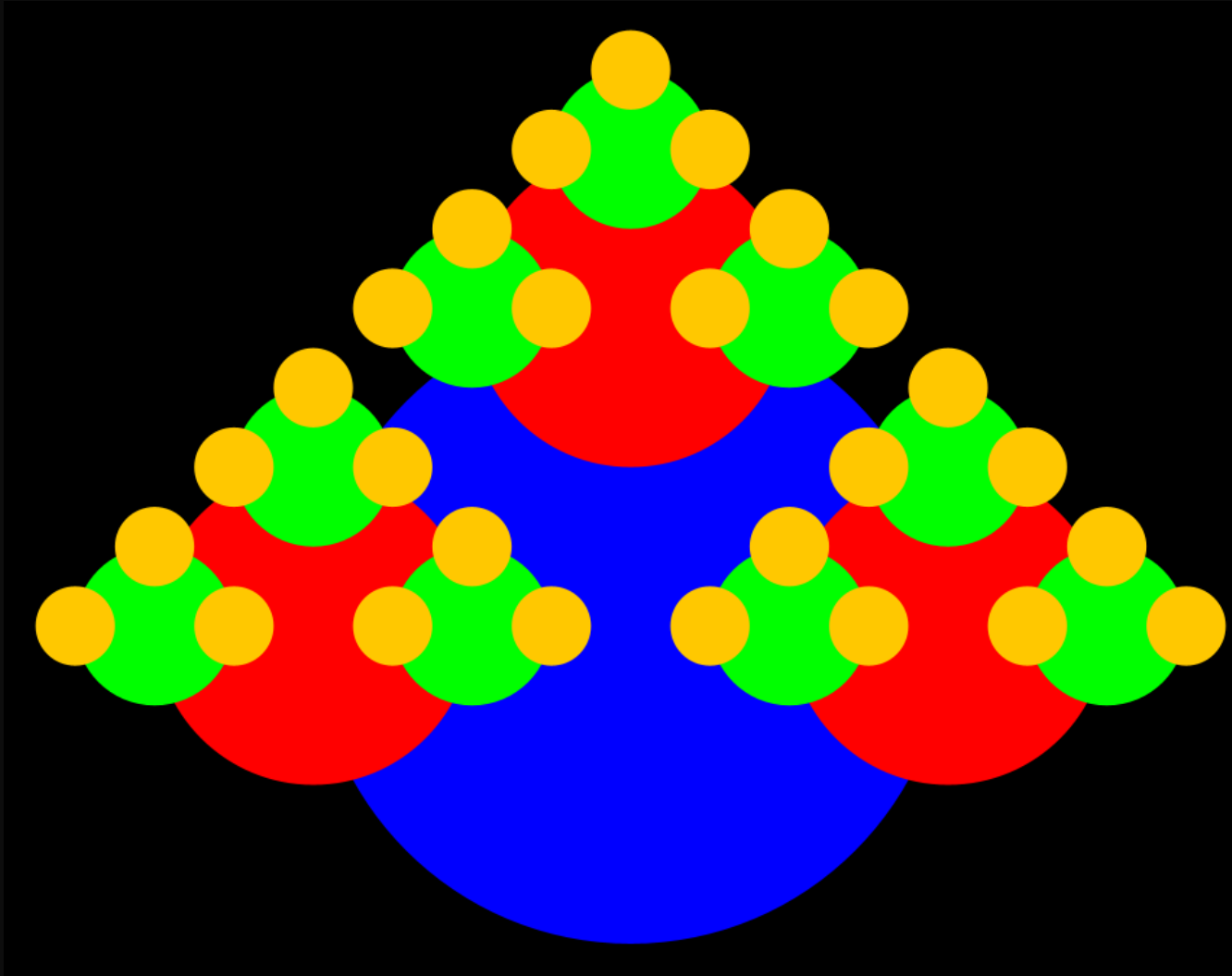
OR

```
public static String reverse(String name){  
    if (name == null) return name;  
    if (name.length() == 1) return name;  
  
    return reverse(name.substring(1)) + name.substring(0, 1);  
}
```

Have the leap of faith that
reverse(substring(1, n))
works

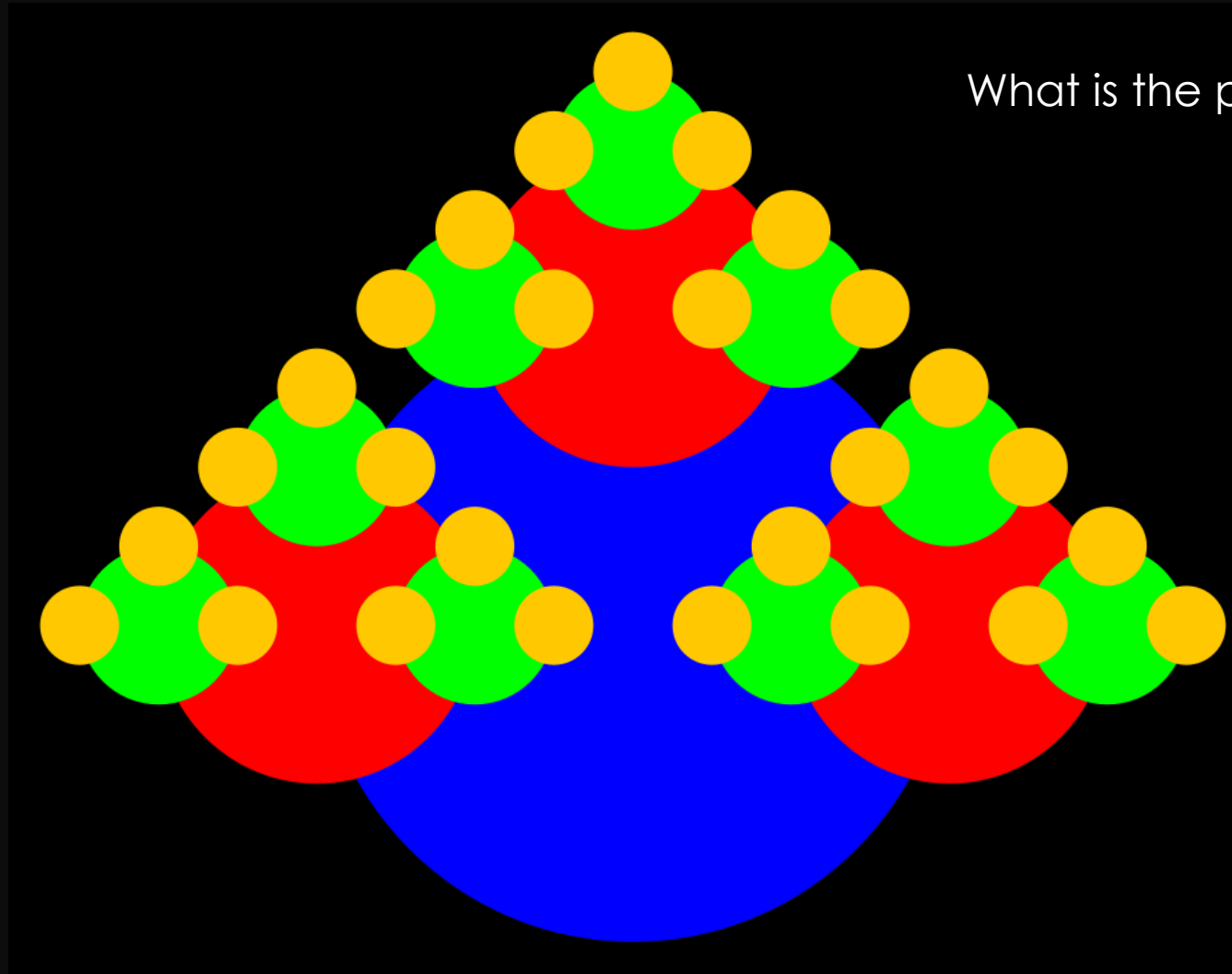
Think Recursively

Draw the pattern below of order/depth = 4



Think Recursively

Draw the pattern below of order/depth = 4

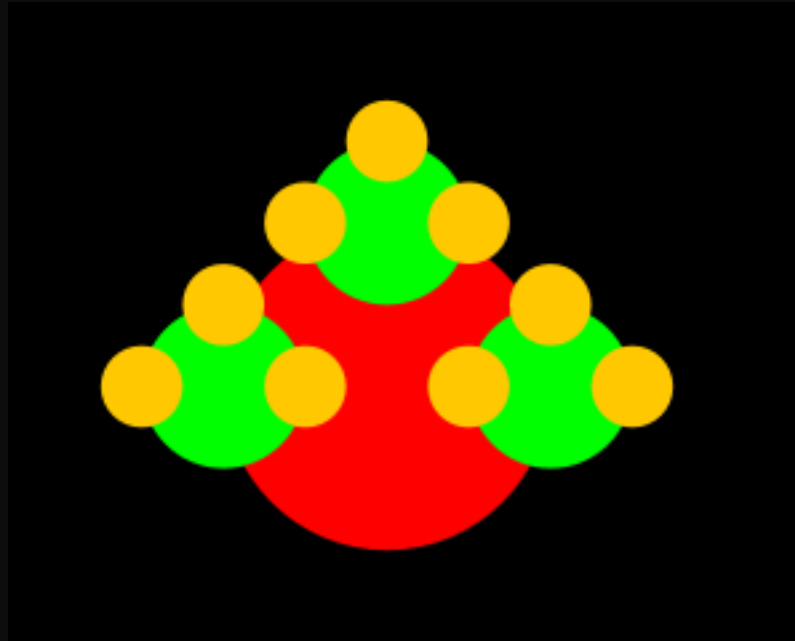


What is the pattern of order/depth = 3?

Think Recursively

Draw the pattern below of order/depth = 3

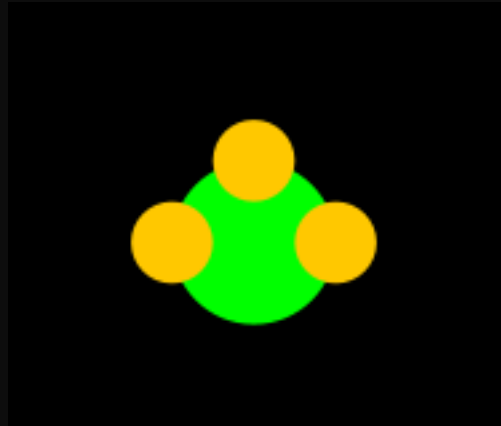
What is the pattern of order/depth = 2?



Think Recursively

Draw the pattern below of order/depth = 2

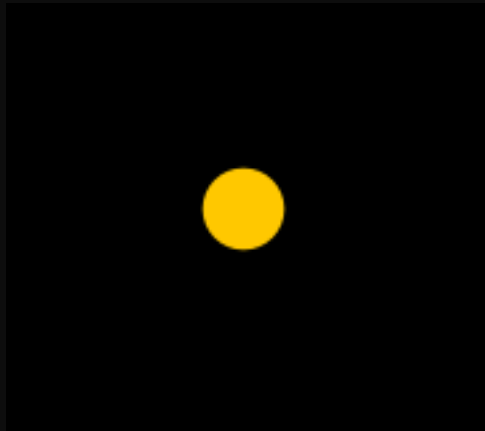
What is the pattern of order/depth = 1?



Think Recursively

Draw the pattern below of order/depth = 1

What is the pattern of order/depth = 0?



Think Recursively

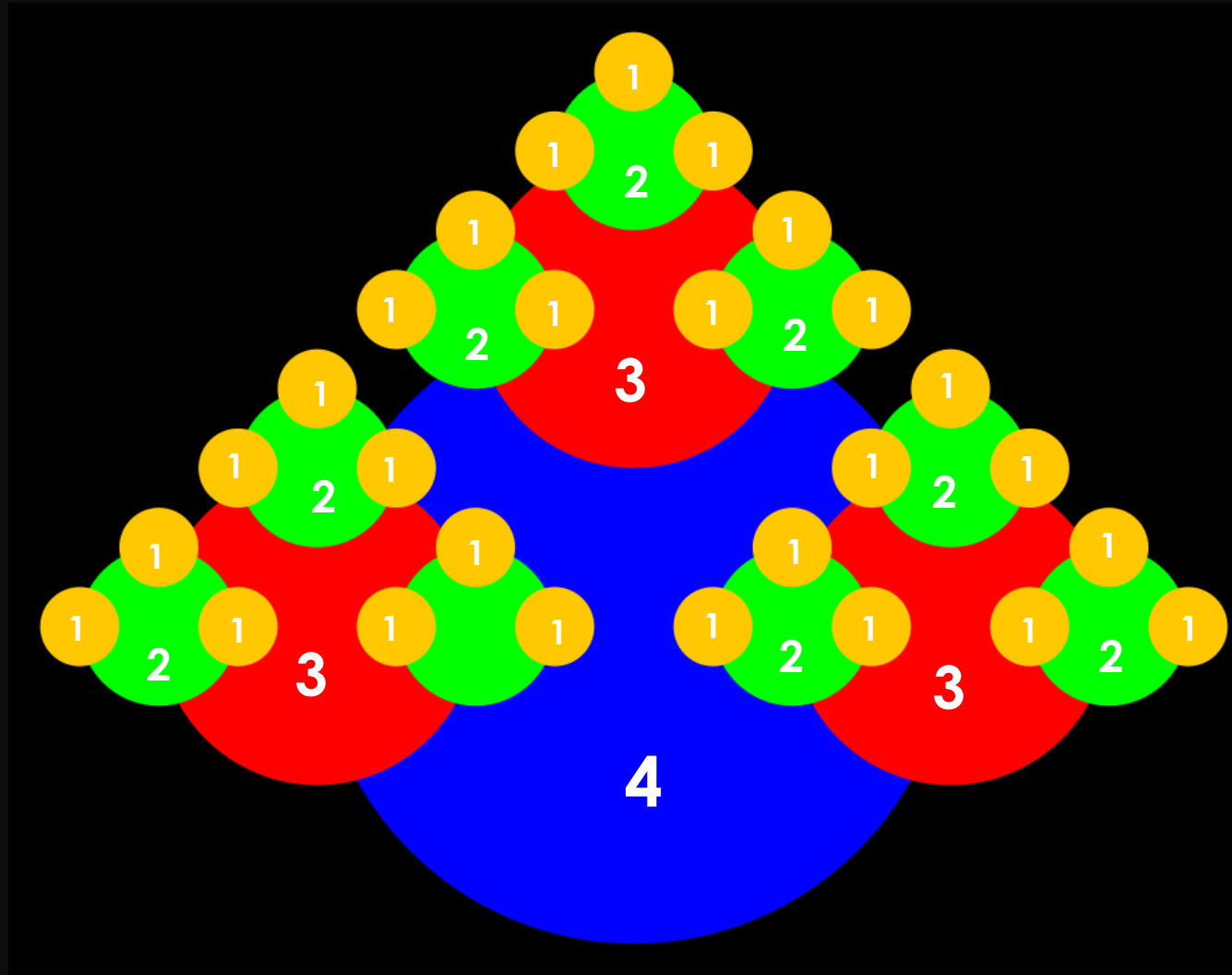
Draw the pattern below of order/depth = 0



Yes, there is nothing
(this is our exit condition)

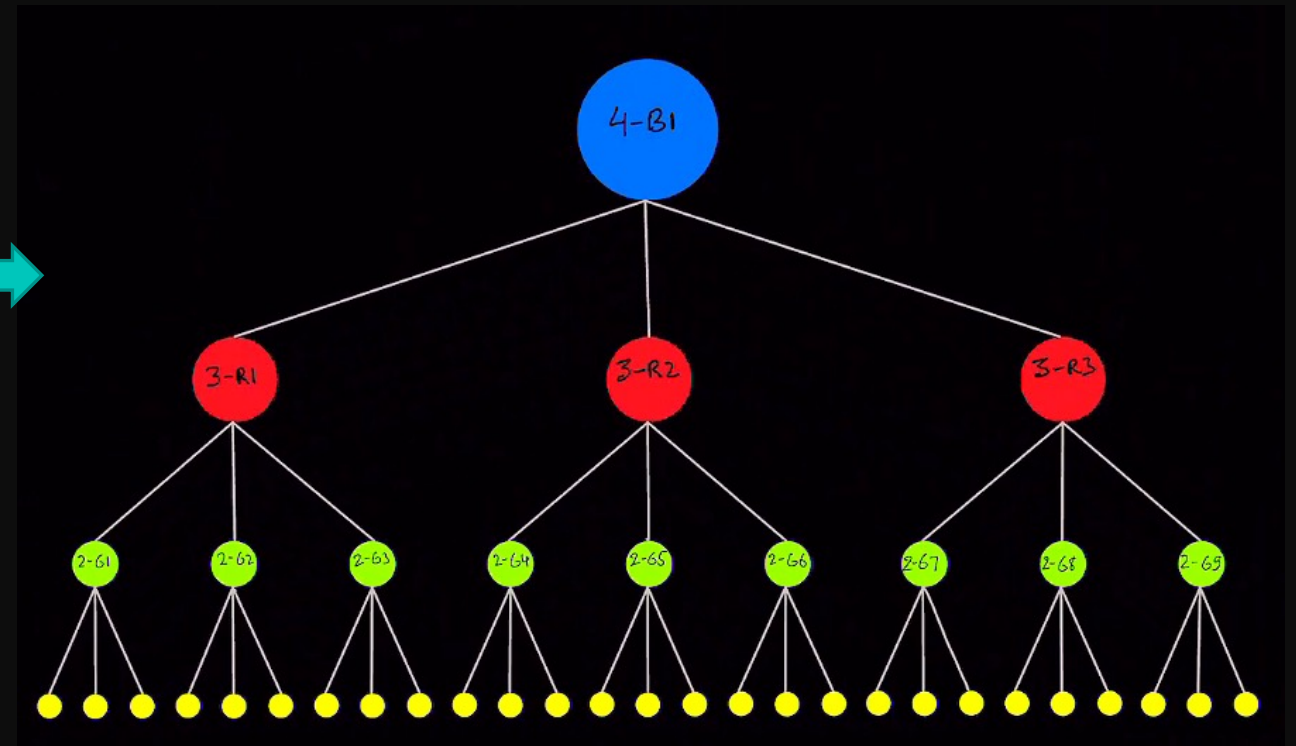
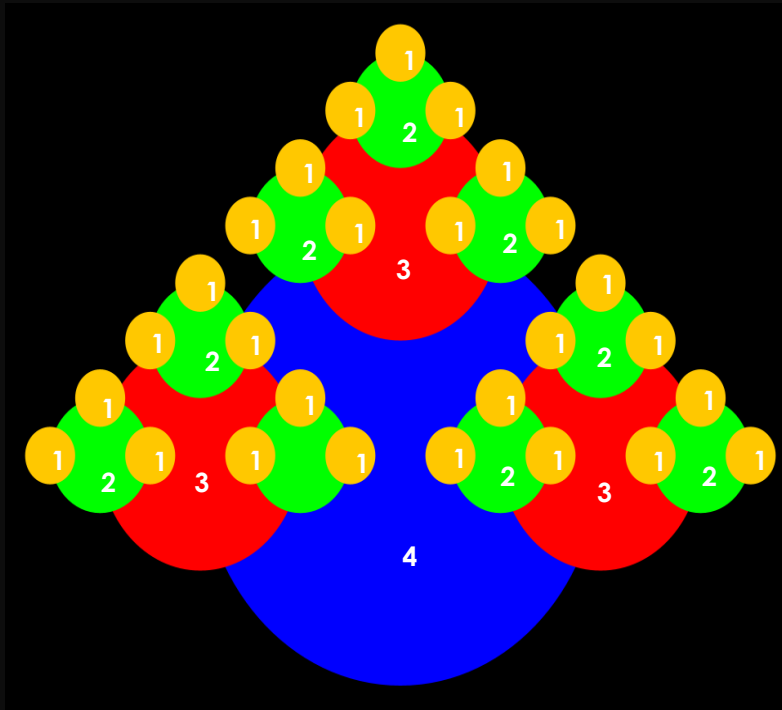
Think Recursively

Draw the pattern below of order/depth = 4



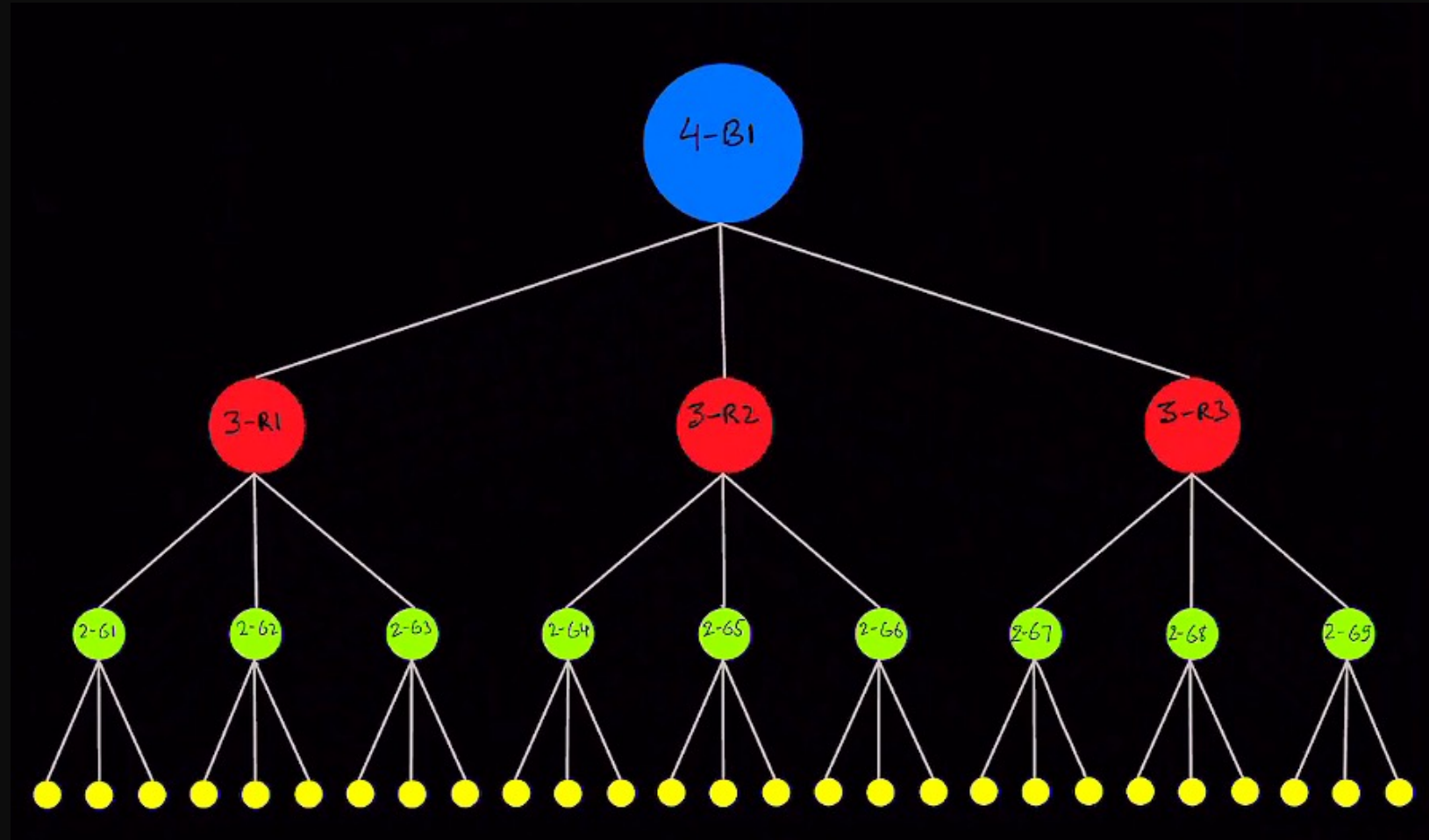
Think Recursively

Draw the Pattern/Tree below of height = 4



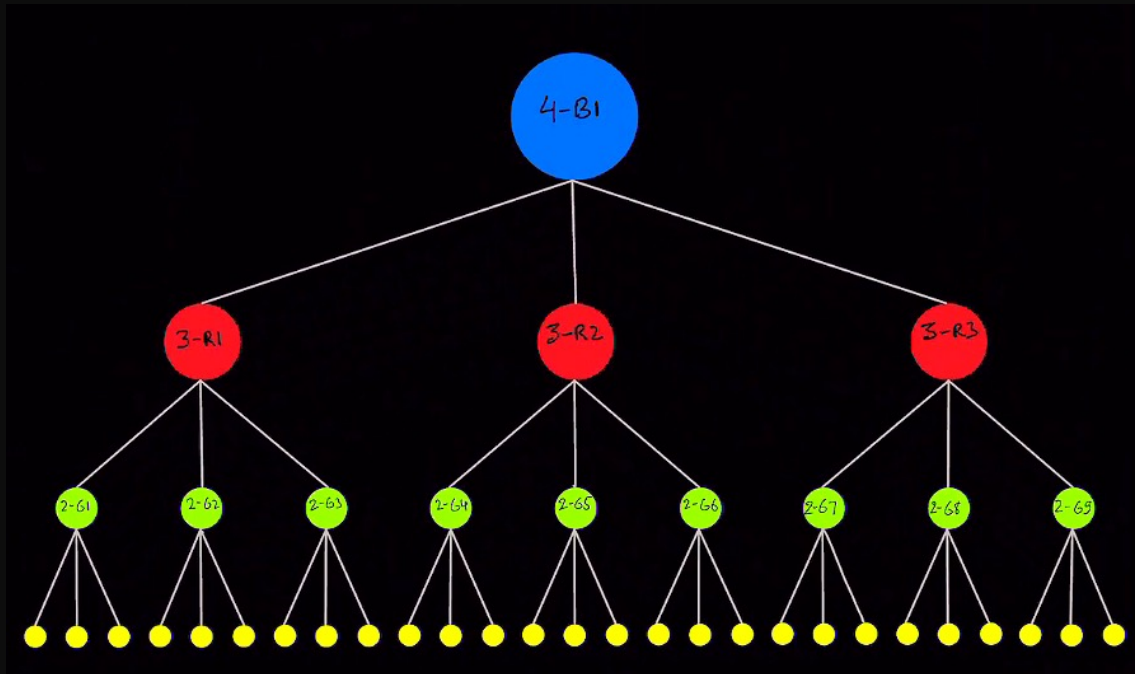
Think Recursively

Draw the Tree below of height = 4



Think Recursively

Draw the Tree below of height = 4



Draw the tree of height = 4

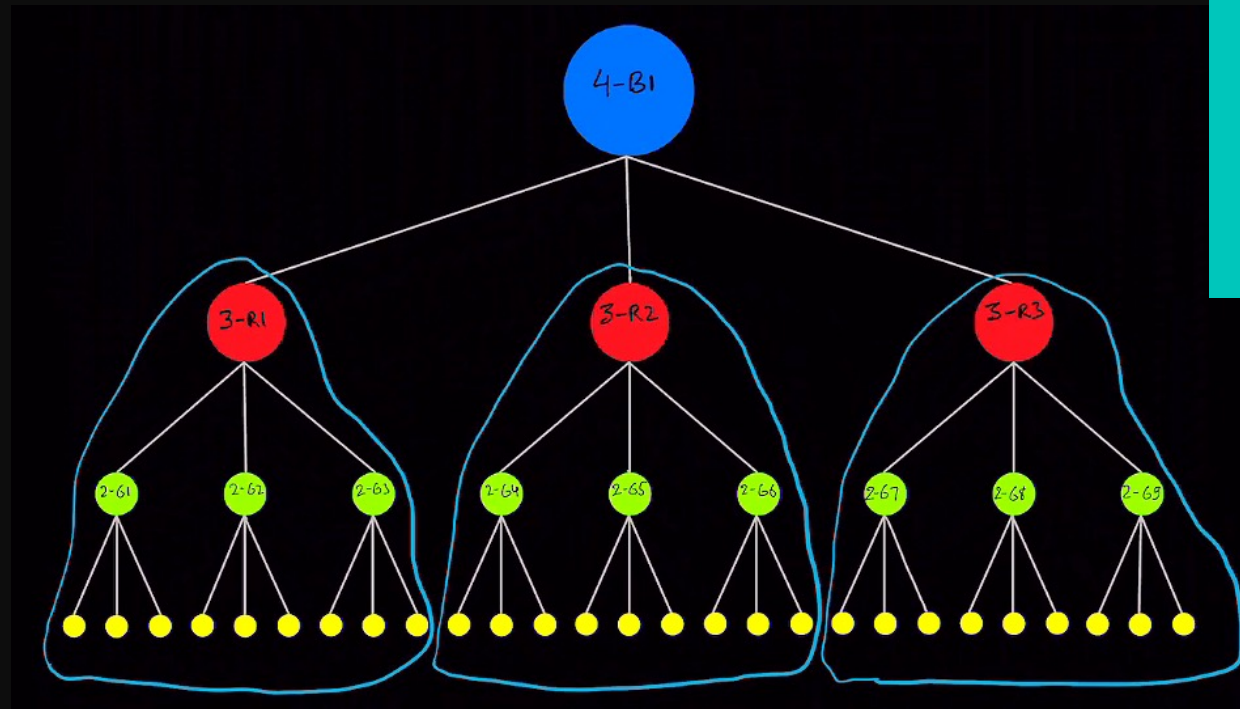
Break this into sub-problems of height = 3

+

Have leap of faith that the subproblem works

Think Recursively

Draw the Tree below of height = 4



Draw the tree of height = 4

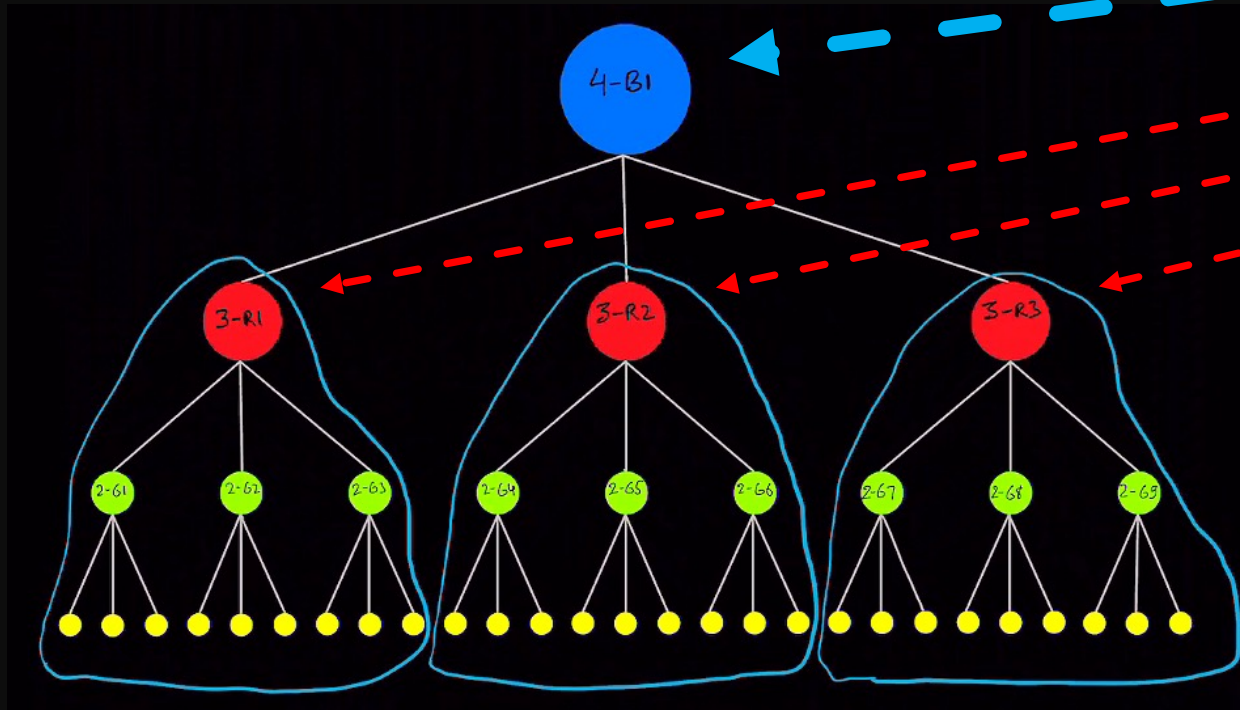
Break this into sub-problems of height = 3

+

Have leap of faith that the subproblem works

Think Recursively

Draw the Tree below of height = 4



Write drawTree recursively

Note: Have the leap of faith that drawTree(height=3) works

```
drawTree(height = 4) {
```

```
    drawNode();
```

```
    drawTree(height = 3);
```

```
    drawTree(height = 3);
```

```
    drawTree(height = 3);
```

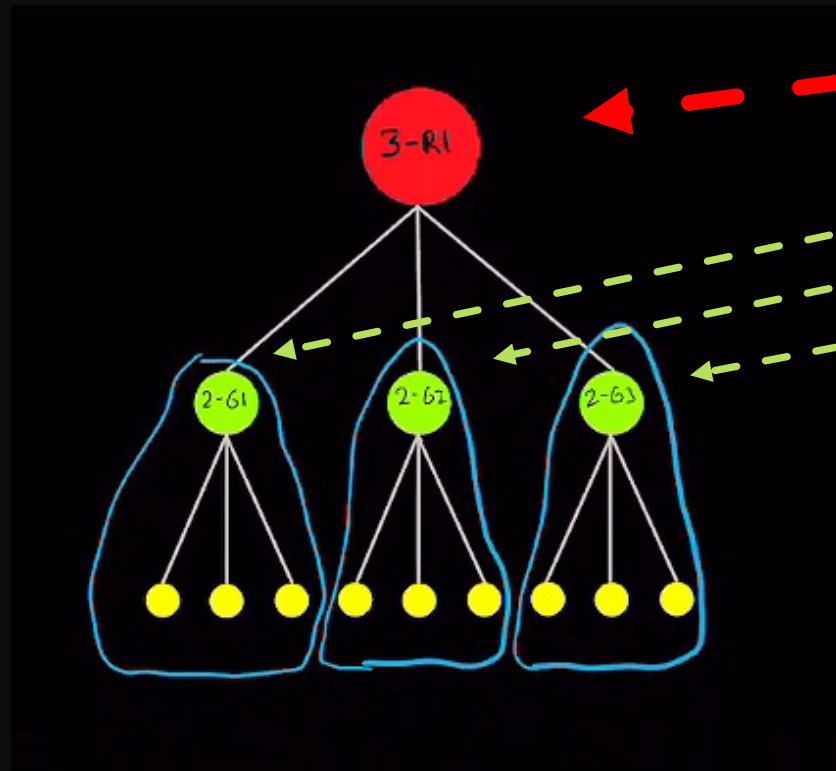
```
}
```

Have faith, that
drawTree(height=3)
works.

(do not trace the code)

Think Recursively

Draw the Tree below of height = 3



Write drawTree recursively

Note: Have the leap of faith that drawTree(height=2) works

```
drawTree(height = 3) {
```

```
    drawNode();
```

```
    drawTree(height = 2);
```

```
    drawTree(height = 2);
```

```
    drawTree(height = 2);
```

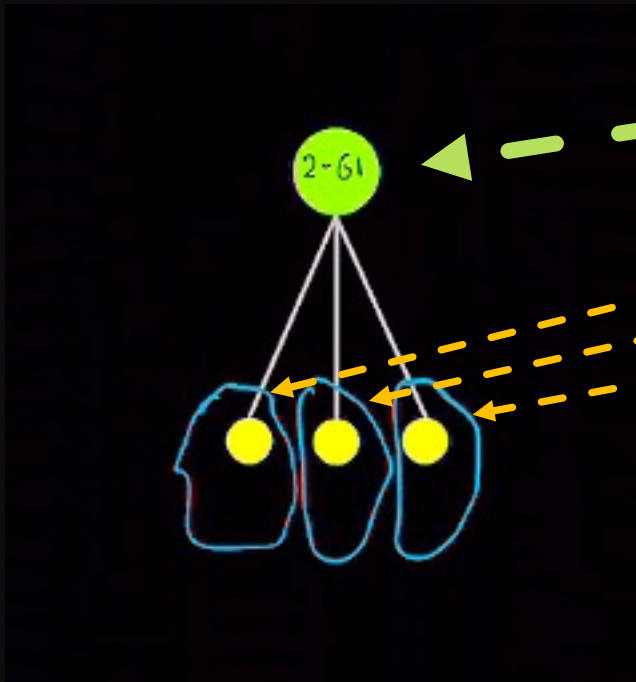
```
}
```

Have faith, that
drawTree(height=2)
works.

(do not trace the code)

Think Recursively

Draw the Tree below of height = 2



Write drawTree recursively

Note: Have the leap of faith that drawTree(height=1) works

```
drawTree(height = 2) {
```

```
    drawNode();
```

```
    drawTree(height = 1);
```

```
    drawTree(height = 1);
```

```
    drawTree(height = 1);
```

```
}
```

Have faith, that
drawTree(height=1)
works.

(do not trace the code)

Think Recursively

Draw the Tree below of height = 1



Draw nothing when height = 0

This is our exit condition

Write drawTree recursively

Note: Have the leap of faith that drawTree(height=0) works

```
drawTree(height = 1) {
```

```
    drawNode();
```

```
    drawTree(height = 0);
```

```
    drawTree(height = 0);
```

```
    drawTree(height = 0);
```

```
}
```

Have faith, that
drawTree(height=0)
works.

(do not trace the code)

Think Recursively

Draw the Tree below of height = n

Write drawTree recursively

Note: Have the leap of faith that drawTree(height = $n-1$) works

```
drawTree(height = n) {  
    drawNode();  
  
    drawTree(height = n - 1);  
    drawTree(height = n - 1);  
    drawTree(height = n - 1);  
}
```

Think Recursively

Draw the Tree below of height = n

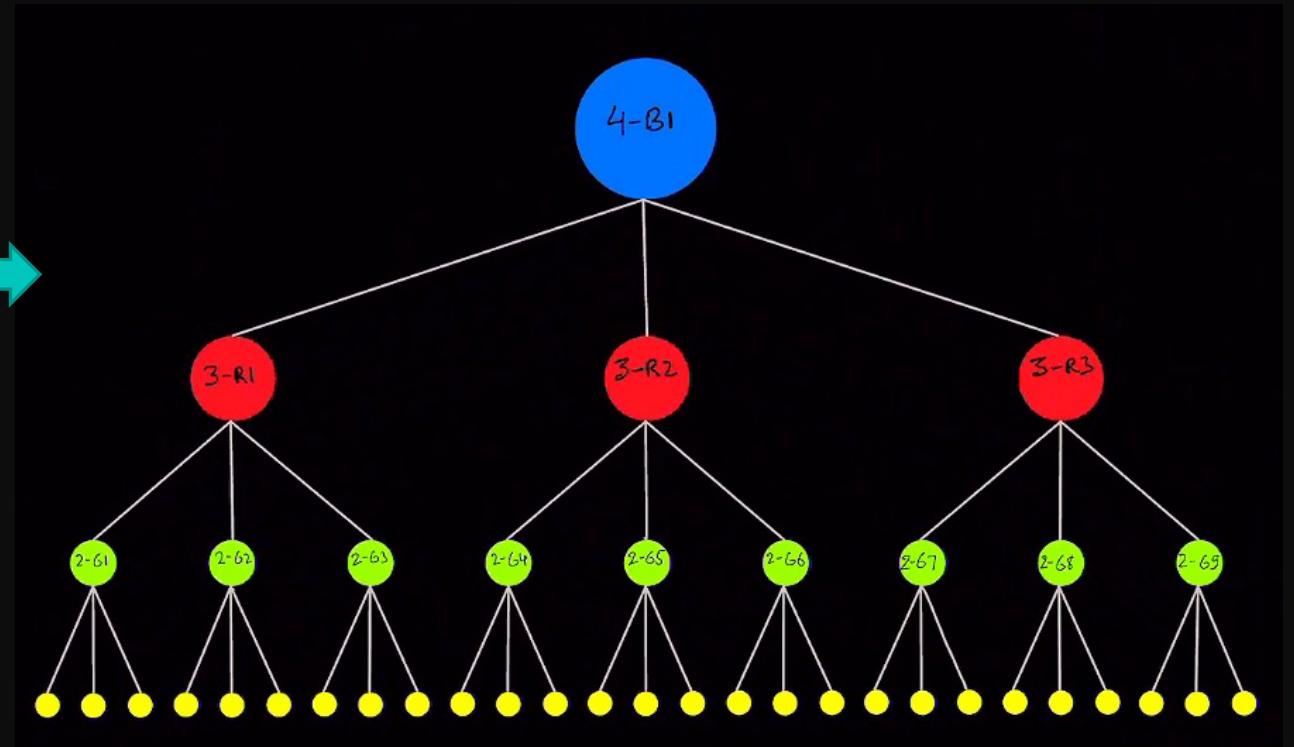
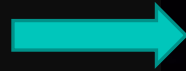
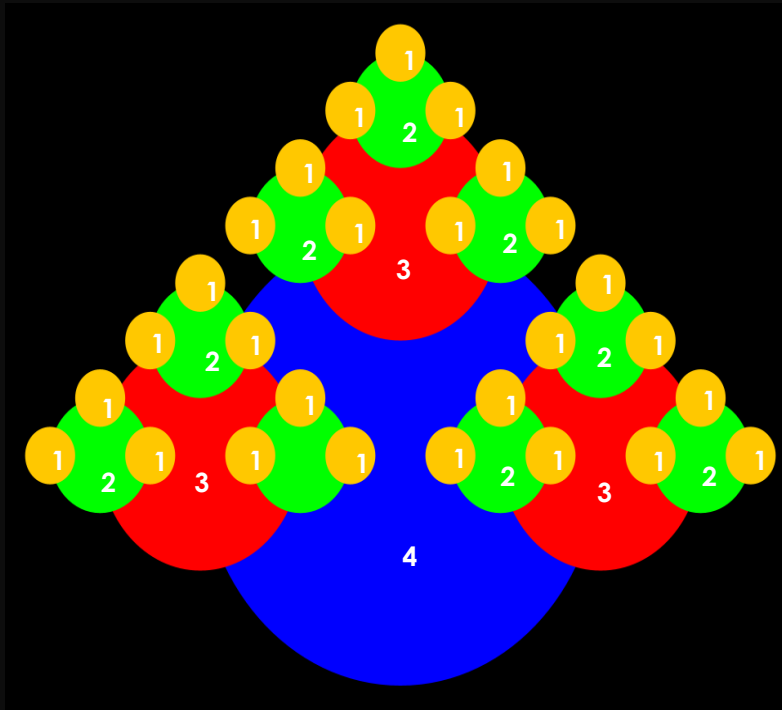
Write drawTree recursively

Note: Have the leap of faith that drawTree(height = $n-1$) works

```
drawTree(height = n) {  
    if (n == 0) return; // draw nothing  
  
    drawNode();  
  
    drawTree(height = n - 1);  
    drawTree(height = n - 1);  
    drawTree(height = n - 1);  
}
```

Think Recursively

Draw the Pattern/Tree below of height = 4



Think Recursively

Draw the Pattern/Tree below of height/depth/order = n

Write drawTree recursively

Note: Have the leap of faith that drawTree(height = $n-1$) works

```
drawTree(height = n) {  
  
    If (n == 0) return; // draw nothing  
  
    drawNode();  
  
    drawTree(height = n - 1);  
    drawTree(height = n - 1);  
    drawTree(height = n - 1);  
}
```

Write drawPattern recursively

Note: Have the leap of faith that drawPattern(order = $n-1$) works

```
drawPattern(order = n, position, size) {  
  
    If (n == 0) return; // draw nothing  
  
    drawCircle(position, size);  
  
    drawPattern(order = n - 1, left, size/2);  
    drawPattern(order = n - 1, center, size/2);  
    drawPattern(order = n - 1, right, size/2);  
}
```