

Section 3: Recurrences and Closed Forms

0. Not to Tree

Consider the function $f(n)$. Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
1 f(n) {  
2     if (n <= 0) {  
3         return 1;  
4     }  
5     return 2 * f(n - 1) + 1;  
6 }
```

- a) Find a recurrence $T(n)$ modeling the *worst-case runtime complexity* of $f(n)$

- b) Find a closed form for $T(n)$

1. To Tree

Consider the function $h(n)$. Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
1 h(n) {  
2   if (n <= 1) {  
3     return 1  
4   } else {  
5     return h(n/2) + n + 2*h(n/2)  
6   }  
7 }
```

a) Find a recurrence $T(n)$ modeling the *worst-case runtime complexity* of $h(n)$

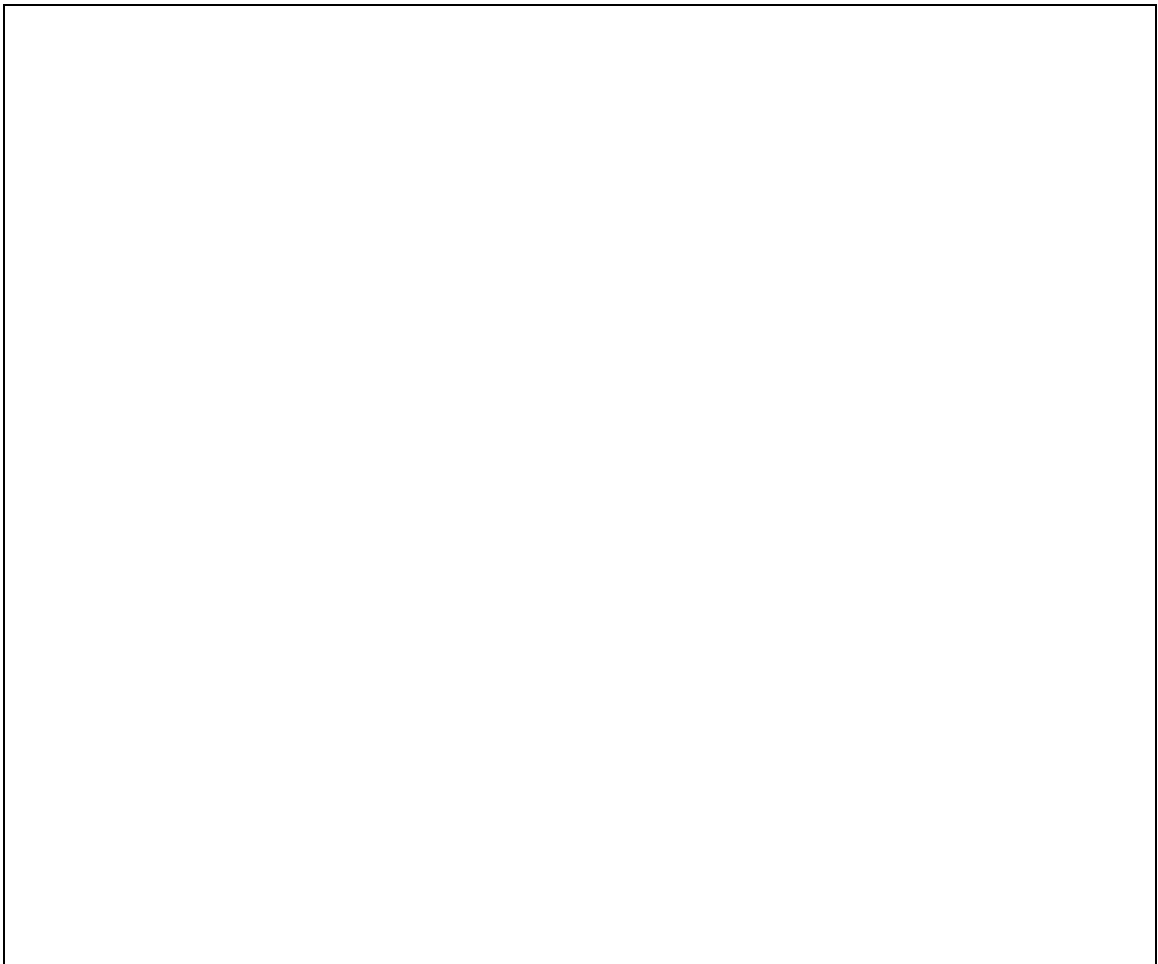
b) Find a closed form for $T(n)$

2. To Tree or Not to Tree

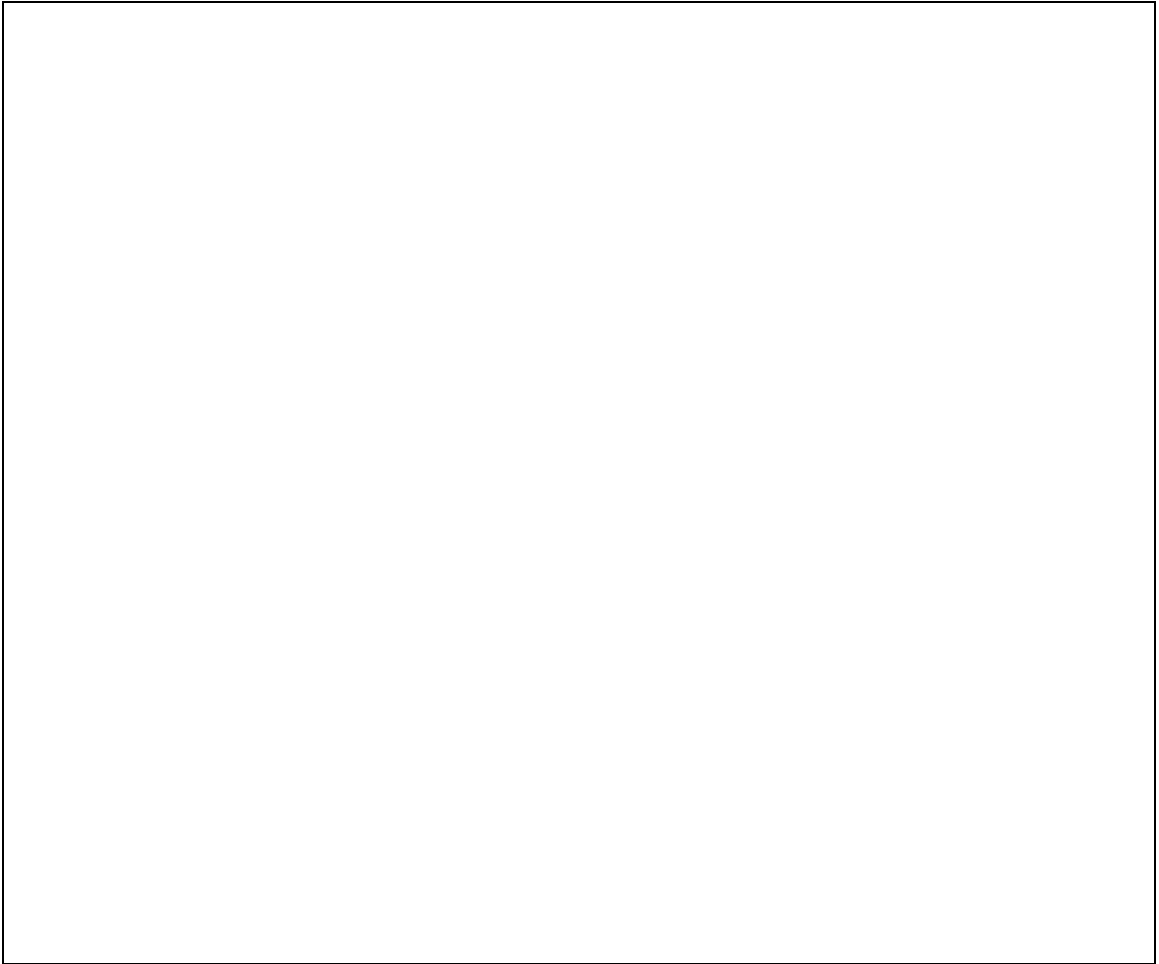
Consider the function $f(n)$. Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
1 f(n) {
2   if (n <= 1) {
3     return 0
4   }
5   int result = f(n/2)
6   for (int i = 0; i < n; i++) {
7     result *= 4
8   }
9   return result + f(n/2)
10 }
```

a) Find a recurrence $T(n)$ modeling the *worst-case runtime complexity* of $f(n)$



b) Find a closed form for $T(n)$

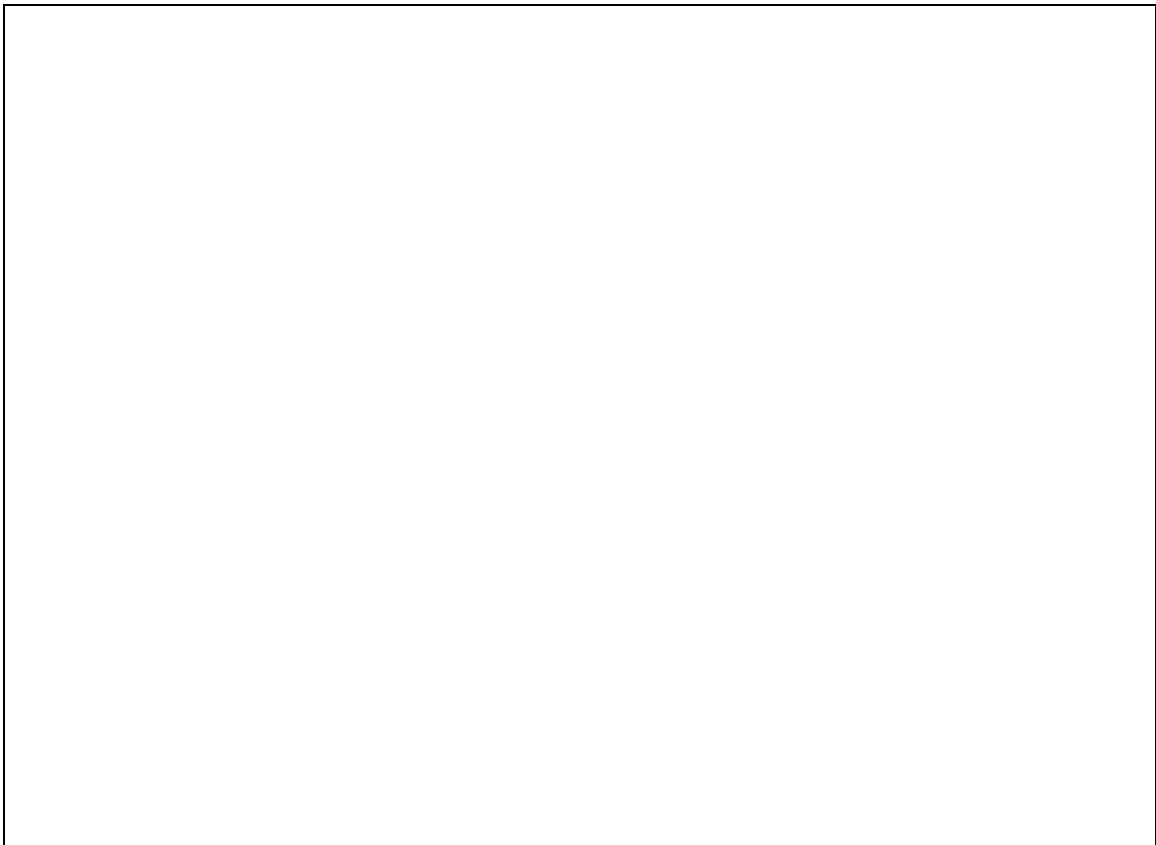
A large empty rectangular box with a thin black border, intended for the student to write their solution for finding a closed form for $T(n)$.

3. Big-Of Bounds

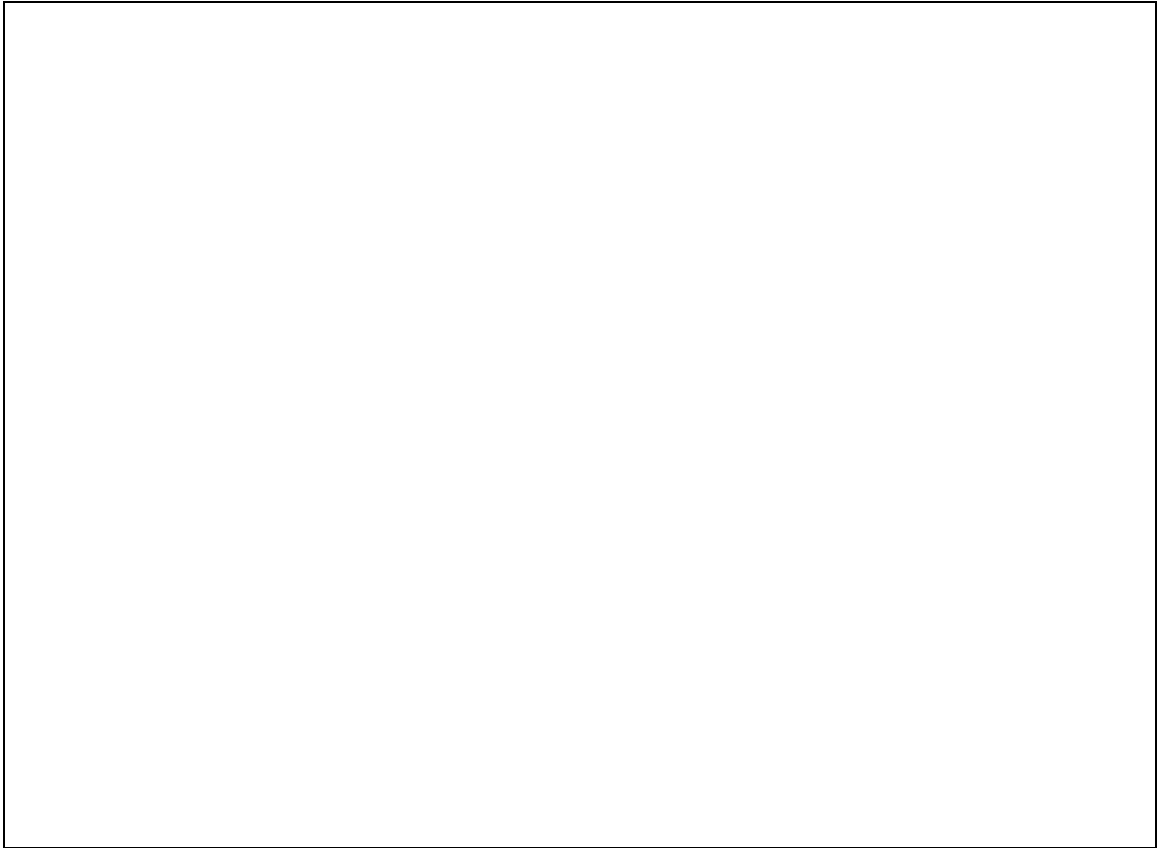
Consider the function $f(n)$. Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
1 f(n) {
2     if (n == 1) {
3         return 0
4     }
5
6     int result = 0
7     for (int i = 0; i < n; i++) {
8         for (int j = 0; j < i; j++) {
9             result += j
10
11         }
12     }
13     return f(n/2) + result + f(n/2)
14 }
```

a) Find a recurrence $T(n)$ modeling the *worst-case runtime complexity* of $f(n)$



b) Find a Big-Oh bound for your recurrence.

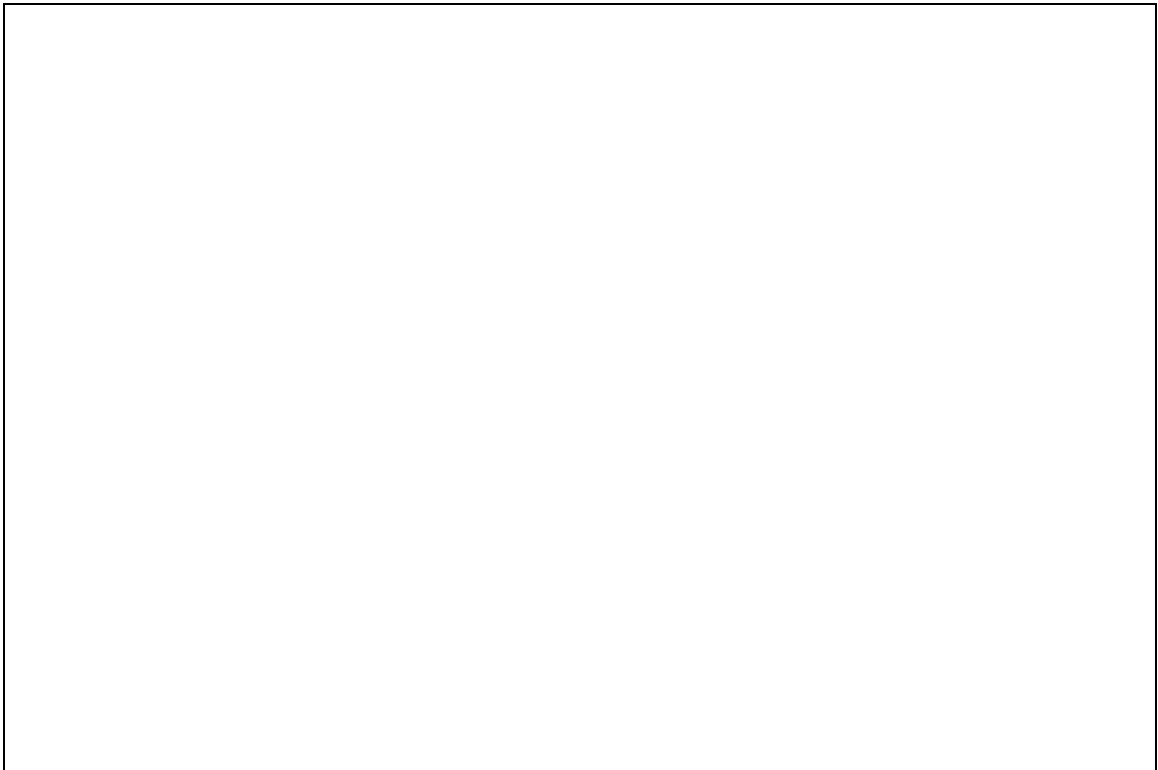
A large empty rectangular box with a thin black border, intended for the student to write their answer to the question above.

4. Odds Not in Your Favor

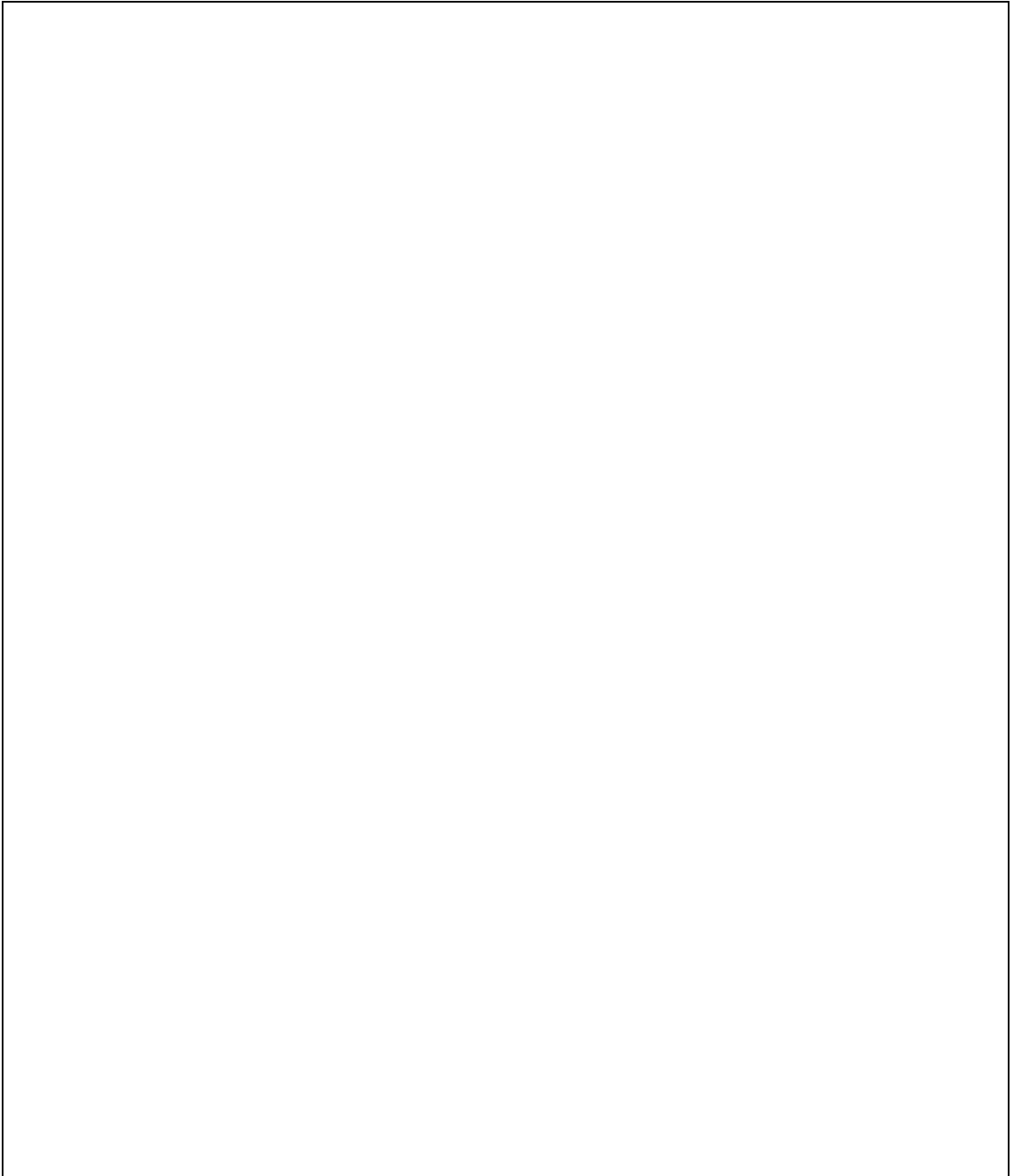
Consider the function $g(n)$. Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
1 g(n) {
2   if (n <= 1) {
3     return 1000
4   }
5   if (g(n/3) > 5) {
6     for (int i = 0; i < n; i++) {
7       println("Yay!")
8     }
9     return 5 * g(n/3)
10  } else {
11    for (int i = 0; i < n * n; i++) {
12      println("Yay!")
13    }
14    return 4 * g(n/3)
15  }
16 }
```

a) Find a recurrence $T(n)$ modeling the *worst-case runtime complexity* of $f(n)$



b) Find a closed form for $T(n)$

A large, empty rectangular box with a thin black border, intended for the student to write their solution to the problem. The box is centered horizontally and vertically on the page.