CSE 332: Data Structures and Parallelism

Section 3: Recurrences and Closed Forms

Terminology	Recurrence Function/Relation	General formula	Closed form General formula evaluated without recurrence function or summations (force them to be in terms of constants or <i>n</i>)	
Definition	Piecewise function that mathematically models the runtime of a recursive algorithm (might want to define constants)	Function written as the number of expansion <i>i</i> and recurrence function (might have a summation)		
Example	$T(n) = c_1 \qquad \text{, for } n = 1$ $T(n) = T\left(\frac{n}{2}\right) + c_2 \text{, otherwise}$	$T(n) = T\left(\frac{n}{2^i}\right) + i \cdot c_2$	Let $i = \log_2 n$, $T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \cdot c_2$ $= T(1) + \log_2 n \cdot c_2$ $= c_1 + \log_2 n \cdot c_2$	

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 }</pre>
```

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

ı				
ı				
ı				
ı				

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.

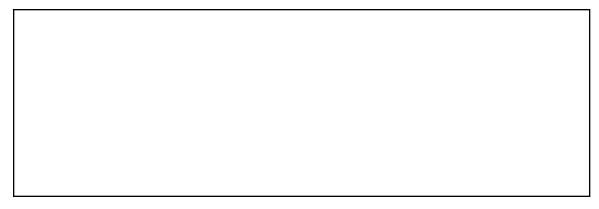
```
1 h(n) {
      if (n <= 1) {
3
           return 1
4
       } else {
           return h(n/2) + n + 2*h(n/2)
5
6
       }
7 }
  a) Find a recurrence T(n) modeling the worst-case runtime complexity of h(n)
  b) Find a closed form for T(n)
```

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.

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

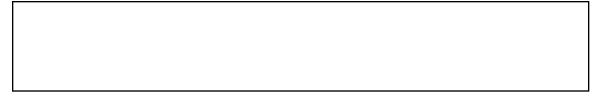


3. Big-Oof Bounds

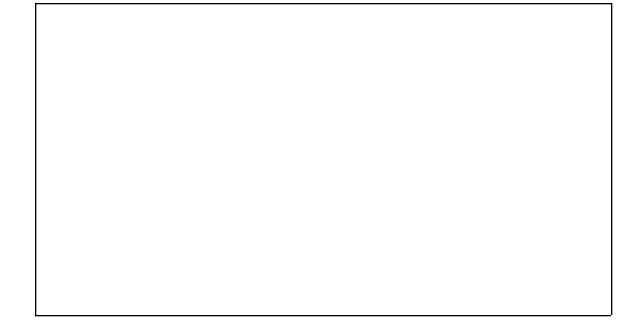
Consider the function f(n). Find a recurrence modeling the worst-case runtime of this function and then find a Big-Oh bound for this recurrence.

```
f(n) {
1
       if (n == 1) {
3
           return 0
4
       }
6
       int result = 0
7
       for (int i = 0; i < n; i++) {
           for (int j = 0; j < i; j++) {
8
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.



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) {
            for (int i = 0; i < n; i++) {
6
7
                println("Yay!")
8
            }
            return 5 * g(n/3)
9
10
       } else {
            for (int i = 0; i < n * n; i++) {
11
                println("Yay!")
12
13
            return 4 * g(n/3)
14
15
        }
16 }
      Find a recurrence T(n) modeling the worst-case runtime complexity of f(n)
  b) Find a closed form for T(n)
```