CSE 332: Data Structures and Parallelism

Section 3: Recurrences and Closed Forms

0. Not to Tree

For the following code snippet, find a recurrence for the worst case runtime of the function, and then find a closed form for the recurrence.

Consider the function f:

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

- Find a recurrence for $f(n)$.

- Find a closed form for $f(n)$.

1. To Tree

Consider the function h:

```
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 to your answer for (a).

2. To Tree or Not to Tree

Consider the function f. 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 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 time complexity of $f(n)$.

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

3. Big-Oof Bounds

Consider the function f. 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 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 time complexity of $f(n)$.

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

4. Odds Not in Your Favor

Consider the function g. Find a recurrence modeling the worst-case runtime of this function, and then find a closed form for the 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    }
11    else {
12       for (int i = 0; i < n * n; i++) {
13          println("Yay!")
14       }
15       return 4 * g(n/3)
16    }
17 }
```

(a) Find a recurrence $T(n)$ modeling the worst-case time complexity of $g(n)$.

(b) Find a closed form for the above recurrence.