### Terminology

<table>
<thead>
<tr>
<th>Definition</th>
<th>Recurrence Function/Relation</th>
<th>General formula</th>
<th>Closed form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piecewise function that mathematically models the runtime of a recursive algorithm (might want to define constants)</td>
<td>Function written as the number of expansion ( i ) and recurrence function (might have a summation)</td>
<td>General formula evaluated without recurrence function or summations (force them to be in terms of constants or ( n ))</td>
<td></td>
</tr>
</tbody>
</table>

### Example

- \( T(n) = c_1 \), for \( n = 1 \)
- \( T(n) = T\left(\frac{n}{2}\right) + c_2 \), otherwise

\[
T(n) = T\left(\frac{n}{2}\right) + i \cdot c_2
\]

Let \( i = \log_2 n \),
\[
T(n) = \left(\frac{n}{2^{\log_2 n}}\right) + \log_2 n \cdot c_2
\]
\[
= T(1) + \log_2 n \cdot c_2
\]
\[
= c_1 + \log_2 n \cdot c_2
\]

### 0. Not to Tree

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

```java
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 $T(n)$
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.

```java
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     return f(n/2) + result + f(n/2)
13 }
```

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) {
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 \( T(n) \)