1. **Memoization for Tribonacci numbers.** The *Tribonacci numbers* are defined by

\[ T_0 = 1, T_1 = 1, T_2 = 2; \]
\[ T_n = T_{n-1} + T_{n-2} + T_{n-3} \text{ for all } n \geq 3. \]

Consider the following pseudocode.

```c
int T[];

int lookupT( n )
    if ( T[n] > 0 )
        return T[n];
    else
        T[n] = lookupT(n - 1) + lookupT(n - 2) + lookupT(n - 3);
        return T[n];

int computeT( n )
    allocate a size n + 1 array for T;
    initialize all entries in T to 0;
    T[0] = 1; T[1] = 1; T[2] = 2;
    return lookupT( n );
```

Argue that `computeT(n)` returns \( T_n \) and analyze its running time.
2. **Graph coloring.** A *coloring* of an undirected graph \( G = (V, E) \) is an assignment of colors to the vertices of \( G \) such that no two adjacent vertices have the same color. The *degree* of a vertex \( u \in V \) is

\[
\delta(u) = |\{v \in V \mid (u, v) \in E\}|,
\]

the number of vertices adjacent to \( u \). Let

\[
\Delta(G) = \max\{\delta(v) \mid v \in V\}
\]

be the largest degree of a vertex in \( G \).

Give an efficient algorithm that colors any graph \( G \) using at most \( \Delta(G) + 1 \) colors. The colors may be named \( c_1, c_2, \ldots \). Show that your algorithm is correct and analyze its running time.
3. **Approximating the minimum hitting set.** Let $U$ be a universe of $n$ elements and let $S$ be a collection of subsets of $U$. A hitting set is a subset $H \subseteq U$ such that for all $S \in S$, $H \cap S \neq \emptyset$.

The *minimum hitting set problem* is to find a hitting set of minimum cardinality. Let

$$m = \max\{|S| \mid S \in S\}.$$

Show that the following algorithm is an $m$-approximation algorithm for this problem.

```python
T = S;
H = \emptyset;
while (T \neq \emptyset)
    choose any set $S \in T$;
    $T = T - \{T \in T \mid S \cap T \neq \emptyset\}$; // remove from $T$ all sets that are hit by elements of $S$
    $H = H \cup S$;
output $H$;
```

*(Hint: Show that the number of iterations of the while loop is a lower bound on \texttt{OPT})*