Problem Set 2: Algorithm Analysis for Recursive Algorithms

1) Solve the following recurrence relation using repeated substitution. Do an inductive proof to show your formula is correct:

$T(1) = 0$
$T(n) = 1/2\ T(n-1) + 1$ for integers $n > 1$

2) Use the Master Theorem to compute the complexity of the MergeSort algorithm by defining a suitable recurrence.

The Master Theorem gives the solutions to recurrences of the form specified below (3 cases).

Let $T(n)$ be a monotonically increasing function that satisfies

$$\begin{aligned} T(n) &= aT(\tfrac{n}{b}) + f(n) \\ T(1) &= c \end{aligned}$$

where $a \geq 1, b \geq 2, c > 0$. If $f(n) \in \Theta(n^d)$ where $d \geq 0$, then

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

3) Let a_n be the number of bit strings of length n that do not have two consecutive 0's.

a) Find a_1 and a_2 .
b) Find a recurrence relation for a_n and hence find hence compute a_6 .

4) Find the big-Oh running time of the following recurrences. Assume that T(1)=1. Use the Master Theorem.

a) $T(n) = 2\ T(n/2) + n^3$
b) $T(n) = 9\ T(n/3) + n$
c) $T(n) = 25\ T(n/5) + n^2$