1. (5 points) Consider the following recursive algorithm that takes an array of positive integers as input, and outputs the largest sum of positions in the array so that no two positions are consecutive, i.e., if we include the \( i^{th} \) element, we cannot include the \((i - 1)^{th}\) element and the \((i + 1)^{th}\) element.

```plaintext
NCS(A[1..n])
- if \( n = 1 \) return(A[1])
- if \( n = 2 \) return(max(A[1], A[2]))
- return(max(a, b))
```

1. Write down a recurrence for the time \( T(n) \) this algorithm takes on an array of size \( n \).
2. Could you use this algorithm on an array of size \( n = 200 \)? Explain your answer.

2. (5 points) Show using induction that for all \( n \geq 1 \),

\[
1 \cdot 7^1 + 2 \cdot 7^2 + 3 \cdot 7^3 + \ldots + n \cdot 7^n = \frac{7n}{6} \cdot 7^n - \frac{7}{36} \cdot 7^n + \frac{7}{36}
\]