Problem 1 (Linear Growth $O(n)$)

Consider the following recursive algorithm to compute a function $F(n)$.

Algorithm F(n):

- if $n \leq 1$, result = 1
- else, result = $F(n-1)+1$

I. Prove that $F(n) = n$.

II. Show that the time complexity of $F(n)$ is $O(n)$.

Problem 2 (Linear Growth $O(n)$)

Consider the following recursive algorithm to compute a function $G(n)$.

Algorithm G(n):

- if $n = 0$, result = 0
- else, result = $G(n-1)+2$

I. Prove that $G(n) = 2n$.

II. Demonstrate that $G(n)$ has a time complexity of $O(n)$.

Problem 3 (Logarithmic Growth $O(\log n)$)

Consider the following recursive algorithm to compute a function $H(n)$.

Algorithm H(n):

- if $n = 1$, result = 1
- else, result = $H(\lfloor n/2 \rfloor)+1$

I. Prove that $H(n) = O(\log n)$.

II. Explain how the recurrence leads to logarithmic growth.

Problem 4 (Quadratic Growth $O(n^2)$)

Consider the following recursive algorithm to compute a function $K(n)$.

Algorithm K(n):

- if $n \leq 1$, result = 1
- else, result = $K(n-1) + n$

I. Prove that $K(n) = \frac{n(n+1)}{2}$.

II. Show that $K(n)$ has a time complexity of $O(n^2)$.

Problem 5 (Exponential Growth $O(2^n)$)

Consider the following recursive algorithm to compute a function $L(n)$.

Algorithm L(n):

- if $n \leq 1$, result = 1
- else, result = $L(n-1) + L(n-2)$

I. Prove that $L(n)$ grows exponentially.

II. Show that $L(n)$ has time complexity $O(2^n)$.

Problem 6 (Linearithmic Growth $O(n \log n)$)

Consider the following recursive algorithm to compute a function $M(n)$.

Algorithm M(n):

- if $n = 1$, result = 1
- else, result = $2M(n/2) + n$

I. Solve the recurrence to show that $M(n) = O(n \log n)$.

II. Show that the algorithm is similar to merge sort and has $O(n \log n)$ time complexity.

Problem 7 (Factorial Growth $O(n!)$)

Consider the following recursive algorithm to compute a function $N(n)$.

Algorithm N(n):

- if $n \leq 1$, result = 1
- else, result = $n \cdot N(n-1)$

I. Prove that $N(n) = n!$.

II. Show that the time complexity is $O(n!)$.

Problem 8 (Exponential Growth $O(3^n)$)

Consider the following recursive algorithm to compute a function $P(n)$.

Algorithm P(n):

- if $n = 1$, result = 1
- else, result = $P(n-1) + 2P(n-2)$

I. Prove that $P(n)$ grows exponentially.

II. Show that $P(n)$ has time complexity $O(3^n)$.

Problem 9 (Cubic Growth $O(n^3)$)

Consider the following recursive algorithm to compute a function $Q(n)$.

Algorithm Q(n):

- if $n = 0$, result = 0
- else, result = $Q(n-1) + n^2$

I. Prove that $Q(n) = \frac{n(n+1)(2n+1)}{6}$.

II. Show that $Q(n)$ has time complexity $O(n^3)$.

Problem 10 (Polynomial Growth $O(n^k)$)

Consider the following recursive algorithm to compute a function $R(n, k)$.

Algorithm R(n, k):

- if $n = 1$, result = 1
- else, result = $R(n-1, k) + n^{k-1}$

I. Prove that $R(n, k) = O(n^k)$.

II. Show that $R(n, k)$ grows polynomially in n.

Problem 11 (Exponential Growth $O(2^n)$)

Consider the following recursive algorithm to compute a function $S(n)$.

Algorithm S(n):

- if $n \leq 2$, result = 1
- else, result = $S(n-1) + S(n-2)$

I. Prove that $S(n)$ grows exponentially like Fibonacci.

II. Show that $S(n)$ has a time complexity of $O(2^n)$.

Problem 12 (Linearithmic Growth $O(n \log n)$)

Consider the following recursive algorithm to compute a function $T(n)$.

Algorithm T(n):

- if $n = 1$, result = 1
- else, result = $2T(n/2) + n$

I. Solve the recurrence to show that $T(n) = O(n \log n)$.

II. Show that $T(n)$ follows the divide and conquer pattern.

Problem 13 (Quadratic Growth $O(n^2)$)

Consider the following recursive algorithm to compute a function $U(n)$.

Algorithm U(n):

- if $n \leq 1$, result = 1
- else, result = $2U(n-1) + n$

I. Prove that $U(n)$ grows quadratically.

II. Show that $U(n)$ has time complexity $O(n^2)$.

Problem 14 (Logarithmic Growth $O(\log n)$)

Consider the following recursive algorithm to compute a function $V(n)$.

Algorithm V(n):

- if $n = 1$, result = 1
- else, result = $V(n/2) + 1$

I. Prove that $V(n) = O(\log n)$.

II. Show that $V(n)$ grows logarithmically.

Problem 15 (Sublinear Growth $O(\sqrt{n})$)

Consider the following recursive algorithm to compute a function $W(n)$.

Algorithm W(n):

- if $n = 1$, result = 1
- else, result = $W(n-1) + 1/\sqrt{n}$

I. Prove that $W(n)$ grows sublinearly like $O(\sqrt{n})$.

II. Show that $W(n)$ grows slower than linear.

Problem 16 (Cubic Growth $O(n^3)$)

Consider the following recursive algorithm to compute a function $X(n)$.

Algorithm X(n):

- if $n = 1$, result = 1
- else, result = $X(n-1) + n^2$

I. Prove that $X(n) = O(n^3)$.

II. Show that $X(n)$ has time complexity $O(n^3)$.

Problem 17 (Log-Linear Growth $O(n \log n)$)

Consider the following recursive algorithm to compute a function $Y(n)$.

Algorithm Y(n):

- if $n = 1$, result = 1
- else, result = $Y(n/2) + n$

I. Solve the recurrence to show that $Y(n) = O(n \log n)$.

Problem 18 (Exponential Growth $O(2^n)$)

Consider the following recursive algorithm to compute a function $Z(n)$.

Algorithm Z(n):

- if $n \leq 2$, result = 1
- else, result = $Z(n-1) + Z(n-2) + Z(n-3)$

I. Prove that $Z(n)$ grows exponentially.

II. Show that the time complexity of $Z(n)$ is $O(2^n)$.