1) Solve the following recurrence relation

a) $x(n) = x(n-1) + 5$ for $n > 1$ with $x(1) = 0$

i) write down the first two terms to identify the pattern

$x(1) = 0$

$x(2) = x(1) + 5 = 5$

$x(3) = x(2) + 5 = 10$

$x(4) = x(3) + 5 = 15$

ii) identify the pattern (or) the general term

→ the first term $x(1) = 0$

the common difference $d = 5$

The general formula for the n^{th} term of an AP is

$$x(n) = x(1) + (n-1)d$$

substituting the given value

$$x(n) = 0 + (n-1)5 = 5(n-1)$$

The solution is $x(n) = 5(n-1)$

b) $x(n) = 3x(n-1)$ for x with $x(1) = 4$.

i) write down the first two terms to identify the pattern.

$x(1) = 4$

$x(2) = 3x(1) = 3 \cdot 4 = 12$

$x(3) = 3x(2) = 36$

$x(4) = 3x(3) = 108$

ii) identify the general term.

→ the first term $x(1) = 4$

→ the common ratio $r = 3$

The general formula for the n^{th} term of a GP is

$$x(n) = x(1) r^{n-1}$$

substituting the given value

$$x(n) = 4 \cdot 3^{n-1}$$

The solution is

$$x(n) = 4 \cdot 3^{n-1}$$

$x(n) = x(\frac{n}{2}) + n$ for $n > 1$ with $x(1) = 1$ (solve for $n = 2^k$)

for $n = 2^k$, we can write recurrence in terms of k

1) substitution $n = 2^k$ in the recurrence

$$x(2^k) = x(2^{k-1}) + 2^k$$

2) write down the first few terms to identify the pattern.

$x(1) = 1$

$x(2) = x(2^1) = x(1) + 2 = 1 + 2 = 3$

$x(4) = x(2^2) = x(2) + 4 = 3 + 4 = 7$

$x(8) = x(2^3) = x(4) + 8 = 15$

3) identify the general term by finding the pattern

we observe that:-

$$x(2^k) = x(2^{k-1}) + 2^k$$

we sum the series:

$$x(2^k) = 2^k + 2^{k-1} + 2^{k-2} + \cdots$$

the geometric series with the term $a = 2$ and the last 2^k except for the additional $+1$ term.

the sum of a geometric series is with ratio r is given by

$$S = \frac{a r^n - 1}{r - 1}$$

here $a = 2$, $r = 2$, and $n = k$

$$S = 2\,\frac{2^k - 1}{2 - 1} = 2(2^k - 1) = 2^{k+1}$$

Adding the $+1$ term

$$x(2^k) = 2^{k+1} - 2 + 1 = 2^{k+1} - 1$$

Solution is

$$x(2^n) = 2^{n+1} - 1$$

Evaluate the following recurrences by iterating.

i) $T(n) = T(n/2) + 1$, where $n = 2^k$ for all $k \geq 1$

The recurrence relation can be solved using iteration method.

1) substitute $n = 2^k$ in the recurrence

2) iterate the recurrence.

for $k = 0$: $T(2^0) = T(1) = T(1)$

$k = 1$: $T(2^1) = T(1) + 1 = T(2)$

$k(2)$: $T(2^2) = T(8) = T(n) + 1 = (T(1) + 1) + 1 = T(1) + 2$

$k = 3$: $T(2^3) = T(8) = T(n) + 1 = (T(1) + 1) + 1 = T(1) + 3$

3) generalize the pattern.

$$T(2^k) = T(1) + k$$

since $n = 2^k$. $k = \log_2 n$

$$T(n) = T(2^k) = T(1) + \log_2 n$$

4) assume $T(1)$ is a constant c

$$T(n) = c + \log_2 n$$

The solution is

$$T(n) = O(\log n)$$

ii) $T(n) = T(2n/3) + T(2n/3) + cn$ where is constant and n is input size.

The recurrence can be solved using the master's theorem for divide-and-conquer recurrence of the form

$$T(n) = aT(n/b) + f(n)$$

where $a = 2$, $b = 3$, and $f(n) = cn$

Let determine the value of $\log_b a$

$$\log_b a = \log_3 2$$

using the Properties of logarithms

$$\log_3 2 = \frac{\log 2}{\log 3}$$

Now we compare $f(n) = cn$ with $n \log_3^2$

$f(n) = d(n)$

$n = n$

since $c > \log_3^2$ we are in the third case of the master theorem

$f(n) = O(n^a)$ with $a \geq \log_b^a$

∴ the solution is

$$T(n) = \Theta(f(n)) = d(n) = cn)$$

3) consider the following recursive algorithm

```
min [A[0 ... n-1]
if n=1 return A[0]
else temp = min[A[0...n-2]]
    if temp < A[n-1] return temp
    else
        return A[n-1]
```

a) what does this algorithm compute

solution: the given algorithm min $[A[0 \ldots n-1]]$ computes the minimum value in the array 'A' from index '0' to 'n-1' it does this by recursively finding the minimum value in the sub array $A[0 \ldots n-2]$ and then comparing it with the last element $A[n-1]$ to determine the over all maximum value.

b) setup a recurrence relation for the algorithm's basic operation count and solve it

to determine the recurrence relation for the algorithms basic operation count, let's analize the steps involved in the algorithm. the basic operation are the comparisons and function calls.

Recurrence relation setup

base case when n=1, the algorithm performs a single operation to return A[0]

2) Recursive case: when n>1, the algorithm
makes a recursive call to min(A[0...n-2]);
performs a comparison b/w temp and A[n-1]

Let T(n) represent the no. of basic operation the algorithm performs for an array of size n

1) Base case:
$T(1) = 1$

2) Recursive case
$T(n) = T(n-1)+1$

Here T(n-1) accounts for the operations performed by the recursive call to min(A[0...n-2]) and the +1 accounts for the comparison b/w temp and A[n-1]

To solve this recurrence relation we can use iteration method.

$T(n) = T(n-1)+1$
$= (T(n-2)+1)+1$
$= ((T(n-3)+1)+1)+1$
$= \cdots$
$= 1 + T(n-1)$
$= n.$

the solution is
$T(n) = n$

This means the algorithm performs n basic operations for an input array of size n.

4) Analyze the order of growth.

i) $f(n) = 2n^2+5$ and $g(n) = 7n$ use the $\Omega(g(n))$ notation

Solution:

To analyze the order of growth and use the Ω notation we need to consider the given functions $f(n)$ and $g(n)$

Given functions

$f(n) = 2n^2+5$

$g(n) = 7n$

order of growth using $\Omega(g(n))$ notation..

The notation $\Omega(g(n))$ describing a lower bound on the rate of a function. Specifically $f(n) = \Omega(g(n))$ means the for sufficient large n, $f(n)$ grows at least as fast as g

formally $f(n) = \Omega g(n)$ if there exist positive constant c and no such that for all $n \geq n_0$

$$f(n) \geq (g(n))$$

Let analyze $f(n) = 2n^2+5$ with respect to $g(n) = 7n$

i) Identify dominant terms

⇒ The dominant terms in $f(n)$ is $2n^2$ since it grows faster than the constant terms as increases

⇒ The dominant terms in $g(n)$ is $7n$

ii) establish the inequality

→ we want to find constants c and no such that

$2n^2+5 \geq c\,7n$ for all $n \geq 0$

3) simplify the inequality.

→ ignore the lower order terms for larger

$2n^2 \geq 7cn$

→ divide both sides by n

$2n \geq 7c$

→ solve for n

$n \geq 7c/2$

4) choose constant.

let $c = 1$

$n \geq \frac{7 \cdot 2}{2} = 3.5$

∴ for $n \geq n_0$, the inequality holds.

$2n^2 + 5 \geq 7n$ for all $n \geq n_0$

we have shown that there exist constants $c = 1$ and $n_0 = n$ such that for all $n \geq n_0$.

$2n^2 + 5 \geq 7n$

Thus, we can conclude that:

$f(n) = 2n^2 + 5 = \Omega(7n)$

in Ω notation, the dominant term $2n^2$ in $f(n)$, clearly grows faster than $7n$. Hence

$f(n) = \Omega(n^2)$

however, for the specific comparision asked $f(n) = \Omega(7n)$ is also correct

showing that $f(n)$ grows at least as fast as $7n$.