COMP2230/COMP6230 Algorithms

Lecture 2

Professor Ljiljana Brankovic

Lecture Overview

- Recurrence Relations
- Analysis of Algorithms

Recurrence Relations

A ***recurrence relation*** for the sequence $a_0, a_1, \ldots$ is an equation that relates a_n to certain of its predecessors $a_0, a_1, \ldots, a_{n-1}$.

Initial conditions for the sequence $a_0, a_1, \ldots$ are explicitly given values for the finite number of terms of the sequence.

Example 1 :

Fibonacci sequence: $f_n = f_{n-1} + f_{n-2}$ *for* $n \geq 3$

Initial conditions: $f_1 = f_2 = 1$

Example 1: Fibonacci sequence

Suppose that at the beginning of the year there is one pair of new-born rabbits and that every month each pair produces a new pair that becomes productive after one month. Suppose further that no deaths occur. Let a_n denote the number of rabbits at the beginning of n^{th} month. Show that $a_n = f_n, \ n \geq 1$.

Months	Pairs	Comment
1	1	
2	1	The pair did not multiply because they just become productive.
3	2	The first pair multiplied.
…	…	…
$n-2$	a_{n-2}	
$n-1$	a_{n-1}	
n	$a_{n-1} + a_{n-2}$	Each pair that was alive 2 months ago reproduced.

Solving Recurrence Relations Using Iteration (Substitution)

Example 2:

$$c_n = c_{n-1} + n, \quad n \geq 1$$
$$c_0 = 0$$

$$c_{n-1} = c_{n-2} + (n-1)$$
$$c_{n-2} = c_{n-3} + (n-2)$$
$$\dots$$
$$c_n = c_{n-1} + n$$
$$= c_{n-2} + (n-1) + n$$
$$= c_{n-3} + (n-2) + (n-1) + n$$
$$\dots$$
$$= c_0 + 1 + 2 + 3 + \cdots + (n-2) + (n-1) + n$$
$$= 0 + 1 + 2 + 3 + \cdots + (n-2) + (n-1) + n$$
$$= \frac{n(n+1)}{2}$$

Solving Recurrence Relations Using Iteration (Substitution)

Example 3:

$$c(n) = n + c(\lfloor \frac{n}{2} \rfloor), \qquad n > 1$$

$$c(1) = 0$$

Solution for $n = 2^k$, for some k.

$$\begin{aligned} c(2^k) &= 2^k + c(2^{k-1}) \\ &= 2^k + 2^{k-1} + c(2^{k-2}) \\ &\dots \\ &= 2^k + 2^{k-1} + \dots + 2^1 + c(2^0) \\ &= 2^{k+1} - 2 = 2n - 2 \end{aligned}$$

Other ways for solving recurrences

- Guess and prove by mathematical induction

- Match with familiar forms
  - homogeneous/ non-homogeneous forms

Example 4

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 2 + 4T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) & \text{otherwise} \end{cases}$$

Restrict n to be a power of 2: $n = 2^k$

Look at the expansion:

n	$T(n)$
2^0	2
2^1	$2 + 4 \times 2$
2^2	$2 + 4 \times 2 + 4^2 \times 2$

Pattern seems to suggest:

$$T(n) = T(2^k) = \sum_{i=0}^{k} (2 \times 4^i) = 2\frac{4^{k+1} - 1}{4 - 1} = \frac{8n^2 - 2}{3} = O(n^2)$$

Example 4

Proof by mathematical induction:

- Base case: $k = 0, n = 2^0$

$$T(1) = \frac{8-2}{3} = 2$$

Basis holds.

- Assumption: True for k.

- Proof that it is true for $k+1$:

$$T\left(2^{k+1}\right) = 2 + 4T\left(2^k\right) = 2 + 4\frac{8\times 2^{2k}-2}{3} = \frac{6+8\times 2^{2k+2}-8}{3} = \frac{8\times 2^{2k+2}-2}{3}$$

Homogeneous Recurrences

Characteristic equation of the recurrence:

$$a_0 t_n + a_1 t_{n-1} + a_2 t_{n-2} + \ldots + a_k t_{n-k} = 0$$

The recurrence is:

- Linear, as it does not contain t_i^2, etc.
- Homogeneous, as the linear combination of t_i's is equal to 0.
- With constant coefficients, as a_i's are constants.

Homogeneous Recurrences

Guess: $t_n = x^n$, for an (as yet) unknown x

$$a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_k x^{n-k} = 0$$

$$(a_0 x^k + a_1 x^{k-1} + a_2 x^{k-2} + \cdots + a_k x^0) x^{n-k} = 0$$

Trivial solution $x = 0$ is not interesting for us. Thus,

$$a_0 x^k + a_1 x^{k-1} + a_2 x^{k-2} + \cdots + a_k x^0 = 0$$

Homogeneous (cont)

$$p(x) = a_0 x^k + a_1 x^{k-1} + a_2 x^{k-2} + \cdots + a_k x^0 = 0$$

$p(x)$ is the characteristic polynomial; any polynomial of degree k has exactly k roots (not necessarily all distinct).

Let $r_1, r_2, \ldots, r_k$ be the roots of this polynomial.

If roots are distinct, the general solution to the recurrence is

$$t_n = \sum_{i=1}^{k} c_i r_i^n$$

where c_i are some constants that can be determined from initial conditions.

Example 5

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) & \text{otherwise} \end{cases}$$

$$T(n) - 2T(n-1) = 0$$

$$T(n) = x^n$$

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

Solve $x^n - 2x^{n-1} = 0$:

$$(x - 2)x^{n-1} = 0, \qquad x - 2 = 0, \qquad x = 2$$

$$T(n) = c_1 2^n$$

$$T(1) = 1 = c_1 2^1$$

$$c_1 = \frac{1}{2}$$

$$T(n) = 2^{n-1}$$

Example 6

Fibonacci Sequence

$$f_n = \begin{cases} n & \text{if } n = 0 \text{ or } n = 1 \\ f_{n-1} + f_{n-2} & \text{otherwise} \end{cases}$$

f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	. . .
0	1	1	2	3	5	8	13	. . .

Example 6 - Fibonacci (cont)

- Recurrence? $f_n - f_{n-1} - f_{n-2} = 0$
- Characteristic equation? $x^2 - x - 1 = 0$
- Roots? $x_1 = \frac{1+\sqrt{5}}{2}$, $x_2 = \frac{1-\sqrt{5}}{2}$
- General form? $f_n = c_1\left(\frac{1+\sqrt{5}}{2}\right)^n + c_2\left(\frac{1-\sqrt{5}}{2}\right)^n$
- Constants? $f_0 = c_1 + c_2 = 0, c = c_1 = -c_2$

  $f_1 = \frac{c}{2}(1+\sqrt{5} - 1+\sqrt{5}) = c\sqrt{5} = 1$

  $c = \frac{\sqrt{5}}{5}$
- f_n ? $f_n = \frac{\sqrt{5}}{5}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$

What if roots are not all distinct ?

If $r_1, \ldots, r_k$ are k distinct roots with multiplicities m_1 to m_k, respectively, then the general solution is:

$$t_n = \sum_{i=1}^{k} \sum_{j=0}^{m_i - 1} c_{ij} n^j r_i^n$$

Example 7

$$T(n) = \begin{cases} n & \text{if } n = 0, 1 \text{ or } 2 \\ 5t_{n-1} - 8t_{n-2} + 4t_{n-3} & \text{otherwise} \end{cases}$$

The recurrence: $t_n - 5t_{n-1} + 8t_{n-2} - 4t_{n-3} = 0$

Characteristic Polynomial:

$$x^3 - 5x^2 + 8x - 4 = (x - 1)(x - 2)^2$$

Roots: $r_1 = 1$, multiplicity $m_1 = 1$,

$r_2 = 2$, multiplicity $m_2 = 2$.

General solution: $t_n = c_1 1^n + c_2 2^n + c_3 n 2^n$

From initial conditions we get $c_1 = -1$, $c_2 = 2$ and $c_3 = -\frac{1}{2}$

Therefore, $t_n = -1 + 2^{n+1} - n2^{n-1}$

Analysis of Algorithms

- Analysis of algorithms refers to estimating the time and space required to execute the algorithm.
- We shall be mostly concerned with time complexity.
- Problems for which there exist polynomial time algorithms are considered to have an efficient solution.

Analysis of Algorithms

Problems can be:

- Unsolvable

  These problems are so hard that there does not exist an algorithm that can solve them. Example is the famous Halting problem.

- Intractable

  These problems can be solved, but for some instances the time required is exponential and thus these problems are not always solvable in practice, even for small sizes of input.

- Problems of unknown complexity

  These are, for example, NP-complete problems; for these problems neither there is a known polynomial time algorithm, nor they have been shown to be intractable.

- Feasible (tractable) problems

  For these problems there is a known polynomial time algorithm.

Analysis of Algorithms

We shall not try to calculate the exact time needed to execute an algorithm; this would be very difficult to do, and it would depend on the implementation, platform, etc.

We shall only be ***estimating*** the time needed for algorithm execution, as a function of the size of the input.

We shall estimate the running time by counting some "dominant" instructions.

A ***barometer*** instruction is an instruction that is executed at least as often as any other instruction in the algorithm.

Analysis of Algorithms

Worst-case time is the ***maximum*** time needed to execute the algorithm, taken over all inputs of size n.

Average-case time is the ***average*** time needed to execute the algorithm, taken over all inputs of size n.

Best-case time is the ***minimum*** time needed to execute the algorithm, taken over all inputs of size n.

It is usually harder to analyse the average-case behaviour of the algorithm than the best and the worst case.

Behaviour of an algorithm that is the most important depends on the application. For example, for an algorithm that controls a nuclear reactor, the worst case analysis is clearly the most important. On the other hand, for an algorithm that is used in a non-critical application and runs on a variety of different inputs, the average case may be more appropriate.

Example 8. Finding the Maximum Value in an Array Using a While Loop

This algorithm finds the largest number in the array $s[1], s[2], \ldots, s[n]$.

```
Input Parameter: s
Output Parameters: None
array_max_ver1(s)
{
    large = s[1]
    i = 2
    while (i ≤ s.last)
    {
        if (s[i] > large)
            large = s[i]    // larger value found
        i = i + 1
    }
    return large
}
```

- The input is an array of size n.
- What can be used as a ***barometer*** ?
- The number of iterations of while loop appears to be a reasonable estimate of execution time - The loop is always executed $n - 1$ times.
- We'll use the comparison $i \leq s.last$ as a **barometer**.
- The worst-case and the average-case (as well as the best case!) times are the same and equal to n.

Suppose that the worst-case time of an algorithm is (for input of size n) is $t(n) = 60n^2 + 5n + 1$.

For large n, $t(n)$ grows as $60n^2$:

n	$t(n) = 60n^2 + 5n + 1$	$60n^2$
10	6,051	6,000
100	600,501	600,000
1000	60,005,001	60,000,000
10,000	6,000,050,001	6,000,000,000

The constant 60 is actually not important as it does not affect the growth of $t(n)$ with n. We say that $t(n)$ is of order n^2, and we write it as

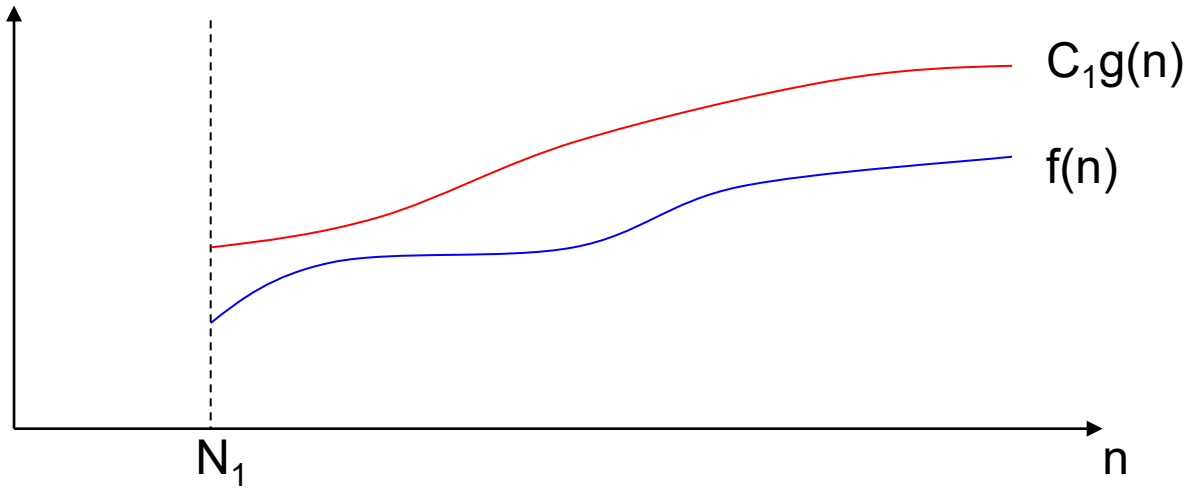
$$t(n) = \Theta(n^2)$$

("$t(n)$ is theta of n^2)

Let f and g be nonnegative functions on the positive integers.

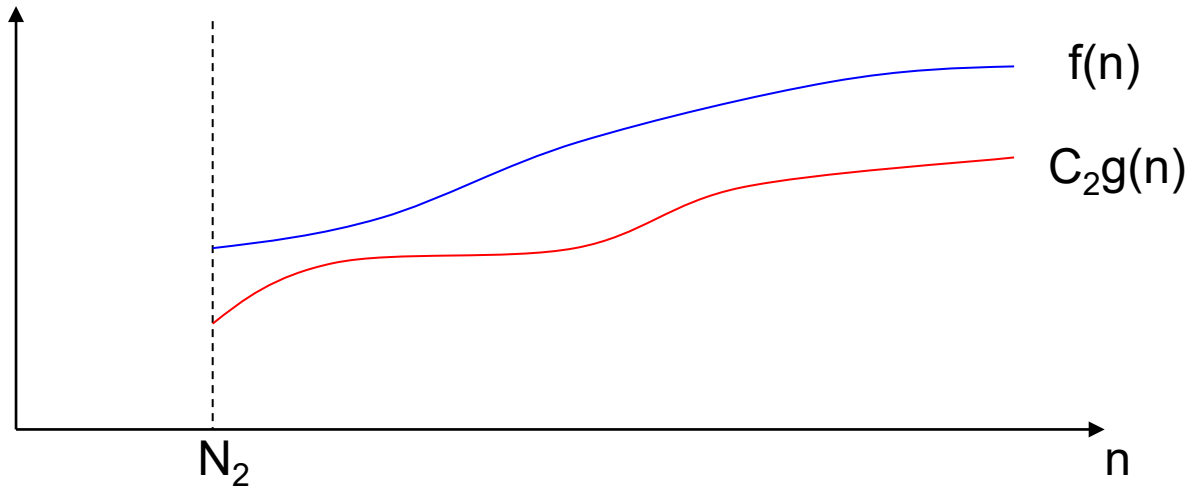
Asymptotic upper bound: $f(n) = O(g(n))$ if there exist constants $C_1 > 0$ and N_1 such that $f(n) \leq C_1 g(n)$ for all $n \geq N_1$.

We read "$f(n) = O(g(n))$" as "$f(n)$ is big oh of $g(n)$" or "$f(n)$ is of order at most $g(n)$".



Asymptotic lower bound: $f(n) = \Omega(g(n))$ if there exist constants $C_2 > 0$ and N_2 such that $f(n) \geq C_2 g(n)$ for all $n \geq N_2$.

We read "$f(n) = \Omega(g(n))$" as "$f(n)$ is of order at least $g(n)$" or "$f(n)$ is omega of $g(n)$".

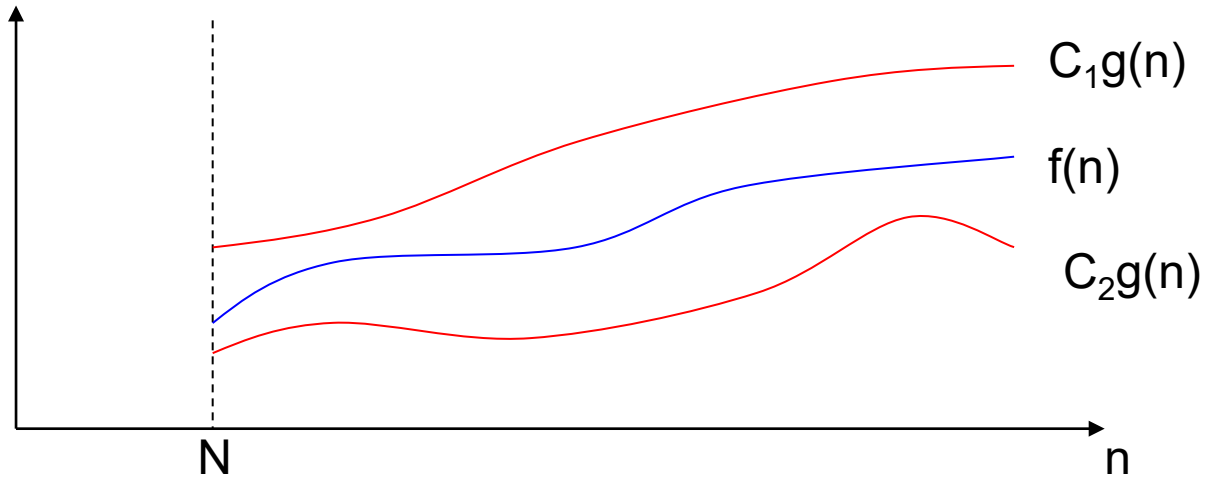


Asymptotic tight bound: $f(n) = \Theta(g(n))$ if there exist constants $C_1, C_2 > 0$ and N such that $C_2 g(n) \leq f(n) \leq C_1 g(n)$ for all $n \geq N$.

Equivalently, $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

We read "$f(n) = \Theta(g(n))$" as "$f(n)$ is of order $g(n)$" or "$f(n)$ is theta of $g(n)$"

Note that $n = O(2^n)$ but $n \neq \Theta(2^n)$.



Example 9

Let us now formally show that if $t(n) = 60n^2 + 5n + 1$ then

$$t(n) = \Theta(n^2)\,.$$

Since $t(n) = 60n^2 + 5n + 1 \leq 60n^2 + 5n^2 + n^2 = 66n^2$ for all $n \geq 1$

it follows that $t(n) = 60n^2 + 5n + 1 = O(n^2)$.

Since $t(n) = 60n^2 + 5n + 1 \geq 60n^2$ for all $n \geq 1$

it follows that $t(n) = 60n^2 + 5n + 1 = \Omega(n^2)$.

Since $t(n) = 60n^2 + 5n + 1 = O(n^2)$ and $t(n) = 60n^2 + 5n + 1 = \Omega(n^2)$ it follows that $t(n) = 60n^2 + 5n + 1 = \Theta(n^2)$.

Asymptotic Bounds for Some Common Functions

- Let $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n^1 + a_0$ be a nonnegative polynomial (that is, $p(n) \geq 0$ for all n) in n of degree k. Then $p(n) = \Theta(n^k)$.

- $\log_b n = \Theta(\log_a n)$ (because $\log_a x = \frac{\log_b x}{\log_b a}$)

- $\log n! = \Theta(n \log n)$
  (because $\log n! = \log n + \log(n-1) + \cdots + \log 2 + \log 1$)

- $\sum_{i=1}^{n} \left(\frac{1}{i}\right) = \Theta(\log n)$

Common Asymptotic Growth Functions

Theta form	Name
$\Theta(1)$	Constant
$\Theta(\log\log n)$	log log
$\Theta(\log n)$	log
$\Theta(n^c), 0 < c < 1$	Sublinear
$\Theta(n)$	Linear
$\Theta(n \log n)$	n log n
$\Theta(n^2)$	Quadratic
$\Theta(n^3)$	Cubic
$\Theta(n^k), k \geq 1$	Polynomial
$\Theta(c^n), c > 1$	Exponential
$\Theta(n!)$	Factorial

The running time of different algorithms on a processor performing one million high level instructions per second (from Algorithm Design, by J. Kleinberg and E Tardos).
For comparison, remember that the **age of the universe is is around 14×10^9 years**

n	n	nlogn	n^2	n^3	1.5^n	2^n	n!
10	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
30	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} y.
50	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	$>10^{25}$ y.
100	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 y.	10^{17} y.	$>10^{25}$ y.
1,000	< 1 sec	< 1 sec	1 sec	18 min	$>10^{25}$ y.	$>10^{25}$ y.	$>10^{25}$ y.
10,000	< 1 sec	< 1 sec	2 min	12 days	$>10^{25}$ y.	$>10^{25}$ y.	$>10^{25}$ y.
100,000	< 1 sec	2 sec	3 hours	32 years	$>10^{25}$ y.	$>10^{25}$ y.	$>10^{25}$ y.
1,000,000	1 sec	20 sec	12 days	31,710 y.	$>10^{25}$ y.	$>10^{25}$ y.	$>10^{25}$ y.

Properties of Big Oh

If $f(x) = O(g(x))$ and $h(x) = O(g'(x))$

then $f(x) + h(x) = O(\max\{ g(x), g'(x) \})$

and $f(x)\, h(x) = O(g(x)\, g'(x))$

If $f(x) = O(g(x))$ and $g(x) = O(h(x))$

then $f(x) = O(h(x))$

Exercises

1. $5n^2 - 1000\,n + 8 = O(n^2)$?
2. $5n^2 - 1000\,n + 8 = O(n^3)$?
3. $5n^2 + 1000\,n + 8 = O(n)$?
4. $n^n = O(2^n)$?
5. If $f(n) = \Theta(g(n))$, then $2^{f(n)} = \Theta(2^{g(n)})$?
6. If $f(n) = O(g(n))$, then $g(n) = \Omega(f(n))$?

Exercises

1. $5n^2 - 1000\,n + 8 = O(n^2)$? True

2. $5n^2 - 1000\,n + 8 = O(n^3)$? True

3. $5n^2 + 1000\,n + 8 = O(n)$? False – n^2 grows faster than n

4. $n^n = O(2^n)$? False – n^n grows faster than 2^n

5. If $f(n) = \Theta(g(n))$, then $2^{f(n)} = \Theta(2^{g(n)})$? False – here constants do matter, as they become exponents !!!
   E.g., $f(n) = 3n$, $g(n) = n$, 2^{3n} □ (2^n)

6. If $f(n) = O(g(n))$, then $g(n) = \Omega(f(n))$? True – if $f(n) \leq Cg(n)$ for $n \geq N$, then $g(n) \geq \frac{1}{C} f(n)$ for $n \geq N$.

Exercises

7. If $f(n) = \Theta(g(n))$, and $g(n) = \Theta(h(n))$, then $f(n) + g(n) = \Theta(h(n))$?

8. If $f(n) = O(g(n))$, then $g(n) = O(f(n))$?

9. If $f(n) = \Theta(g(n))$, then $g(n) = \Theta(f(n))$?

10. $f(n) + g(n) = \Theta(h(n))$ where $h(n) = \max\{f(n), g(n)\}$?

11. $f(n) + g(n) = \Theta(h(n))$ where $h(n) = \min\{f(n), g(n)\}$?

Exercises

7. If $f(n) = \Theta(g(n))$, and $g(n) = \Theta(h(n))$, then $f(n) + g(n) = \Theta(h(n))$? True

8. If $f(n) = O(g(n))$, then $g(n) = O(f(n))$? False - e.g., $f(n) = n$, $g(n) = n^2$.

9. If $f(n) = \Theta(g(n))$, then $g(n) = \Theta(f(n))$? True

10. $f(n) + g(n) = \Theta(h(n))$ where $h(n) = \max\{f(n), g(n)\}$? True

11. $f(n) + g(n) = \Theta(h(n))$ where $h(n) = \min\{f(n), g(n)\}$?
False - $f(n) = n$, $g(n) = n^2$, $h(n) \neq n$

Exercises – find theta for the following

12. $6n + 1$

13. $2 \lg n + 4n + 3n \lg n$

14. $\frac{(n^2 + \lg n)(n + 1)}{n + n^2}$

15.
```
for i=1 to 2n
    for j=1 to n
            x=x+1
```

16.
```
i=n
while (i≥1) {
    x=x+1
    i=i/2
}
```

17.
```
j=n
while (j≥1) {
    for i=1 to j
            x=x+1
    j=j/2
}
```

Exercises – find theta for the following

12. $6n + 1 = \Theta(n)$

13. $2 \lg n + 4n + 3n \lg n = \Theta(n \log n)$

14. $\frac{(n^2 + \lg n)(n+1)}{n + n^2} = \Theta(n)$

15.
```
for i=1 to 2n
    for j=1 to n
            x=x+1
```
$\Theta(n^2)$

12.
```
i=n
while (i≥1) {
    x=x+1
    i=i/2
}
```
$\Theta(\log n)$

17.
```
j=n
while (j≥1) {
    for i=1 to j
            x=x+1
    j=j/2
}
```
$\Theta(n)$