Instituto Politécnico Nacional

Escuela Superior de Cómputo

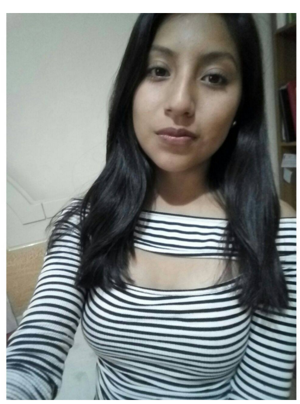
Ejercicio 05 - Análisis de Algoritmos recursivos

Unidad de aprendizaje: Análisis de algoritmos

Grupo: 3CM3

Alumno(a):
Nicolás Sayago Abigail

Profesor(a):
Edgardo Adrian Franco



23 Octubre de 2018

Índice

1. Ejercicio 1

Calcula la cota de complejidad para el algoritmos de la siguiente función recursiva.

```
int FuncionRecursiva(int num)
{
    int i, resultado;
    if(num == 0)
        return 1;
    else if(num < 2)
    {
        resultado = 0;
        for(i = 0; i < num * num; i++)
            resultado *= num;
        return resultado;
    }
    else
        return FuncionRecursiva(num - 1) * FuncionRecursiva(num - 2);
}
```

Obteniendo casos base:

$$T(0) = 1 \text{ Retorno}$$
$$T(1) = 3 \text{ Aritmética y el retorno}$$
$$T(2) = 2 + T(n-1) + T(n-2) = 2 + 1 + 3 = 6$$
$$T(n) = 1 + T(n-1) + T(n-2)$$

Es una recurrencia lineal no homogénea. Reacomodando términos:

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

Generando ecuación característica:

$$\left(x^2 - x - 1\right)(x-1) = 0$$

Sus raíces son:

$$r_1 = 1$$
$$r_2 = \frac{1+\sqrt{5}}{2}$$
$$r_3 = \frac{1-\sqrt{5}}{2}$$

Sustituyendo:

$$T(n) = C_1(1)^n + C_2\left(\frac{1+\sqrt{5}}{2}\right)^n + C_3\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Obtenemos C_1 y C_2 con los casos base:

$$T(0) = C_1(1)^n + C_2\left(\frac{1+\sqrt{5}}{2}\right)^n + C_3\left(\frac{1-\sqrt{5}}{2}\right)^n = C_1 + C_2 + C_3 = 1$$

$$T(1) = C_1 + C_2\left(\frac{1+\sqrt{5}}{2}\right) + C_3\left(\frac{1-\sqrt{5}}{2}\right) = 3$$

$$T(2) = C_1 + C_2\left(\frac{1+\sqrt{5}}{2}\right)^2 + C_3\left(\frac{1-\sqrt{5}}{2}\right)^2 = 5$$

Resolviendo el sistema de ecuaciones, tenemos que:

$$C_1 = -1$$

$$C_2 = -0.35$$

$$C_3 = 2.34$$

Sustituimos en nuestra ecuación anterior:

$$T(n) = (-1)(1)^n - 0.35\left(\frac{1+\sqrt{5}}{2}\right)^n + 2.34\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Por lo tanto

$$O\left(\frac{1-\sqrt{5}}{2}\right)^n = O\left(1.61\right)^n$$

2. Ejercicio 2

Calcular la complejidad de la implementación recursiva del producto.

```
int Producto(int a, int b)
{
    if(b == 0)
        return 0;
    else
        return a * Producto(a, b-1);
}
```

Obteniendo casos base:

$$T(0) = 1 \text{ Retorno}$$
$$T(1) = 1 + 1 = 2 \text{ Aritmética y recursion}$$
$$T(n) = 1 + T(n-1)$$

Es una recurrencia lineal no homogénea. Reacomodando términos:

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

Generando ecuación característica:

$$(x-1)(x-1) = 0$$

Sus raíces son:

$$r_1 = 1$$
$$r_2 = 1$$

Sustituyendo:

$$T(n) = C_1(1)^n + C_2 n(1)^n$$

Obtenemos C_1 y C_2 con los casos base:

$$T(0) = C_1(1)^0 + C_2(0)(1)^0 = C_1 = 1$$
$$T(1) = C_1(1)^1 + C_2(1)(1)^1 = C_1 + C_2 = 2$$

Resolviendo el sistema de ecuaciones, tenemos que:

$$C_1 = 1$$

$$C_2 = 1$$

Sustituimos en nuestra ecuación anterior:

$$T(n) = (1)(1)^n + (1)n(1)^n = 1 + n$$

Por lo tanto

$$O(n)$$

3. Ejercicio 3

Calcular el costo de un recorrido In-orden en un Árbol Binario completamente lleno.

```
// Recorrido IN-ORDEN de un Arbol binario completamente lleno
void TraveerseInorden(TreeNode root)
{
    if(root != null)
    {
        TraveerseInorden(root.getLeft());
        process(root.getValue());
        TraveerseInorden(root.getRight());
    }
```

En la figura resaltamos que en el caso n, se hacen dos llamadas recursivas que envían el lado derecho del árbol y en otra el lado izquierdo, debido a que tenemos un árbol completamente lleno decimos que es $\frac{n}{2}$.

Obteniendo casos base:

$$T(0) = 0 \text{ Retorno}$$

$$T(n) = 1 + 2T\left(\frac{n}{2}\right)$$

Es una recurrencia no lineal, en estos casos usamos el teorema maestro.. Reacomodando términos:

$$T(n) = 1 + 2T\left(\frac{n}{2}\right)$$

Identificamos que:

$$a = 2$$

$$b = 2$$

$$f(n) = 1$$

Sustituimos en el caso 1:

$$f(n) = O(n^{log_b^a - \epsilon}) = O(n^{log_2^2 - \epsilon}) = O(n^{1-\epsilon}) = O(n^{1-1}) = f(n)$$

En este caso se cumple la condición, por lo tanto:

$$T(n) = \Theta\left(n^{log_2^2}\right) = \Theta(n)$$

4. Ejercicio 4

Calcular la cota de complejidad del algoritmo de búsqueda ternaria.

```
double BusquedaTernaria(double f[], int l, int r, double absolutePrecision)
{
    if(r-1 <= absolutePrecision)
        return (i + r) / 2.0;
    else
    {
        int m1 = (2 * l + r) / 3;
        int m2 = (l + 2 * r) / 3;

        if(f[m1] < f[m2])
            return BusquedaTernaria(f, m1, r, absolutePrecision);
        else
            return BusquedaTernaria(f, l, m2, absolutePrecision);
    }
}
```

Obteniendo casos base:

$$T(0) = 1 \text{ Comparación}$$

$$T(n) = 1 + T\left(\frac{2n}{3}\right)$$

Es una recurrencia no lineal, en estos casos usamos el teorema maestro.. Reacomodando términos:

$$T(n) = 1 + 2T\left(\frac{n}{2}\right)$$

Identificamos que:

$$a = 1$$

$$b = \frac{3}{2}$$

$$f(n) = 1$$

Sustituimos en el caso 1:

$$f(n) = O(n^{log_b^a - \epsilon}) = O(n^{log_{\frac{3}{2}}^{1} - \epsilon}) = O(n^{0-\epsilon}) = O(n^0)$$

Descartamos el caso, puesto que ningún ϵ cumple la condición.

Sustituimos en el caso 2:

$$f(n) = O(n^{log_b^a}) = O(n^{log_{\frac{3}{2}}^1}) = O(n^0) = O(1) = f(n)$$

Como cumple las condiciones de este caso, tenemos que:

$$T(n) = \Theta\left(n^{log_{\frac{3}{2}}^1} logn\right) = \Theta\left(logn\right)$$

5. Ejercicio 5

Calcular la cota de complejidad del algoritmo de ordenamiento QuickSort.

```
// Ordenamiento QuickSort
QuickSort(lista, inf, sup)
{
    elem_div = lista[sup];
    i = inf - 1;
    j = sup;
    cont = 1;

    if(inf >= sup)
        return;
    while(cont)
    {
        while(lista[i++] < elem_div);
        while(lista[--j] > elem_div);
        if(i < j)
        {
            temp = lista[i];
            lista[i] = lista[j];
            lista[j] = temp;
        }
        else
            cont = 0;
    }
    temp = lista[i];
    lista[i] = lista[sup];
    lista[sup] = temp;

    QuickSort(lista, inf, i - 1);
    QuickSort(lista, i + 1, sup);
}
```

Obteniendo casos base:

$$T(0) = 0$$
$$T(1) = 1 \text{ Comparación}$$
$$T(2) = 2$$
$$T(n) = n + T(n-1)$$

Es una recurrencia lineal no homogénea. Reacomodando términos:

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

Generando ecuación característica:

$$(x-1)(x-1)^2 = 0$$

Sus raíces son:

$$r_1 = 1$$

$$r_2 = 1$$

$$r_3 = 1$$

Sustituyendo:

$$T(n) = C_1(1)^n + C_2 n(1)^n + C_3 n^2 1^n = C_1 + C_2 n + C_3 n^2$$

Obtenemos C_1 , C_2 Y C_3 con los casos base:

$$T(0) = C_1 = 0$$
$$T(1) = 0 + C_2 + C_3 = 1$$
$$T(2) = 0 + 2C_2 + 4C_3 = 2$$

Resolviendo el sistema de ecuaciones, tenemos que:

$$C_1 = 0$$

$$C_2 = \frac{1}{2}$$

$$C_3 = \frac{1}{2}$$

Sustituimos en nuestra ecuación anterior:

$$T(n) = \left(\frac{1}{2}\right) n + \left(\frac{1}{2}\right) n^2$$

Por lo tanto

$$O\left(n^2\right)$$

6. Ejercicio 6

Resolver las siguientes ecuaciones y dar su orden de complejidad.

- $T(n) = 3T(n-1) + 4T(n-2) \Rightarrow n > 1; T(0) = 0; T(1) = 1$

  Es una recurrencia homogénea. Reacomodando los términos:

  $$T(n) - 3T(n-1) - 4T(n-2) = 0$$

  Sustituyendo por x, obtenemos una ecuación a resolver:

  $$x^2 - 3x^1 - 4 = 0$$

  Obteniendo las raíces:

  $$\begin{aligned} x^2 - 3x^1 - 4 &= 0 \\ (x-4)(x+1) &= 0 \end{aligned}$$

  Las raices son:

  $$r_1 = 4$$

  $$r_2 = -1$$

  La ecuación con recurrencia es:

  $$\begin{gathered} C_1 r_1^n + C_2 r_2^n \\ C_1 4^n + C_2 (-1)^n \end{gathered}$$

  Para encontrar C_1 y C_2, tomamos los casos base $T(0) = 0; T(1) = 1$:

  $$\begin{aligned} T(0) &= C_1 4^0 + C_2(-1)^0 = 0 \\ &= C_1 + C_2 = 0 \\ T(1) &= C_1 4^1 + C_2(-1)^1 = 1 \\ &= 4C_1 - C_2 = 1 \end{aligned}$$

  Tenemos el siguiente sistema de ecuaciones:

  $$\begin{aligned} C_1 + C_2 &= 0 \\ 4C_1 - C_2 &= 1 \end{aligned}$$

  Resolviendo:

  $$\begin{aligned} C_1 &= -C_2 \\ 4(-C_2) - C_2 = -5C_2 &= 1 \end{aligned}$$

Los valores de las constantes son:

$$C_1 = \frac{1}{5}$$

$$C_2 = -\frac{1}{5}$$

Sustituyendo:

$$C_1 4^n + C_2(-1)^n$$
$$\frac{1}{5}4^n - \frac{1}{5}(-1)^n$$

Su orden de complejidad es:

$$O(4^2)$$

- $T(n) - 2T(n-1) = 3^n \Rightarrow n \geq 2; T(0) = 0, T(1) = 1$ Es una recurrencia lineal no homogénea. Reacomodando términos:

$$T(n) - 2T(n-1) = 3^n$$

Generando ecuación característica:

$$(x-2)\,(x-3^x)^2 = 0$$

Sus raíces son:

$$r_1 = 2$$

Sustituyendo:

$$T(n) = C_1(2)^n + 3^n$$

Obtenemos C_1 con los casos base, resolviendo el sistema de ecuaciones, tenemos que:

$$C_1 = -1$$

Sustituimos en nuestra ecuación anterior:

$$T(n) = -(2)^n + 3^n$$

Por lo tanto

$$O\left(3^n\right)$$

7. Ejercicio 7

Calcular la cota de complejidad que tendrían los algoritmos con los siguientes modelos recurrentes.

- $T(n) = T(\frac{n}{3}) + 4T(\frac{n}{2}) + 2n^2 + n$ Usando el Teorema Maestro:

  Dividimos el problema y primero tenemos:

  $$T(n) = T(\frac{n}{3}) + 2n^2$$

  Identificamos que:

  $$a = 1$$
  $$b = 3$$
  $$f(n) = 2n^2$$

  Sustituimos en el caso 1:

  $$f(n) = O(n^{log_b^a - \epsilon}) = O(n^{log_3^1 - \epsilon}) = O(n^{0-\epsilon})$$

  Observamos que en ese caso no funcionaria con algún $\epsilon > 0$. En el caso 2 no se prueba puesto que se puede ignorar. Probamos el caso 3:

  $$f(n) = \Omega(n^{log_b^a + \epsilon}) = \Omega(n^{log_3^1 + \epsilon}) = \Omega(n^{0+\epsilon}) = \Omega(n^{0+2}) = \Omega(n^2)$$

  Con $\epsilon = 2$ se cumple el caso 3. Por lo tanto:

  $$\Theta\left(n^2\right)$$

- $T(n) = T(\frac{n}{2} + 2T(\frac{n}{4}) + 2)$ Usando el teorema maestro:

  $$T(n1) = T(\frac{n}{2}) + 1$$

  Identificamos que:

  $$a = 1$$
  $$b = 2$$
  $$c = 0$$
  $$f(n) = 1$$

  El caso uno no cumple con las condiciones puesto que no hay ϵ , entonces probamos con el caso 2.

  $$log_2(1) == 0 -> verdadero$$

Por lo tanto

$$T(n1) = \theta(log(n))$$

La siguiente parte a analizar es:

$$T(n2) = 2T(\frac{n}{4} + 1)$$

Identificamos que:

$$a = 2$$
$$b = 4$$
$$c = 0$$
$$f(n) = 1$$

Se descarta el caso 1 y 2, usando caso 3:

- Primera condición

$$log_4(2) = \frac{1}{2} < c$$

- Segunda condición

$$2 * \frac{1}{4} \leq k * 1, k = \frac{1}{2}$$

Ambas condiciones se cumplen, por lo tanto:

$$T(n2) = \theta(1)$$

Juntando ambos análisis, la función complejidad queda de la siguiente manera:

$$T(n) = \theta(log(n) + 1)$$

- $T(n) = 2T(\frac{n}{2}) + 4T(\frac{n}{4}) + 10n^2 + 5n$ Teniendo que:

$$T(n1) = 2T(\frac{n}{2}) + 5n$$

Identificamos:

$$a = 2$$
$$b = 2$$
$$c = 1$$
$$f(n) = 5n$$

Se descarta el caso 1, usando caso 2:

$$T(n1) = \theta(nlog(n))$$

Analizamos la segunda parte:

$$T(n2) = 2T(\frac{n}{4}) + 1$$

$$a = 4$$

$$b = 4$$

$$c = 2$$

$$f(n) = 10n^2$$

Verificamos las condiciones:

- Primera condición

$$log_4(4) = 1 < c$$

- Segunda condición

$$4 * \frac{10n^2}{4} \leq k * 10n^2, k = 1$$

Ambas condiciones se cumplen, por lo tanto:

$$T(n2) = \theta(10n^2)$$

Juntando ambos análisis, la función complejidad queda de la siguiente manera:

$$T(n) = \theta(nlog(n) + 10n^2)$$