Instituto Politécnico Nacional

Escuela Superior de Cómputo

Análisis de Algoritmos

Profesor: Edgardo Adrián Franco Martínez

Alumno: Ortega Victoriano Ivan

Fecha: 17 de Mayo 2017

Análisis de algoritmos recursivos.
Ejercicio 1: Calcular la complejidad para el siguiente algoritmo.

```
#include <stdio.h>
#include <stdlib.h>

int FuncionRecursiva(int num )
{
  if ( num == 0 )
    return 1;
  else if ( num < 3 )
  {
    int resultado=0;
    for(int i=0;i<num*num;i++)
      resultado*=num;
    return resultado;
  }
  else
    return FuncionRecursiva( num - 1 )*
      FuncionRecursiva(num - 2)/FuncionRecursiva(num - 3);
}
```

Sol: Del algoritmo podemos ver que si $n = 0$, entonces $T(0) = 1$, además si $n = 1$ entonces $T(1) = 5$ (comparar 1 vez el *if* de la función, 1 asignación de la variable resultado, 1 ejecución del for, 2 comparaciones del for), y por último si $n = 2$ entonces $T(2) = 9$ (se analiza igual que el caso de $n = 1$). Los análisis anteriores son para los costos de los casos base.
Ahora, analizando para un n en general, tendremos que:

$$T(n) = T(n-1) + c_{mult} + T(n-2) + c_{div} + T(n-3)$$

Donde $c_{mult} = c_{div} = 1$, luego:

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

Que se trata de una ecuación en recurrencia no homogénea.
Reacomodando la ecuación, tenemos:

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

Haciendo el cambio $T(n) = x^3$, $b = 2$ y $d = 0$, obtenemos la ecuación característica:

$$(x^3 - x^2 - x - 1)(x - 2) = 0$$

De donde las raíces son:

$$r_1 = 1.84$$
$$r_2 = -0.42 + i0.61$$
$$r_3 = -0.42 - i0.61$$
$$r_4 = 2$$

Que son raíces distintas, así:

$$T(n) = c_1(1.84)^n + c_2(-0.42 + i0.61)^n + c_3(-0.42 - i0.61)^n + c_4 2^n$$

Dado que tenemos dos números complejos, de acuerdo con sus propiedades, la ecuación puede reescribirse de la siguiente manera:

$$T(n) = c_1(1.84)^n + c_2(0.73^n(\cos(124.68) + i\sin(124.68)))$$
$$+c_3(0.73^n(\cos(124.68n) - i\sin(124.68n))) + c_4 2^n$$

Donde 0.73 y 124.68 representan la magnitud del vector y su ángulo respectivamente. Luego

$$T(n) = c_1(1.84)^n + 0.73^n((c_2+c_3)\cos(124.68n) + (c_2-c_3)i\sin(124.68n))) + c_4 2^n$$

Llamemos $k_1 = c_2 + c_3$ y $k_2 = (c_2 - c_3)i$, así

$$T(n) = c_1(1.84)^n + 0.73^n(k_1\cos(124.68n) + k_2\sin(124.68n))) + c_4 2^n$$

Evaluando las condiciones iniciales

$$T(0) = c_1 + k_1 + c_4$$
$$T(1) = 1.84c_1 - 0.41k_1 + 0.60k_2 + 2c_4$$
$$T(2) = 3.38c_1 - 0.18k_1 - 0.49k_2 + 4c_4$$

Como $T(3) = T(2) + T(1) + T(0) + 2$ entonces:

$$1 + 5 + 9 + 2 = 6.22c_1 + 0.37k_1 + 0.09k_2 + 8c_4$$

Luego

$$1 = c_1 + k_1 + c_4$$
$$5 = 1.84c_1 - 0.41k_1 + 0.60k_2 + 2c_4$$
$$9 = 3.38c_1 - 0.18k_1 - 0.49k_2 + 4c_4$$
$$17 = 6.22c_1 + 0.37k_1 + 0.09k_2 + 8c_4$$

Que es el sistema de ecuaciones resultante de la evaluacion de las condiciones iniciales.
Pasando este sistema a la matriz aumentada, tenemos:

$$\left(\begin{array}{cccc|c} 1 & 1 & 0 & 1 & 1 \\ 1.84 & -0.41 & 0.60 & 2 & 5 \\ 3.38 & -0.18 & -0.49 & 4 & 9 \\ 6.22 & 0.37 & 0.09 & 8 & 17 \end{array}\right)$$

Resolviendo la matriz por Gauss-Jordan, se llega a la solución:

$$\left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 0.296 \\ 0 & 1 & 0 & 0 & -1.247 \\ 0 & 0 & 1 & 0 & 0.066 \\ 0 & 0 & 0 & 1 & 1.951 \end{array}\right)$$

Que es equivalente a que

$$c_1 = 0.296$$

$$k_1 = -1.247$$

$$k_2 = 0.066$$

$$c_4 = 1.951$$

$$\therefore T(n) = 0.296(1.84)^n + 0.73^n[(0.066)\sin(n\theta) - (1.247)\cos(n\theta)] + (1.951)2^n$$

En donde $\theta = 124.68$

Comentario: Al parecer hay unos cuantos errores en cuanto a lo que hace el algoritmo, de hecho, en la parte donde $n < 3$, al hacer a $resultado = 0$, este valor nunca va a cambiar, ya que en el loop *for* se hace la operación $resultado* = num$, pero como $resultado = 0$, siempre va a retornar un valor de cero. De lo que a partir de un $n > 3$ va a haber divisiones entre 0, lo cual será un error en el programa.

Ejercicio 2: Calcular la complejidad de la implementación recursiva del termino n de la serie de Tribonacci (0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, 274, 504, 927, 1705, ...)

```
#include <stdio.h>
#include <stdlib.h>

int Tribonacci( int num )
{
  if (num==0)
    return 0;
  else if (num==1||num==2)
    return 1;
  else
    return Tribonacci(num-1)+Tribonacci(num-2)+Tribonacci(num-3);
}
```

Sol: Si nos damos cuenta, el análisis es similar al caso anterior, solo hay ligeras modificaciones en cuanto a las condiciones iniciales y a la ecuacion en recurrencia.
Es evidente que $T(0) = 1$, $T(1) = T(2) = 2$, ya que son las comparaciones que se hacen al entrar a la función.
Ahora para un n en general:

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

Donde el 4 resulta de las dos comparaciones de la función, mas el costo de las 2 sumas.
Avanzando un poco más (ya que el análisis es exactamente igual al del ejercicio anterior), la ecuación característica quedaría de la siguiente manera:

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

Donde nuestras raíces ahora son:

$$r_1 = 1.84$$

$$r_2 = -0.42 + i0.61$$

$$r_3 = -0.42 - i0.61$$

$$r_4 = 4$$

Siguiendo el análisis del algoritmo anterior, llegaremos a que la complejidad de la función, estará dada por:

$$T(n) = c_1(1.84)^n + 0.73^n(k_1\cos(124.68n) + k_2\sin(124.68n))) + c_4 4^n$$

Resolviendo las condiciones iniciales:

$$T(0) = c_1 + k_1 + c_4$$
$$T(1) = 1.84c_1 - 0.41k_1 + 0.60k_2 + 4c_4$$
$$T(2) = 3.38c_1 - 0.18k_1 - 0.49k_2 + 16c_4$$

Como $T(3) = T(2) + T(1) + T(0) + 4$ entonces:

$$1 + 2 + 2 + 4 = 6.22c_1 + 0.37k_1 + 0.09k_2 + 32c_4$$

Luego

$$1 = c_1 + k_1 + c_4$$
$$2 = 1.84c_1 - 0.41k_1 + 0.60k_2 + 4c_4$$
$$2 = 3.38c_1 - 0.18k_1 - 0.49k_2 + 16c_4$$
$$9 = 6.22c_1 + 0.37k_1 + 0.09k_2 + 32c_4$$

Pasando este sistema a la matriz aumentada, tenemos:

$$\left(\begin{array}{cccc|c} 1 & 1 & 0 & 1 & 1 \\ 1.84 & -0.41 & 0.60 & 4 & 2 \\ 3.38 & -0.18 & -0.49 & 16 & 2 \\ 6.22 & 0.37 & 0.09 & 32 & 9 \end{array}\right)$$

Resolviendo la matriz por Gauss-Jordan, se llega a la solución:

$$\left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & -0.604 \\ 0 & 1 & 0 & 0 & 1.229 \\ 0 & 0 & 1 & 0 & 3.529 \\ 0 & 0 & 0 & 1 & 0.374 \end{array}\right)$$

Que es equivalente a que

$$c_1 = -0.604$$
$$k_1 = 1.229$$
$$k_2 = 3.529$$
$$c_4 = 0.374$$

$$\therefore T(n) = -0.604(1.84)^n + 0.73^n[(1.229)\cos(n\theta) + (3.529)\sin(n\theta)] + (0.374)2^n$$
$$\text{Con } \theta = 124.68$$

Ejercicio 3: Resolver las siguientes ecuaciones y dar su orden de complejidad:

1) $T(n) = 3T(n-1) + 4T(n-2)$ si $n > 1$; $T(0) = 0$, $T(1) = 1$

2) $T(n) = 2T(n-1) - (n+5)3^n$ si $n > 0$; $T(0) = 0$

3) $T(n) = 3T(n-1) + 4T(n-2) + (n+5)2^n$ si $n > 1$; $T(0) = 0$, $T(1) = 100$

4) $T(n) - 2T(n-1) = 3^n$ si $n \geq 2$; $T(0) = 0$, $T(1) = 1$

1) Sol: Empezamos reescribiendo la ecuación, teniendo así:

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

Pasando a su ecuación característica, tenemos:

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

Luego

$$(x-4)(x+1) = 0$$

Que es una ecuación en recurrencia homogénea, cuyas raíces son:

$$r_1 = 4$$

$$r_2 = -1$$

Que son raíces distintas, luego así:

$$T(n) = c_1 4^n + c_2(-1)^n$$

Evaluando las condiciones iniciales, como $T(0) = 0$ y $T(1) = 1$, entonces

$$0 = c_1 - c_2$$

$$1 = 4c_1 - c_2$$

Resolviendo el sistema de ecuaciones llegamos a que $c_1 = \frac{1}{5}$ y $c_2 = -\frac{1}{5}$. Luego

$$T(n) = \frac{1}{5}4^n - \frac{1}{5}(-1)^n$$

$$\therefore T(n) = \tfrac{1}{5}[4^n + (-1)^{n+1}]$$

2) Sol: Reescribiendo la ecuación, tenemos que:

$$T(n) - 2T(n-1) = 3^n(n+5)$$

Que es una ecuación en recurrencia no homogénea con $b = 3$ y $d = 1$, así entonces, obteniendo su ecuación característica:

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

Teniendo así las raíces $r_1 = 2$ y $r_2 = r_3 = 3$. Así entonces:

$$T(n) = c_1 2^n + c_2 3^n + n c_3 3^n$$

Evaluando las condiciones iniciales, como $T(0) = 0$, entonces:

$$0 = c_1 + c_2$$

Además $T(1) = 2T(0) + (1+5)3^1 = 18$, entonces:

$$18 = 2c_1 + 3c_2 + 3c_3$$

Para obtener un sistema de ecuaciones de 3x3, evaluemos $T(2)$, donde $T(2) = 2T(1) + (2+5)3^2 = 36 + 63 = 99$. Luego entonces:

$$99 = 4c_1 + 9c_2 + 18c_3$$

Teniendo así el siguiente sistema de ecuaciones:

$$c_1 + c_2 = 0$$

$$2c_1 + 3c_2 + 3c_3 = 18$$

$$4c_1 + 9c_2 + 18c_3 = 99$$

Así,resolviendo el sistema de ecuaciones, obtenemos que

$$c_1 = -9,\ c_2 = 9,\ c_3 = 3$$

Sustituyendo los valores, tendremos que:

$$T(n) = -9(2^n) + 3^{n+2} + 3^{n+1}n$$

Simplificando la ecuación, tendremos:

$$T(n) = 3^{n+1}(n+3) - 9(2^n)$$

Que es la complejidad de la ecuación.

Comentario: La ecuación original $T(n) = 2T(n-1) - (n+5)3^n$ se me hizo extraña desde que el signo de el polinomio al que es igual la ecuación, es negativo. De hecho, al seguir el análisis como lo hice en lo que resolví, al evaluar $T(1)$ se obtendrían tiempos negativos. De hecho, se me hizo raro el hecho de que se restaran tiempos en la complejidad, por ello recurrí a cambiar el signo de la ecuación, para obtener resultados que creo yo, son los adecuados.

3) Sol: Rescribimos la ecuación original:

$$T(n) - T(n-1) - 4T(n-2) = (n+5)2^n$$

Que es una ecuación en recurrrencia no homogénea. Obteniendo su ecuación característica:

$$(x^2 - x - 4)(x-2)^2 = 0$$

Simplificando:

$$(x + 1 + \sqrt{5})(x + 1 - \sqrt{5})(x-2)^2 = 0$$

Donde las raíces de la ecuación son:

$$r_1 = 1 - \sqrt{5}$$

$$r_2 = 1 + \sqrt{5}$$

$$r_3 = r_4 = 2$$

Donde tenemos 2 raíces iguales, ahora bien:

$$T(n) = c_1(1 - \sqrt{5})^n + c_2(1 + \sqrt{5})^n + c_3 2^n + c_4 n 2^n$$

Evaluando las condiciones iniciales de la ecuación, tenemos que $T(0) = 0$ y $T(1) = 100$, entonces:

$$0 = c_1 + c_2 + c_3$$

$$100 = c_1(1 - \sqrt{5}) + c_2(1 + \sqrt{5}) + 2c_3 + 2c_4$$

Para obtener un sistema de ecuaciones de 4x4, como $T(2) = T(1) + 4T(0) + (2+5)2^2$, además $T(3) = T(2) + 4T(1) + (3+5)2^3$. Entonces, para $T(2)$:

$$100 + 4(0) + 28 = c_1(1 - \sqrt{5})^2 + c_2(1 + \sqrt{5})^2 + 4c_3 + 8c_4$$

Luego

$$128 = c_1(1-\sqrt{5})^2 + c_2(1+\sqrt{5})^2 + 4c_3 + 8c_4$$

Para $T(3)$:

$$128 + 4(100) + 64 = c_1(1-5\sqrt{5})^3 + c_2(1+1\sqrt{5})^3 + 8c_3 + 12c_4$$

Luego

$$592 = c_1(1-5\sqrt{5})^3 + c_2(1+1\sqrt{5})^3 + 8c_3 + 12c_4$$

Teniendo así finalmente nuestro sistema de ecuaciones de 4x4:

$$0 = c_1 + c_2 + c_3$$

$$100 = c_1(1-\sqrt{5}) + c_2(1+\sqrt{5}) + 2c_3 + 2c_4$$
$$128 = c_1(1-\sqrt{5})^2 + c_2(1+\sqrt{5})^2 + 4c_3 + 8c_4$$
$$592 = c_1(1-5\sqrt{5})^3 + c_2(1+1\sqrt{5})^3 + 8c_3 + 12c_4$$

De lo cuál pasandolo a su forma matricial, tenemos:

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 0 & 0 \\ (1-\sqrt{5}) & (1+\sqrt{5}) & 2 & 2 & 100 \\ (1-\sqrt{5})^2 & (1+\sqrt{5})^2 & 4 & 8 & 128 \\ (1-\sqrt{5})^3 & (1+\sqrt{5})^3 & 8 & 12 & 592 \end{array}\right)$$

Aplicando Gauss-Jordan a la matriz, llegamos a que:

$$\left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & \frac{160-134\sqrt{5}}{5} \\ 0 & 1 & 0 & 0 & \frac{160+134\sqrt{5}}{5} \\ 0 & 0 & 1 & 0 & -64 \\ 0 & 0 & 0 & 1 & -52 \end{array}\right)$$

Que es equivalente a decir que:

$$c_1 = \frac{160-134\sqrt{5}}{5}$$

$$c_2 = \frac{160+134\sqrt{5}}{5}$$

$$c_3 = -64$$

$$c_4 = -52$$

Así

$$T(n) = (\frac{160 - 134\sqrt{5}}{5})(1-\sqrt{5})^n + (\frac{160 + 134\sqrt{5}}{5})(1+\sqrt{5})^n - 64(2^n) - 52n(2^n)$$

Finalmente

$$T(n) = (\frac{160 - 134\sqrt{5}}{5})(1-\sqrt{5})^n + (\frac{160 + 134\sqrt{5}}{5})(1+\sqrt{5})^n - 2^{n+6}) - 52n(2^n)$$

4) Sol: De la ecuación en recurrencia no homogénea $T(n) - 2T(n-1) = 3^n$ obtenemos su ecuación característica:

$$(x-2)(x-3) = 0$$

Donde sus raíces son:

$$r_1 = 2$$

$$r_2 = 3$$

Teniendo así que

$$T(n) = c_1 2^n + c_2 3^n$$

Evaluando las condiciones iniciales, tenemos que $T(0) = 0$ y $T(1) = 1$, luego entonces:

$$0 = c_1 + c_2$$

$$1 = 2c_1 + 3c_2$$

Resolviendo el sistema de ecuaciones, llegamos a que $c_1 = -1$ y $c_2 = 1$.

$$\therefore T(n) = 3^n - 2^n$$

Ejercicio 4: Calcular la cota de complejidad del algoritmo de búsqueda binaria recursiva:

```
int BusquedaBinaria(int num_buscado, int numeros[],
  int inicio, int centro, int final)
{
  if (inicio>final)
    return -1;
  else if (num_buscado == numeros[centro])
    return centro;
  else if (num_buscado < numeros[centro])
    return BusquedaBinaria(num_buscado,numeros,inicio,
      (int)((inicio+centro-1)/2),centro-1);
  else
    return BusquedaBinaria(num_buscado,numeros,centro+1,
      (int)((final+centro+1)/2),final);
}
```

Sol: Para obtener la ecuación en recurrencia, hay que ver como funciona el algoritmo, es fácil ver que el costo de cada caso base es de 1, sin embargo, para buscar un número que no necesariamente se encuentre a la primera, o que bien, no cumpla las primeras dos condiciones del algoritmo, siempre va a ejecutarlas y va a entrar a alguna de las últimas dos condiciones, donde nuestro arreglo de tamaño n va a ser particionado a la mitad, es decir $n/2$, así hasta que lo encuentre, o no. De lo que podemos decir que su ecuación en recurrencia puede formarse de la siguiente manera:

$$T(n) = 2 + T(n/2)$$

Donde 2 representa los casos base, y $T(n/2)$ representa el costo de la llamada a la función para $n/2$. Ahora bien, de acuerdo al teorema maestro, tenemos $a = 1$, $b = 2$ y $c = 0$, dado que se cumple que $a = b^c$, entonces diremos que

$$T(n) = \Theta(n^c \log n)$$

$$\therefore T(n) = \Theta(\log n)$$

Ejercicio 5: Calcular la cota de complejidad del algoritmo Merge-Sort recursivo:

```
Merge-Sort(a, p, r)
{
  if ( p < r )
  {
    q = parteEntera((p+r)/2);
    Merge-Sort(a, p, q);
    Merge-Sort(a, q+1,r);
    Merge(a, p, q, r);
  }
}
```

Sol: Para la solución de este problema, nos damos cuenta que por cada recursión, el algoritmo dividirá el arreglo en dos subarreglos de tamaño $n/2$ y que además realizará n divisiones, hasta que las últimas recursiones tengan únicamente un elemento. De lo cuál podemos modelar su ecuación en recurrencia de la siguiente forma:

$$T(n) = 2T(n/2) + n$$

La cual, al igual que el ejercicio anterior podemos resolver por medio del teorema maestro. Tenemos que $a = 2$, $b = 2$ y $c = 1$, dado que se cumple que $a = b^c$, entonces diremos que

$$T(n) = \Theta(n^c \log n)$$

$$\therefore T(n) = \Theta(n \log n)$$