Instituto Politécnico Nacional
Escuela Superior de Cómputo

Análisis de algoritmos recursivos
Ejercicios 06

Resendiz Chavez Luis Fernando
Análisis de Algoritmos
Profesor Edgardo Adrian Franco Martinez
16/12/2020

Ejercicio 01

```
1  int coef(int n,int k)
2  {
3      if (k==0||k==n)
4          return 1;
5      else if (k>0&&k<n)
6          return coef(n-1,k-1)+coef(n-1,k);
7  }
```

Conteo de instrucciones

- Línea 3: 3 comparaciones
- Línea 5: 3 comparaciones
- Línea 6: 4 aritméticas y 2* T(n-1)

Encontrando la función complejidad temporal

$T(n) = 2T(n-1) + 10$

$T(n) - 2T(n-1) = 10,\ con\ b = 1,\ d = 0,\ k = 1$

Sustituyendo en la fórmula y encontrando las raíces:

$(x-2)(x-1)^1 = 0,\ r1 = 2,\ r2 = 1$

Reescribiendo la función a partir de las raíces obtenidas:

$T(n) = c1\cdot 2^n + c2\cdot 1^n\ \varepsilon\ O(2^n)\ con\ c1 \neq 0$

Ejercicio 02

```
1  Busqueda(A[],i,val)
2  {
3      if(i<0)
4          return -1
5      if(A[i]==val)
6          return i
7      return Busqueda(A[],i-1,val)
8  }
```

Conteo de instrucciones

- Línea 3: 1 comparación
- Línea 5: 1 comparación
- Línea 7: 1 aritmética + T(n-1)

Encontrando la función complejidad temporal

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

Sustituyendo en la fórmula y encontrando las raíces:

$T(n) - T(n-1) = 3,\ con\ k = 1,\ b = 1,\ d = 0$

$(x-1)(x-1) = 0,\ r_1 = 1,\ r_2 = 1$

Reescribiendo la función a partir de las raíces obtenidas:

$T(n) = c_1 n^0 \cdot 1^n + c_2 \cdot n^1 \cdot 1^n$

$T(n) = c_1 + c_2 \cdot n\ \varepsilon\ O(n)\ con\ c_2 \neq 0$

Ejercicio 03

```
1   Palindromo(cadena)
2   {
3       if(longitud(cadena)==1)
4           return TRUE
5       if(primer_caracter(cadena)!=ultimo_caracter(cadena))
6           return FALSE
7
8       cadena=remover_primer_ultimo_caracter(cadena)
9       Palindromo(cadena)
10  }
```

Conteo de instrucciones

- Línea 3: 1 comparación + 1(función longitud)
- Línea 5: 1 comparación + 1(función primer_character) + 1(función ultimo_character)
- Línea 8: 1 asignacion + 1(función remover_characteres)
- Línea 9: T(n-2)

Encontrando la función complejidad temporal

$T(n) = T(n-2) + 7$

Sustituyendo en la fórmula y encontrando las raíces:

$T(n) - T(n-2) = 7$

$T(n) + 0 \cdot T(n-1) - T(n-2) = 7,\ con\ k = 2,\ b = 1,\ d = 0$

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

$(x-1)(x+1)(x-1) = 0,\ r_1 = 1,\ r_2 = -1,\ r_3 = 1$

Reescribiendo la función a partir de las raíces obtenidas:

$T(n) = c_1 \cdot n^0 \cdot 1^n + c_2 \cdot -1^n + c_3 \cdot n^2 \cdot 1^n$

$T(n) = c_1 + c_2 + c_3 \cdot n\ \ \varepsilon\ O(n),\ con\ c_3 \neq 0$

Ejercicio 04

```
1   SubAlgoritmo Volados(n,cadena)
2       Si n!=0
3           Volados(n-1,concatenar(cadena,'S'))
4           Volados(n-1,concatenar(cadena,'A'))
5       SiNo
6           Mostrar cadena
7       FinSi
8   FinSubAlgoritmo
```

Conteo de instrucciones

- Línea 2: 1 comparación
- Línea 3: T(n-1) + 1 aritmética
- Línea 4: T(n-1) + 1 aritmética
- Línea 6: 1(función mostrar_cadena)

Encontrando la función complejidad temporal

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

Sustituyendo en la fórmula y encontrando las raíces:

$T(n) - 2T(n-1) = 4,\ con\ k = 1,\ b = 1,\ d = 0$

$(x-2)(x-4) = 0,\ con\ r_1 = 2,\ r_2 = 4$

Reescribiendo la función a partir de las raíces obtenidas:

$T(n) = c_1 \cdot 2^n + c_2 \cdot 4^n\ \ \varepsilon\ O(4^n),\ con\ c_2 \neq 0$

Ejercicio 05

```
1 DecABin(n)
2 {
3     if(n>1)
4     DecABin(n/2)
5     Mostrar(n%2)
6 }
```

Conteo de instrucciones

- Línea 3: 1 comparación
- Línea 4: 1 aritmética
- Línea 5: 1 aritmética + 1(función mostrar)

Casos de prueba

```
T(0) = 2
T(1) = 2
T(2) = 3 + T(1) = 3 + 2 = 5
T(3) = 3 + T(2) = 3 + 5 = 8
T(4) = 3 + T(3) = 3 + 8 = 11
T(5) = 3 + T(4) = 3 + 11 = 14
```

Encontrando la función complejidad temporal

Basándonos en el peor de los casos, sin importar el valor de la variable ‘a’ y ‘n’ siendo el valor de ‘b’:

$T(n) = 3n - 1 \; ε \; O(n)$

Ejercicio 06

```
1   int Producto( int a, int b)
2   {
3
4       if (b==0)
5           return 0;
6       else
7           return a + Producto(a,b-1);
8   }
```

Conteo de instrucciones

- Línea 4: 1 comparación
- Línea 7: 2 aritméticas y T(b-1)

Casos de prueba

```
T(0) = 2
T(1) = 3 + T(0) = 3 + 0 = 3
T(2) = 3 + T(1) = 3 + 3 = 6
T(3) = 3 + T(2) = 3 + 6 = 9
```

Encontrando la función complejidad

Basándonos en el peor de los casos

$T(n) = 3n \; \varepsilon \; O(n)$