Programación II . Algoritmos y EDD. 2024

- Agenda 15-4

- Complejidad Temporal en fragmentos de código

- Estructuras Dinámicas
  - Listas enlazadas

- TDA PILA
  - Implementación con Estructuras Dinámicas

Cómo medimos la eficiencia de los algoritmos

- Estudiamos la eficiencia asintótica de los algoritmos

  Es decir, cómo se incrementa la cantidad de trabajo realizada por un algoritmo a medida que se incrementa el tamaño de la entrada con valores suficientemente grandes.

- Usaremos la notación asintótica O (notación Big-Oh), ya que provee un límite máximo que no será superado por la función de complejidad.

Consideraciones

El tiempo de ejecución de sentencias simples (Operaciones simples, independientes del tamaño de la entrada)

- **Operaciones aritméticas**

  a=a*2 ; a % 2 ; resto = num1 % num2; …..

- **Operaciones lógicas**

  edad >= 18 && tieneRegistro

- **Operaciones de comparación**

  num1 <= num2 ; num1 == num2

Todas se ejecutan en un tiempo constante O(1)

Consideraciones

El tiempo de ejecución de sentencias simples (Operaciones simples, independientes del tamaño de la entrada)

- **Operaciones de acceso a variables, miembros de estructuras o a elementos de un arreglo**

```
int edad = 30;
   public void mostrarEdad() {
      // Acceso a la variable de instancia desde un método
      System.out.println("La edad es: " + edad)
…………………………………………………………………………..
public class Persona {
   String nombre;
   int edad;

   // Constructor
   public Persona(String nombre, int edad) {
      this.nombre = nombre;
      this.edad = edad;
   }
```

Todas se ejecutan en un tiempo constante O(1)

Consideraciones

El tiempo de ejecución de sentencias simples (Operaciones simples, independientes del tamaño de la entrada)

- **Asignaciones simples**

```
// Asignación a variables simples
int edad = 25;
double altura = 1.75;
String nombre = "María";

// Asignación a un arreglo de enteros
int[] numeros = {10, 20, 30, 40, 50};
```

Todas se ejecutan en un tiempo constante O(1)

Consideraciones

El tiempo de ejecución de sentencias simples (Operaciones simples, independientes del tamaño de la entrada)

- **Entrada y salida de datos simples**

```
// Solicitar al usuario que ingrese su nombre
    System.out.print("Ingrese su nombre: ");
    String nombre = scanner.nextLine(); // Leer una línea de texto

// Mostrar información utilizando System.out.println
    System.out.println("Nombre: " + nombre);
    System.out.println("Edad: " + edad);
    System.out.println("Altura: " + altura);
```

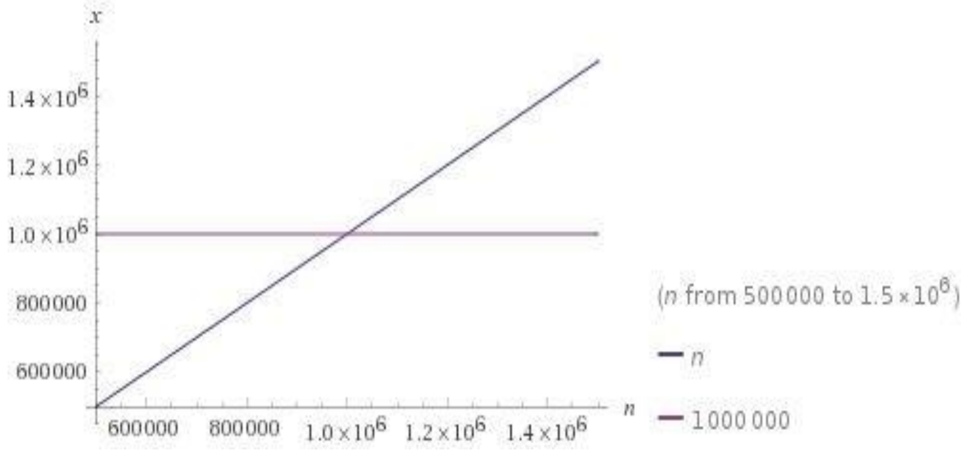
Todas se ejecutan en un tiempo constante O(1)

Ejemplos

Dado un fragmento de código:

```
for (int i = 0; i < (pow(10, 6)); i++) {
      int a;
      a = 10 * 10;
}
```

$$\sum_{i=0}^{10^6-1} c_0 = 10^6 c_0 \quad \rightarrow \in O(1)$$




```
for (int j = 0; j < n; j++) {
      int a;
      a = 10 * 10;
}
```

$$\sum_{j=0}^{n-1} c_0 = n c_0 \quad \rightarrow \in O(n)$$

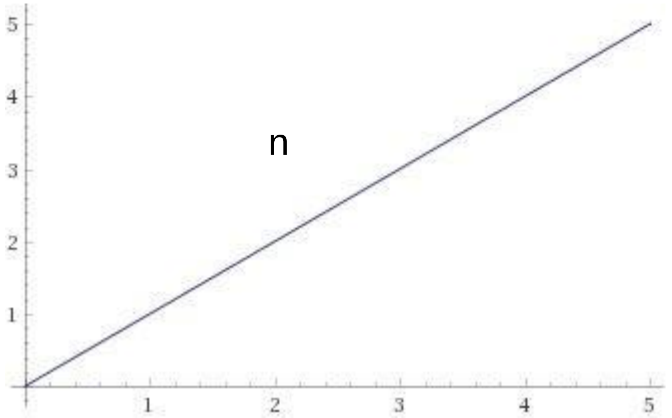
Ejemplos

Otro fragmento de código

```
for (int j = n; j < 0; j--) { int
      a;
      a = 10 * 10;
}
```

Lo planteamos de la misma manera que en el caso anterior y llegamos al mismo resultado:

$$\sum_{j=0}^{n-1} c_0 = nc_0 \quad \rightarrow \quad \in O(n)$$



Tiempos de ejecución más comunes (Big-Oh)

Ordenes de funciones:

$$\left.\begin{array}{l} O(1) \rightarrow \text{Constante} \\ O(\log n) \rightarrow \text{Logarítmico} \\ O(n) \rightarrow \text{Lineal} \\ O(n \log n) \rightarrow n \log n \\ O(n^2) \rightarrow \text{Cuadrático} \\ O(n^3) \rightarrow \text{Cúbico} \end{array}\right\} \text{Polinomial}$$

$$\left.\begin{array}{l} O(2^n) \\ O(n!) \\ O(n^n) \end{array}\right\} \text{Exponencial}$$

Ejemplo:

Analizamos cada sentencia

```
void Calculo (double a, double b, double c)
     { double resultado;     → O(1)
     resultado = a + b + b*c + (a+b-c)/(a+b) + 4.0;      → O(1)
     cout << resultado << endl;       → O(1)
}

→ Operaciones simples, independientes del tamaño de entrada → O(1).
→ Resolvemos: T(n) = max(O(1), O(1), O(1)) → O(1)
```

Ejemplo:

- Ciclos

```
float Suma (float arreglo[ ], int cantidad)
     { float suma= 0; (sería c0)
     for (int i = 0; i < cantidad; i++)
          suma += arreglo [i]; (el c1)
     return suma;  (seria c2)
}
```

$\rightarrow$ **T(n) $\in$ O(n)**

$$T(n) \leq c_0 + \sum_{i=0}^{n-1}(c_1) + c_2$$
$$\leq c_0 + c_1(n) + c_2$$

$$T(n) \in O(n)$$

$\rightarrow$ con n = cantidad

Ejemplo:

- Ciclos

$$
\begin{aligned}
T(n) &\le c_0 + \textstyle\sum_{i=2}^{n} (c_1 + c_2 + c_3) + c_4 \\
&\le c_0 + c\,(n-2+1) + c_4 \qquad\qquad c = c_1 + c_2 + c_3 \\
&\le c_0 + c\,(n-1) + c_4
\end{aligned}
$$

$$T(n) \in O(n)$$

```
unsigned int Fibonacci (unsigned int i) {
      int Fi_1 = 1, Fi_2 = 1, Fi= 1;
      for (int j = 2; j <= i; j++) {
            Fi= Fi_1 + Fi_2;
            Fi_2 = Fi_1;
            Fi_1 = Fi;
      }
      return Fi;
}
```

→ T(n) ∈ O(n)   donde n=i

Ejemplo:

- Ciclos anidados

```
void Transpuesta (int Matriz [][SIZE]){
      for (int i = 0; i < SIZE; i++)
            for (int j = i+1; j < SIZE; j++) {
                  int aux= Matriz [i][j];
                  Matriz [i][j]= Matriz [j][i];
                  Matriz [j][i]= aux;
            }
}
```

→ Considerando SIZE variable y no constante

→ **$T(n) \in O(n^2)$ donde n=SIZE**

$$T(n) \le c_0 + \sum_{i=0}^{n-1}\left(c_1 + \sum_{j=i+1}^{n-1}(c_2)\right) + c_3$$

$$\le c_0 + \sum_{i=0}^{n-1}\left(c_1 + ((n-1)-(i+1)+1)(c_2)\right) + c_3$$

$$\le c_0 + \sum_{i=0}^{n-1}\left(c_1 + (n-1-i)(c_2)\right) + c_3$$

$$\le c_0 + \sum_{i=0}^{n-1}\left(c_1 + nc_2 - c_2 - ic_2\right) + c_3$$

$$\le c_0 + \sum_{i=0}^{n-1}(c_1) + \sum_{i=0}^{n-1}(nc_2) - \sum_{i=0}^{n-1}(c_2) - \sum_{i=0}^{n-1}(ic_2) + c_3$$

$$\le c_0 + c_1 n + \sum_{i=0}^{n-1}(nc_2) - c_2 n - \sum_{i=0}^{n-1}(ic_2) + c_3$$

$$\le c_0 + c_1 n + n^2 c_2 - c_2 n - c_2 \sum_{i=0}^{n-1}(i) + c_3$$

$$\le c_0 + c_1 n + n^2 c_2 - c_2 n - c_2((n-1)(((n-1)+1)/2)) + c_3$$

$$T(n) \in O(n^2)$$

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
     int k = N;
     while (k > 0) {
          k/=2;
     }
}
```

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
    int k = N;
    while (k > 0) {
        k/=2;
    }
}
```

N	k	1ª it	2ª it	3ª it	4ª it	5ª it	#it
4	4	2	1	0	-	-	3

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
    int k = N;
    while (k > 0) {
        k/=2;
    }
}
```

N	k	1ª it	2ª it	3ª it	4ª it	5ª it	#it
4	4	2	1	0			3
5	5	2	1	0			3
6	6	3	1	0			3
7	7	3	1	0			3
8	8	4	2	1	0		4
9	9	4	2	1	0		4
10	10	5	2	1	0		4
11	11	5	2	1	0		4
12	12	6	3	1	0		4
13	13	6	3	1	0		4
14	14	7	3	1	0		4
15	15	7	3	1	0		4
16	16	8	4	2	1	0	5

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
    int k = N;
    while (k > 0) {
        k/=2;
    }
}
```

N	k	1ª it	2ª it	3ª it	4ª it	5ª it	#it
4	4	2	1	0			3
5	5	2	1	0			3
6	6	3	1	0			3
7	7	3	1	0			3
8	8	4	2	1	0		4
9	9	4	2	1	0		4
10	10	5	2	1	0		4
11	11	5	2	1	0		4
12	12	6	3	1	0		4
13	13	6	3	1	0		4
14	14	7	3	1	0		4
15	15	7	3	1	0		4
16	16	8	4	2	1	0	5

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
     int k = N;
     while (k > 0) {
          k/=2;
     }
}
```

N	k	1ª it	2ª it	3ª it	4ª it	5ª it	#it
4	4	2	1	0			3
5	5	2	1	0			3
6	6	3	1	0			3
7	7	3	1	0			3
8	8	4	2	1	0		4
9	9	4	2	1	0		4
10	10	5	2	1	0		4
11	11	5	2	1	0		4
12	12	6	3	1	0		4
13	13	6	3	1	0		4
14	14	7	3	1	0		4
15	15	7	3	1	0		4
16	16	8	4	2	1	0	5

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
     int k = N;
     while (k > 0) {
          k/=2;
     }
}
```

N	k	1ª it	2ª it	3ª it	4ª it	5ª it	#it
4	4	2	1	0			3
5	5	2	1	0			3
6	6	3	1	0			3
7	7	3	1	0			3
8	8	4	2	1	0		4
9	9	4	2	1	0		4
10	10	5	2	1	0		4
11	11	5	2	1	0		4
12	12	6	3	1	0		4
13	13	6	3	1	0		4
14	14	7	3	1	0		4
15	15	7	3	1	0		4
16	16	8	4	2	1	0	5

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
    int k = N;
    while (k > 0) {
        k/=2;
    }
}
```

$$\boxed{\lfloor \log_2 N \rfloor + 1}$$

N	k	1ª it	2ª it	3ª it	4ª it	5ª it	#it
4	4	2	1	0			3
5	5	2	1	0			3
6	6	3	1	0			3
7	7	3	1	0			3
8	8	4	2	1	0		4
9	9	4	2	1	0		4
10	10	5	2	1	0		4
11	11	5	2	1	0		4
12	12	6	3	1	0		4
13	13	6	3	1	0		4
14	14	7	3	1	0		4
15	15	7	3	1	0		4
16	16	8	4	2	1	0	5

Ejemplos

Código:

```
for (int i = 1; i <= M; i++) {
    int k = N;
    while (k > 0) {
        k/=2;
    }
}
```

$$T(N,M) \approx \sum_{i=1}^{M}\left(c_1 + c_2\left(\boxed{\lfloor \log_2 N \rfloor + 1}\right)\right)$$

$$T(N,M) \approx M\left(c_1 + c_2\lfloor \log_2 N \rfloor + c_2\right)$$

$$T(N,M) \in O(M \log N)$$