Semana 6 Control de Lectura

Tema 4.2 – Análisis de las Estructuras de Control

Este apartado se centra en cómo analizar el **costo computacional** de estructuras de control fundamentales: **secuencias, condicionales y bucles**. El objetivo es determinar el número de operaciones que se ejecutan como función del tamaño de la entrada.

1. Secuencias

Cuando se tienen varias instrucciones ejecutadas una tras otra (sin condiciones ni repeticiones), el **costo total** es simplemente la **suma del costo de cada instrucción**.

Ejemplo:

```
makefile
CopiarEditar
x := a + b       # 1 operación
y := x * c       # 1 operación
```

Total: 2 operaciones

2. Condicionales (if / then / else)

El costo de una estructura condicional depende de **qué rama se ejecuta**:

```
markdown
CopiarEditar
si condición entonces
    bloque1
sino
    bloque2
```

Casos de análisis:

- **Peor caso:** se considera la rama más costosa.
- **Mejor caso:** se considera la menos costosa.
- **Promedio:** se puede usar una ponderación si se conocen las probabilidades de cada rama.

Ejemplo:

```
nginx
CopiarEditar
si x > 0 entonces
    hacer A   # Costo: 3
sino
    hacer B   # Costo: 10
```

- Peor caso: 10

- Mejor caso: 3
- Promedio (si A y B se ejecutan 50% cada uno):
  (3 + 10) / 2 = 6.5

3. Bucles (for / while)

El análisis de un bucle requiere contar cuántas veces se ejecuta el cuerpo del bucle, y multiplicar eso por el costo de cada iteración.

a) Bucles for

```
css
CopiarEditar
para i de 1 hasta n hacer
    instrucción de costo c
```

Se ejecuta **n veces**.
Costo total: c · n

Ejemplo:

```
go
CopiarEditar
para i := 1 hasta n hacer
    x := x + 1   # 1 operación
```

Costo: n operaciones

b) Bucles anidados

Si un bucle está dentro de otro, el costo se multiplica:

```
go
CopiarEditar
para i := 1 hasta n hacer
    para j := 1 hasta n hacer
        instrucción de costo c
```

Costo total: c · n · n = O(n²)

c) Bucles while

El número de iteraciones no está explícito, por lo que se requiere **razonar sobre el comportamiento del ciclo**.

Ejemplo:

```
go
CopiarEditar
x := n
mientras x > 1 hacer
```

```
    x := x / 2
```

El número de iteraciones es $\log_2(n)$
Costo total: O(log n)

4. Consideraciones Importantes

- El análisis debe incluir el costo de **comparaciones, asignaciones** y otras operaciones.
- A veces, el número de iteraciones depende del **valor de la entrada**, no sólo de su tamaño.
- En ciclos con condiciones complejas o entradas dinámicas, puede ser difícil obtener una fórmula exacta; en esos casos se usan **cotas asintóticas**.

Ejemplo Integral

Supón el siguiente algoritmo:

```
go
CopiarEditar
para i := 1 hasta n hacer
    si A[i] > 0 entonces
        x := x + A[i]
```

- El bucle se ejecuta **n veces**.
- Cada iteración incluye:
    - 1 comparación
    - En el peor caso, 1 suma y 1 asignación

Total en el peor caso:

- Comparaciones: n
- Operaciones adicionales: hasta n
- **Costo total: O(n)**