Tecnológico de Monterrey

Instituto Tecnológico de Estudios Superiores Monterrey

CAMPUS QUERÉTARO

Análisis y diseño de algoritmos avanzados

Ramona Fuéntes Valdéz

Grupo 602

Actividad 1.2_Análisis de complejidad

PRESENTAN

Rodrigo Terán Hernández A01704108

Fecha:
17/08/2023

Cálculo de la complejidad

1)

```
for (int i=0; i<10; i++) {
  instruccion;
}
```

Complejidad: $O(1)$

El bucle siempre va a iterar un número constante de veces. Es decir, no cambia… Por eso es constante. La complejidad es $O(10)$ pero se simplifican las constantes, dando $O(1)$.

2)

```
for (int i=0; i<n; i++) {
  instruccion;
}
```

Complejidad: $O(n)$

La relación que tiene la variable n con el bucle es lineal ya que a medida que n crezca/disminuya la cantidad de iteraciones va a crecer/disminuir de la misma manera.

3)

```
for (int i=0; i<n/2; i++) {
  instruccion;
}
```

Complejidad: $O(n)$

Aquí el crecimiento de iteraciones a medida que n crezca es de $O\left(\frac{n}{2}\right)$ pero como en la notación “Big O” las constantes no son relevantes entonces solo escribimos la n.

4)

```
for (int i=1; i<=n; i+=4) {
  instruccion;
}
```

Complejidad: $O(n)$

De la misma manera, la variable i va a ir iterando cada 4 unidades, entonces la complejidad es de $O\left(\frac{n}{4}\right)$ pero al omitir la constante nos queda la n.

5)

```
for (int i=1; i<=n/2; i+=5) {
  instruccion;
}
```

Complejidad: $O(n)$

La variable i va a ir iterando cada 5 unidades y la n, que es el límite, crece partida por 2. Entonces la complejidad es de $O\left(\frac{n}{2} * \frac{1}{5}\right)$ que simplifica a $O\left(\frac{n}{10}\right)$ pero al omitir la constante nos queda la n.

6)

```
for (int i=1; i<=n; i*=2) {
  instruccion;
}
```

Complejidad: $O(log\ n)$

Aquí nuestra variable i va a ir creciendo de manera geométrica… Entonces en este caso el número de iteraciones no decrece de manera lineal, sino logarítmica. Por lo tanto, la complejidad es de $O\left(log_2 n\right)$ que simplifica a $O(log\ n)$.

7)

```
for (int i=1234; i>0; i--) {
  instruccion;
}
```

Complejidad: $O(1)$

Aquí el número de iteraciones siempre va a ser el mismo, entonces es constante. La complejidad es $O(1234)$ pero se simplifican las constantes, dando $O(1)$.

8)

```
for (int i=n; i>0; i--) {
  instruccion;
}
```

Complejidad: $O(n)$

El número de iteraciones va a partir de n hasta un número constante… Entonces la complejidad es simplemente lineal, $O(n)$.

9)

```
for (int i=n/5; i>0; i--) {
  instruccion;
}
```

Complejidad: $O(n)$

El número de iteraciones va a partir de n partido por 5, entonces la complejidad es simplemente lineal, $O\left(\frac{n}{5}\right)$, y esto se simplifica a $O(n)$.

10)

```
for (int i=n; i>0; i-=5) {
  instruccion;
}
```

Complejidad: $O(n)$

La variable i va a ir disminuyendo de 5 en 5, entonces la complejidad es simplemente lineal, $O\left(\frac{n}{5}\right)$, y esto se simplifica a $O(n)$.

11)

```
for (int i=n/7; i>0; i-=3) {
  instruccion;
}
```

Complejidad: $O(n)$

Aquí partimos n en 7 y la variable i va a ir disminuyendo de 3 en 3. Por lo tanto la cantidad de iteraciones se va a reducir por esas constantes, $O\left(\frac{n}{7} * \frac{1}{3}\right)$ que da $O\left(\frac{n}{21}\right)$, y esto se simplifica a $O(n)$.

12)

```
for (int i=n; i>0; i/=3) {
  instruccion;
}
```

Complejidad: $O(log\ n)$

Aquí i va a ir disminuyendo de manera geométrica… Por lo tanto la cantidad de iteraciones se va a reducir de manera logarítmica, esto da $O\left(log_3 n\right)$ que simplifica a $O(log\ n)$.

13)

```
if (n%=2){
  for (int i=1; i<=n; i*=2){
          instruccion;
  }
}
else{
  for (int i=4; i<100; i++){
     instruccion;
  }
}
```

Complejidad: $O(log\ n)$

Si n es impar entonces se itera desde 1 a n pero iterando en progresión geométrica… Esto nos da $O\left(log_2 n\right)$ que simplifica a $O(log\ n)$.

Ahora, si n es par entonces se itera siempre de manera constante, que da $O(96)$ iteraciones que simplifica a $O(1)$ por las constantes.

Como es una o la otra se puede escribir como una suma: $O(log\ n) + O(1)$. Al tener esto tomamos la más alta, la cual es $O(log\ n)$.

14)

```
if (n%=2){
  for (int i=1; i<=n; i*=2){
          instruccion;
  }
}
else{
  for (int i=4; i<n; i++){
     instruccion;
  }
}
```

Complejidad: $O(n)$

Si n es impar entonces se itera desde 1 a n pero iterando en progresión geométrica… Esto nos da $O\left(log_2 n\right)$ que simplifica a $O(log\ n)$.

Ahora, si n es par entonces se itera de manera lineal, la cual es $O(n - 4)$ que simplifica a $O(n)$ por las constantes.

Como es una o la otra se puede escribir como una suma: $O(log\ n) + O(n)$. Al tener esto tomamos la más alta, la cual es $O(n)$.

15)

```
for (int i=1; i<=n; i*=2) {
  for (int j=4; j<n; j++){
    instruccion;
  }
}
```

Complejidad: $O(n\ log\ n)$

Aquí en cada iteración del bucle interior se aumenta de manera constante lo cual da la complejidad de $O(n - 4)$ en donde se simplifica a $O(n)$.

Pero cuando termine el bucle se va a volver a iterar en cada iteración del bucle exterior. La cantidad de iteraciones del bucle exterior va a ir aumentando de manera logarítmica, lo cual da $O(log_2 n)$ simplifica a $O(log\ n)$.

Como en cada iteración sucede un conjunto de otras iteraciones esto resulta en una multiplicación. Lo cual resulta en una complejidad de $O(n\ log\ n)$.

16)

```
for (int i=1; i<=n; i+=2) {
  for (int j=1; j<n; j++){
    instruccion;
  }
}
```

Complejidad: $O(n^2)$

Al igual que en el problema anterior, por cada iteración del bucle exterior se va a volver a ejecutar las instrucciones del bucle inferior. Ambos bucles resultan en una complejidad lineal ya que el aumento de la variable i es aritmética.

Cuando se multiplican $O(n)\ *\ O(n)$ nos da $O(n^2)$.