Complejidad de método burbuja y burbuja optimizado

Roberto Benitez Zamora

September 2023

La notación "big O" se denota como $O(f(n))$, donde "n" representa el tamaño de la entrada y "$f(n)$" describe la relación entre el tamaño de la entrada y la cantidad de operaciones que realiza el algoritmo. A menudo, $f(n)$ es una función matemática que representa el peor caso de tiempo o recursos que el algoritmo puede consumir en función del tamaño de la entrada.

1 Burbuja

Para el siguiente código del método de burbuja:

```
def burbuja(lista):
    n = len(lista)
    for i in range(n):
        for j in range(0, n-i-1):
            if lista[j] > lista[j+1]:
                lista[j], lista[j+1] = lista[j+1], lista[j]
```

El análisis de la notación Big O es el siguiente:

1. **Peor Caso:** En el peor caso, el bucle externo se ejecutará n veces, donde n es la longitud de la lista. El bucle interno se ejecutará aproximadamente

   $$\frac{n \cdot (n-1)}{2}$$

   veces en la primera pasada, luego aproximadamente

   $$\frac{n \cdot (n-2)}{2}$$

   veces en la segunda pasada, y así sucesivamente. En general, el bucle interno ejecutará un total de

   $$\sum_{i=1}^{n-1} \frac{n \cdot (n-i)}{2}$$

   veces en el peor caso. Esto simplifica a una complejidad cuadrática, es decir, $O(n^2)$ en notación Big O.

2. **Mejor Caso:** El mejor caso ocurre cuando la lista ya está ordenada y el bucle interno nunca necesita realizar intercambios. Sin embargo, incluso en el mejor caso, el algoritmo realiza las

mismas comparaciones y ejecuta el bucle interno $n(n-1)/2$ veces. Por lo tanto, el mejor caso también es cuadrático: $O(n^2)$.

3. **Caso Promedio:** En el caso promedio, el algoritmo también tiende a ser cuadrático debido a la naturaleza de los intercambios que realiza el bucle interno.

Por lo tanto, en resumen:

- Peor Caso: $O(n^2)$
- Mejor Caso: $O(n^2)$
- Caso Promedio: $O(n^2)$

Este algoritmo de ordenamiento burbuja tiene una complejidad cuadrática y no es eficiente para listas grandes. Se recomienda considerar algoritmos de ordenamiento más eficientes, como QuickSort, MergeSort u otros algoritmos con mejor rendimiento en el caso promedio.

2 Burbuja Optimizado

Para el siguiente código del método de burbuja optimizado:

```
def burbuja_op(lista):
    n = len(lista)
    for i in range(n):
        intercambiado = False
        for j in range(0, n-i-1):
            if lista[j] > lista[j+1]:
                lista[j], lista[j+1] = lista[j+1], lista[j]
                intercambiado = True
        if not intercambiado:
            break
```

El análisis de la notación Big O es el siguiente:

1. **Peor Caso:** En el peor caso, el bucle interno se ejecutará aproximadamente
$$\frac{n \cdot (n-1)}{2}$$
veces, donde n es la longitud de la lista. El bucle externo se ejecuta n veces. Por lo tanto, el peor caso es cuadrático en el número de elementos en la lista, es decir, $O(n^2)$.
2. **Mejor Caso:** El mejor caso ocurre cuando la lista ya está ordenada y la bandera 'intercambiado' nunca se establece en 'True', lo que hace que el bucle externo se ejecute solo una vez. El bucle interno se ejecutará igual que en el peor caso, pero como el bucle externo se ejecuta solo una vez, el mejor caso también es cuadrático, es decir, $O(n^2)$.
3. **Caso Promedio:** El caso promedio depende de la distribución de datos de entrada, pero en general tiende a ser cuadrático, similar al peor caso.

Por lo tanto, en resumen:

- Peor Caso: $O(n^2)$
- Mejor Caso: $O(n^2)$
- Caso Promedio: $O(n^2)$

3 Comparar Tiempos

Para el método de burbuja, en el caso de tener un arreglo con 1000 valores generados con números aleatorios, el tiempo requerido para ejecutarse fue de 0.03499 Segundos (Este siendo el peor o caso promedio).
Para el mismo caso pero ahora con el método de burbuja optimizado, el tiempo de ejecución fue de 0.03409 Segundos.

Ahora en el caso de tener todos los valores del arreglo ordenado con el método de burbuja, el tiempo de ejecución fue de 0.02599 Segundos (El mejor caso).
Para el método de burbuja optimizado, el tiempo de ejecución fue de 0.0 Segundos.

Cabe aclarar que estos valores fueron dados con el programa ejecutándose en mi computador de escritorio, variando los valores dependiendo de la potencia de cada maquina.

4 Conclusión

Ambos códigos son esencia lo mismo, pero para el código/método de burbuja optimizado, al revisar primero si es que el arreglo deseado ya se encuentra ordenado, se evita que se siga ejecutando al ya no haber intercambio en un recorrido completo del arreglo, así reduciendo el tiempo de ejecución