Análisis de Algoritmos de Ordenamiento

Unidad 1
Análisis de Complejidad Computacional

Práctica 01: Análisis de Casos

Docente:
M. en C. Erika Sánchez-Femat

Alumno:
Escobar Hernández Alejandro Daniel

Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas
Instituto Politécnico Nacional

Septiembre 2023

Análisis de los Algoritmos de Ordenamiento Burbuja y Burbuja Optimizada

Burbuja

- **Mejor Caso:** El mejor caso ocurre cuando la lista ya está ordenada. En este caso, el algoritmo realizará una pasada completa sin realizar ningún intercambio. Complejidad: $O(n)$.

  Ejemplo: Lista ya ordenada: $[1, 2, 3, 4, 5]$

- **Peor Caso:** El peor caso ocurre cuando la lista está ordenada en orden inverso, y el algoritmo debe realizar intercambios en cada comparación. Complejidad: $O(n^2)$.

  Ejemplo: Lista ordenada en orden inverso: $[5, 4, 3, 2, 1]$

- **Caso Promedio:** El caso promedio para el algoritmo de burbuja también es $O(n^2)$, ya que se basa en comparaciones y swaps repetitivos.

  Ejemplo: Lista desordenada aleatoriamente: $[3, 1, 5, 2, 4]$

Burbuja Optimizada

- **Mejor Caso:** El mejor caso es similar al de burbuja. Ocurre cuando la lista ya está ordenada, pero la burbuja optimizada detecta que no hay intercambios en la primera pasada. Complejidad: $O(n)$.

  Ejemplo: Lista ya ordenada: $[1, 2, 3, 4, 5]$

- **Peor Caso:** El peor caso es el mismo que el de burbuja, cuando la lista está ordenada en orden inverso. Complejidad: $O(n^2)$.

  Ejemplo: Lista ordenada en orden inverso: $[5, 4, 3, 2, 1]$

- **Caso Promedio:** Al igual que el caso de burbuja, el caso promedio para la burbuja optimizada es $O(n^2)$.

  Ejemplo: Lista desordenada aleatoriamente: $[3, 1, 5, 2, 4]$

En cuanto a la clasificación de la complejidad de cada caso:

- $O(1)$: Ninguno de los casos de estos algoritmos pertenece a esta categoría, ya que todos tienen un tiempo de ejecución dependiente del tamaño de la entrada.
- $O(\log n)$: Ninguno de los casos de estos algoritmos tiene una complejidad logarítmica.
- $O(n)$: El mejor caso de ambos algoritmos es $O(n)$, ya que solo requieren una pasada para verificar que la lista ya está ordenada.
- $O(n \log n)$: Ninguno de los casos de estos algoritmos tiene una complejidad logarítmica.
- $O(n^2)$: El peor caso de ambos algoritmos es $O(n^2)$, ya que en el peor de los casos deben realizar comparaciones y swaps cuadráticos.
- $O(2^n)$: Ninguno de los casos de estos algoritmos tiene una complejidad exponencial.

Análisis de los Algoritmos de Ordenamiento Burbuja y Burbuja Optimizada

Mejor Caso

Para ambos algoritmos (burbuja y burbuja optimizada), el mejor caso se produce cuando la lista ya está ordenada. En este escenario, ambos algoritmos tienen un rendimiento similar y ejecutan en tiempo lineal ($O(n)$).

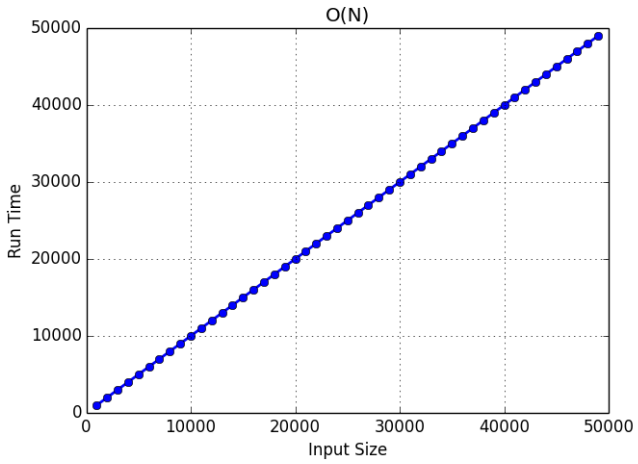

Figure 1: Rendimiento en el Mejor Caso

La principal conclusión es que, en el mejor caso, la burbuja optimizada no ofrece una ventaja significativa sobre el algoritmo de burbuja estándar, ya que ambos tienen un rendimiento similar.

Peor Caso

Tanto el algoritmo de burbuja como el de burbuja optimizada tienen un peor rendimiento en el caso en que la lista está ordenada en orden inverso (peor caso). Ambos algoritmos tienen una complejidad cuadrática ($O(n^2)$).

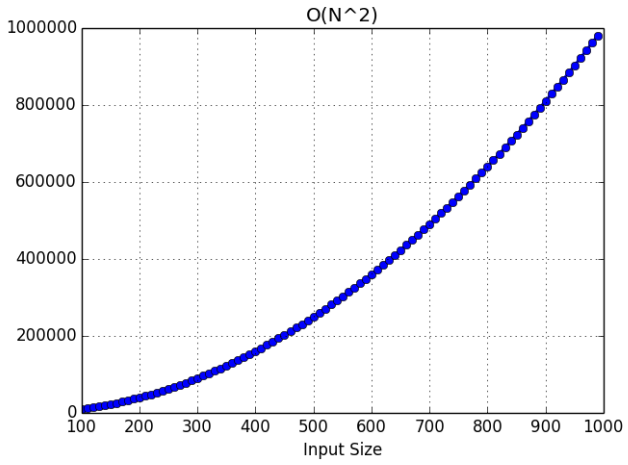

Figure 2: Rendimiento en el Peor Caso

La conclusión es que, en el peor caso, la burbuja optimizada tampoco mejora significativamente el rendimiento en comparación con el algoritmo de burbuja estándar. Ambos son ineficientes en situaciones en las que se necesita ordenar una lista en orden inverso.

Caso Promedio

En el caso promedio, que se produce cuando la lista está desordenada aleatoriamente, nuevamente vemos que tanto la burbuja como la burbuja optimizada tienen un rendimiento similar, con una complejidad cuadrática ($O(n^2)$).

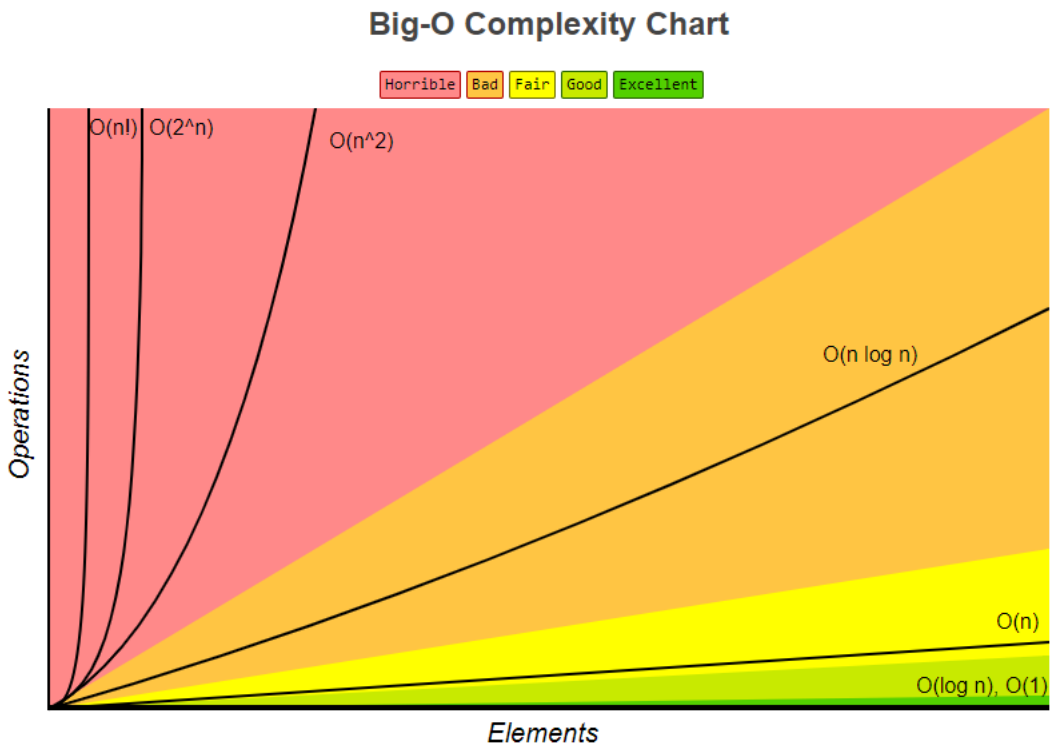

Figure 3: Rendimiento en el Caso Promedio

La conclusión es que, en general, la burbuja optimizada no ofrece una mejora significativa en el rendimiento en comparación con la burbuja estándar en casos promedio.

En resumen, tanto la burbuja como la burbuja optimizada son algoritmos de ordenamiento simples pero ineficientes en términos de tiempo de ejecución en el peor y caso promedio, ya que tienen una complejidad cuadrática. Su rendimiento es más aceptable en el mejor caso, cuando la lista ya está ordenada. Sin embargo, en la práctica, existen algoritmos de ordenamiento más eficientes, como QuickSort, MergeSort o HeapSort, que son preferibles para listas grandes o cuando se requiere un rendimiento más rápido. Estos resultados subrayan la importancia de elegir el algoritmo de ordenamiento adecuado según las características de los datos de entrada y los requisitos de rendimiento.

Programa Python: Ordenamiento de Burbuja

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

def ordenamiento_burbuja_optimizado(arr):
    n = len(arr)
    for i in range(n):
        intercambios = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                intercambios = True
        if not intercambios:
            break

def main():
    print("Seleccione el m todo de ordenamiento:")
    print("1. Burbuja")
    print("2. Burbuja Optimizada")

    opcion = int(input("Ingrese su elecci n (1/2): "))

    if opcion not in [1, 2]:
        print("Opci n no v lida. Por favor, seleccione 1 o 2.")
        return

    n = int(input("Ingrese el tama o de la lista: "))
    arr = []
    for i in range(n):
        elemento = int(input(f"Ingrese el elemento {i+1}: "))
        arr.append(elemento)

    if opcion == 1:
        ordenamiento_burbuja(arr)
        print("Lista ordenada usando Burbuja:", arr)
    elif opcion == 2:
        ordenamiento_burbuja_optimizado(arr)
        print("Lista ordenada usando Burbuja Optimizada:", arr)

if __name__ == "__main__":
    main()
```

Ejemplos de Ejecución

Ejemplo de Ejecución 1 (Mejor Caso):
Seleccione el método de ordenamiento:
1. Burbuja
2. Burbuja Optimizada

Ingrese su elección (1/2): 1

Ingrese el tamaño de la lista: 10

Ingrese el elemento 1: 1
Ingrese el elemento 2: 2
Ingrese el elemento 3: 3
Ingrese el elemento 4: 4
Ingrese el elemento 5: 5
Ingrese el elemento 6: 6
Ingrese el elemento 7: 7
Ingrese el elemento 8: 8
Ingrese el elemento 9: 9
Ingrese el elemento 10: 10
Lista ordenada usando Burbuja: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Ejemplo de Ejecución 2 (Peor Caso):
Seleccione el método de ordenamiento:
1. Burbuja
2. Burbuja Optimizada

Ingrese su elección (1/2): 1

Ingrese el tamaño de la lista: 10

Ingrese el elemento 1: 10
Ingrese el elemento 2: 9
Ingrese el elemento 3: 8
Ingrese el elemento 4: 7
Ingrese el elemento 5: 6
Ingrese el elemento 6: 5
Ingrese el elemento 7: 4
Ingrese el elemento 8: 3
Ingrese el elemento 9: 2
Ingrese el elemento 10: 1
Lista ordenada usando Burbuja: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Ejemplo de Ejecución 3 (Caso Promedio):
Seleccione el método de ordenamiento:
1. Burbuja
2. Burbuja Optimizada

Ingrese su elección (1/2): 2

Ingrese el tamaño de la lista: 10

Ingrese el elemento 1: 5
Ingrese el elemento 2: 2
Ingrese el elemento 3: 9
Ingrese el elemento 4: 1
Ingrese el elemento 5: 7
Ingrese el elemento 6: 3
Ingrese el elemento 7: 10
Ingrese el elemento 8: 6
Ingrese el elemento 9: 4
Ingrese el elemento 10: 8
Lista ordenada usando Burbuja Optimizada: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Con respecto a las conclusiones basadas en estos ejemplos y el conocimiento teórico:

En el mejor caso, cuando la lista ya está ordenada, tanto el algoritmo de burbuja como el de burbuja optimizada ejecutarán rápidamente en tiempo lineal ($O(n)$). Ambos algoritmos ofrecen un rendimiento similar en este escenario.
En el peor caso, cuando la lista está ordenada en orden inverso, ambos algoritmos son ineficientes y requieren un tiempo cuadrático ($O(n^2)$) para completar la ordenación.
En el caso promedio, que es el escenario más común en la práctica, ambos algoritmos también tienen un rendimiento similar, con una complejidad cuadrática ($O(n^2)$). Aunque la burbuja optimizada puede detectar la finalización temprana en algunos casos, su rendimiento general sigue siendo cuadrático.
En resumen, estos algoritmos de ordenamiento, aunque fáciles de implementar, no son eficientes en términos de tiempo de ejecución para listas grandes. Para conjuntos de datos más grandes, se recomienda utilizar algoritmos de ordenamiento más eficientes como QuickSort, MergeSort o HeapSort, que tienen un mejor rendimiento en el caso promedio y en el peor caso.