INSTITUTO POLITÉCNICO NACIONAL
Unidad Profesional Interdisciplinaria de Ingeniería
Campus Zacatecas.

PRACTICA 1

MATERIA:
Ánalisis y Diseño de Algoritmos

DOCENTE:
Erika Sanchez Femat

GRUPO:
3CM2

ALUMNA:
America Lizbet Flores Alcala

Introduccion

En el siguiente reporte se abordaran dos algoritmos los cuales son: Burbuja y Burbuja optimizado, aquí mismo nos podremos dar cuenta que el algoritmo de burbuja y el algoritmo de burbuja optimizado son variantes del mismo método de ordenamiento básico, pero el algoritmo de burbuja optimizado incluye una optimización que reduce el número de comparaciones y posiblemente el número de intercambios en ciertas situaciones. A través de algunos ejemplos nos quedara un poco mas claro el uso, diferencias y comparaciones de estos dos algoritmos.

Desarrollo

Implementacion de los algoritmos

El algoritmo realiza la ordenación de un arreglo de números utilizando dos algoritmos diferentes, el algoritmo de burbuja y el algoritmo de burbuja optimizada. Primero importe la biblioteca 'time', que se utiliza para medir el tiempo de ejecución de los algoritmos. Después la función 'IngresarDatos()' se encarga de solicitar al usuario el tamaño del arreglo y luego pedir que ingrese los elementos del arreglo uno por uno, en la posición correspondiente. Los elementos se almacenan en una lista 'N', que luego se muestra como "Arreglo Original". La función 'Mostrar(N)' simplemente muestra el contenido del arreglo 'N' en pantalla como "Arreglo Ordenado". La función 'Burbuja(N)' implementa el algoritmo de ordenación burbuja. Este algoritmo funciona comparando pares de elementos adyacentes en el arreglo y realizando intercambios si es necesario hasta que el arreglo esté ordenado. La función 'BurbujaOptimizada(N)' implementa el algoritmo de ordenación burbuja optimizada. Este algoritmo es similar al de burbuja, pero tiene una optimización que puede detener el proceso de ordenación si el arreglo ya está ordenado antes de completar todas las iteraciones utilizando una bandera. Este código inicia un bucle 'while' que muestra un menú de opciones al usuario y permite seleccionar una de las tres opciones: Burbuja, Burbuja Optimizada o Salir. Si el usuario selecciona la opción 1 (Burbuja), el programa mide el tiempo de inicio, llama a 'IngresarDatos()' para obtener el arreglo del usuario, luego llama a 'Burbuja(N)' para ordenarlo y finalmente muestra el tiempo de ejecución. Si el usuario selecciona la opción 2 (Burbuja Optimizada), el programa hace lo mismo que en la opción 1, pero utilizando el algoritmo de burbuja optimizada, el cual utiliza menos pasos. Si el usuario selecciona la opción 3 (Salir), el programa muestra un mensaje de salida y termina. En pocas palabras este código permite al usuario ordenar arreglos de números utilizando los algoritmos de burbuja y burbuja optimizada y ver cuánto tiempo tarda cada algoritmo en ordenar el arreglo.

El algoritmo de ordenamiento burbuja es un algoritmo simple pero ineficiente, va recorriendo repetidamente un arreglo que hay que ordenar, comparando cada par de elementos adyacentes y los intercambia si están en el orden equivocado.

Algunos ejemplos de este caso son:

Supongamos que tenemos el siguiente arreglo [16, 12, 3, 19, 4, 2]

1. Mejor caso
   Ocurre cuando el arreglo ya esta ordenado, pues el algoritmo de burbuja ya no tiene que hacer intercambios y solo pasa por el arreglo para confirmar que están ordenados.
   - Ejemplo: Arreglo original [2, 3, 4, 12, 16, 19]
   - Comparacion: [2, 3, 4, 12, 16, 19] (sin intercambios)
   - Fin del Algoritmo

2. Peor Caso
   Ocurre cuando el arreglo esta ordenado en orden inverso y el algoritmo debe realizar el máximo numero de comparaciones e intercambios en cada paso
   • Ejemplo: Arreglo original [19, 16, 12, 4, 3, 2]
   • Pasos:
   [16, 12, 4, 3, 2, 19] (intercambio 19 y 16)
   [12, 4, 3, 2, 16, 19] (intercambio 16 y 12)
   [4, 3, 2, 12, 16, 19] (intercambio 12 y 4)
   [3, 2, 4, 12, 16, 19] (intercambio 4 y 3)
   [2, 3, 4, 12, 16, 19] (intercambio 3 y 1)
   • Fin del Algoritmo

3. Caso Promedio
   Ocurre cuando implica un numero significativo de comparaciones e intercambios, especialmente en arreglos desordenados aleatoriamente
   • Ejemplo: Arreglo original [12, 2, 19, 4, 3, 16]
   • Pasos: *varían según la implementación*
   [2, 12, 4, 3, 16, 19] (varios intercambios)
   [2, 4, 3, 12, 16, 19] (varios intercambios)
   [2, 3, 4, 12, 16, 19] (varios intercambios)
   • Fin del Algoritmo

Imaginemos que tenemos la siguiente lista de colores: verde, azul, amarillo, naranja y rojo

1. Mejor Caso:
   En este caso el algoritmo realizara una sola pasada por la lista sin necesidad de intercambios ya que se encuentra ordenada alfabéticamente
   • Ejemplo: Arreglo original: [amarillo, azul, naranja, rojo, verde]
   • Comparación: [amarillo, azul, naranja, rojo, verde] (sin intercambios)
   • Fin del Algoritmo

2. Peor caso
   En este caso tendríamos lo colores ordenados alfabéticamente en orden inverso y el algoritmo tendrá que realizar el máximo numero de comparaciones e intercambios en cada paso
   • Ejemplo: Arreglo original [verde, rojo, naranja, azul, amarillo]
   • Pasos:
   [rojo, naranja, azul, amarillo, verde] (intercambio verde y rojo)
   [naranja, azul, amarillo, rojo, verde] (intercambio rojo y naranja)
   [azul, amarillo, naranja, rojo, verde] (intercambio naranja y azul)
   [amarillo, azul, naranja, rojo, verde] (intercambio azul y amarillo)
   • Fin del Algoritmo

3. Caso Promedio
   En este caso el algoritmo realizara un numero significativo de comparaciones e intercambios, especialmente en arreglos de colores desordenados aleatoriamente
   • Ejemplo: Arreglo original [naranja, verde, rojo, azul, amarillo]
   • Pasos:
   [naranja, rojo, verde, azul, amarillo] (varios intercambios)
   [azul, naranja, rojo, verde, amarillo] (varios intercambios)
   [amarillo, azul, naranja, rojo, verde] (varios intercambios)
   • Fin del Algoritmo

Supongamos que tenemos una baraja de cartas ordenada ascendentemente por palo y numero
[A♡, 2♣, 3♠, 4♦, 5♡, 6♣, 7♠, 8♦, 9♡, 10♣]

1. Mejor Caso
   En este caso, el algoritmo de burbuja realizará una sola pasada por la baraja sin necesidad de realizar intercambios, ya que ya está en orden.
   • Ejemplo: Arreglo original [A♡, 2♣, 3♠, 4♦, 5♡, 6♣, 7♠, 8♦, 9♡, 10♣]
   • Comparación: [A♡, 2♣, 3♠, 4♦, 5♡, 6♣, 7♠, 8♦, 9♡, 10♣] (sin intercambios)
   • Fin del Algoritmo

2. Peor Caso
   Supongamos que tienes una baraja de cartas completamente desordenada, de modo que todas las cartas deben intercambiarse en cada paso.
   • Ejemplo: Arreglo original [5♣, A♠, 9♦, 2♡, 7♣, 4♠, 10♦, 3♡, 8♣, 6♠]
   • Pasos:
   [A♠, 5♣, 2♡, 7♣, 4♠, 9♦, 3♡, 8♣, 6♠, 10♦] (varios intercambios)
   [A♠, 2♡, 5♣, 4♠, 7♣, 3♡, 8♣, 6♠, 9♦, 10♦] (varios intercambios)
   [A♠, 2♡, 4♠, 5♣, 3♡, 7♣, 6♠, 8♣, 9♦, 10♦] (varios intercambios)
   [A♠, 2♡, 4♠, 3♡, 5♣, 6♠, 7♣, 8♣, 9♦, 10♦] (varios intercambios)
   • Fin del Algoritmo

3. Caso Promedio
   En este caso el algoritmo de burbuja realizará un número significativo de comparaciones e intercambios, especialmente en una baraja de cartas desordenada de manera aleatoria. • Ejemplo: Arreglo original [8♠, 4♡, 7♣, 2♦, 6♠, 5♣, 10♠, A♦, 3♡, 9♣]
   • Pasos: (el número de pasos puede variar según la implementación)
   [4♡, 7♣, 2♦, 6♠, 5♣, 8♠, A♦, 3♡, 9♣, 10♠] (varios intercambios)
   [4♡, 2♦, 6♠, 5♣, 7♣, A♦, 3♡, 8♠, 9♣, 10♠] (varios intercambios)
   [2♦, 4♡, 5♣, 6♠, 7♣, 3♡, A♦, 8♠, 9♣, 10♠] (varios intercambios)
   [2♦, 4♡, 5♣, 6♠, 3♡, 7♣, 8♠, A♦, 9♣, 10♠] (varios intercambios)
   • Fin del Algoritmo

El algoritmo de burbuja optimizada, también llamado "burbuja mejorada" o "burbuja con bandera", mejora el rendimiento del algoritmo de burbuja básico al introducir una bandera que verifica si se realizaron intercambios en una pasada.
Algunos ejemplos de este caso son:

Supongamos que tenemos el siguiente arreglo

[16, 12, 3, 19, 4, 2]

1. Mejor Caso
   Este caso curre cuando el arreglo ya está completamente ordenado de manera ascendente, igual que en el algoritmo de burbuja básico optimizado. La diferencia es que la bandera evitará que se realicen más pasadas innecesarias.
   • Ejemplo: Arreglo original [2, 3, 4, 12, 16, 19]
   • Pasos:
   Primera pasada $\longrightarrow$ [2, 3, 4, 12, 16, 19] (sin intercambios)
   • Fin del Algoritmo

2. Peor Caso
   Este caso sigue siendo cuando el arreglo está ordenado en orden inverso, y el algoritmo debe realizar el máximo número de comparaciones e intercambios posible en cada paso, similar al algoritmo de burbuja básico optimizado.
   • Ejemplo: Arreglo original [19, 16, 12, 4, 3, 2]
   • Pasos:
   Primera pasada ⟶ [16, 12, 4, 3, 2, 19] (intercambio 19 y 16)
   Segunda pasada ⟶ [12, 4, 3, 2, 16, 19] (intercambio 16 y 12)
   Tercera pasada ⟶ [4, 3, 2, 12, 16, 19] (intercambio 12 y 4)
   Cuarta pasada ⟶ [3, 2, 4, 12, 16, 19] (intercambio 4 y 3)
   Quinta pasada ⟶ [2, 3, 4, 12, 16, 19] (intercambio 3 y 2)
   • Fin del Algoritmo

3. Caso Promedio
   El caso promedio es similar al caso promedio del algoritmo de burbuja básico optimizado. La bandera mejora el rendimiento al detectar cuando el arreglo ya está ordenado, pero aún se realizan comparaciones e intercambios significativos en arreglos desordenados aleatoriamente.
   • Ejemplo: Arreglo original [12, 2, 19, 4, 3, 16]
   • Pasos *(el número de pasos puede variar según la implementación):*
   Primera pasada ⟶ [2, 12, 4, 3, 16, 19] (varios intercambios)
   Segunda pasada ⟶ [2, 4, 3, 12, 16, 19] (varios intercambios)
   Tercera pasada ⟶ [2, 3, 4, 12, 16, 19] (varios intercambios)
   • Fin del Algoritmo

Supongamos que tenemos una lista de nombres en orden alfabético

["Ana", "Carlos", "Elena", "Beto", "David"]

1. Mejor Caso
   Este ocurre cuando la lista ya está ordenada alfabéticamente de manera ascendente. En este caso, el algoritmo de burbuja optimizado realizará una sola pasada por la lista sin necesidad de realizar intercambios.
   • Ejemplo: Arreglo original ["Ana", "Beto", "Carlos", "David", "Elena"]
   • Pasos:
   Primera pasada ⟶ ["Ana", "Beto", "Carlos", "David", "Elena"] (sin intercambios)
   • Fin del Algoritmo

2. Peor Caso
   El peor caso ocurre cuando la lista está ordenada alfabéticamente en orden inverso, y el algoritmo debe realizar el máximo número de comparaciones e intercambios posible en cada paso.
   • Ejemplo: Arreglo original ["Elena", "David", "Carlos", "Beto", "Ana"]
   • Pasos:
   Primera pasada ⟶ ["David", "Carlos", "Beto", "Ana", "Elena"] (intercambio "Elena" y "David")
   Segunda pasada ⟶ ["Carlos", "Beto", "Ana", "David", "Elena"] (intercambio "David" y "Carlos")
   Tercera pasada ⟶ ["Beto", "Ana", "Carlos", "David", "Elena"] (intercambio "Carlos" y "Beto")
   Cuarta pasada ⟶ ["Ana", "Beto", "Carlos", "David", "Elena"] (intercambio "Beto" y "Ana")
   • Fin del algoritmo

3. Caso Promedio
   En este caso el algoritmo de burbuja optimizada realizará un número significativo de comparaciones e intercambios, especialmente en listas desordenadas aleatoriamente. La cantidad exacta de pasos puede variar según la implementación y el orden de los elementos en la lista.
   • Ejemplo: Arreglo original ["Beto", "Elena", "Ana", "David", "Carlos"]
   • Pasos *(el número de pasos puede variar según la implementación):*
   Primera pasada ⟶ ["Beto", "Ana", "David", "Carlos", "Elena"] (varios intercambios)
   Segunda pasada ⟶ ["Ana", "Beto", "Carlos", "David", "Elena"] (varios intercambios)
   Tercera pasada ⟶ ["Ana", "Beto", "Carlos", "David", "Elena"] (sin intercambios)
   • Fin del Algoritmo

Supongamos que tenemos una lista de matrículas de estudiantes en orden ascendente

["2021001", "2021002", "2021003", "2021004", "2021005"]

1. Mejor Caso
   El mejor caso ocurre cuando la lista ya está ordenada alfabéticamente de manera ascendente. En este caso, el algoritmo de burbuja optimizado realizará una sola pasada por la lista sin necesidad de realizar intercambios.
   • Ejemplo: Arreglo original ["2021001", "2021002", "2021003", "2021004", "2021005"]
   • Pasos: Primera pasada ⟶ ["2021001", "2021002", "2021003", "2021004", "2021005"] (sin intercambios)
   • Fin del Algoritmo

2. Peor Caso
   El peor caso ocurre cuando la lista está ordenada alfabéticamente en orden inverso, y el algoritmo debe realizar el máximo número de comparaciones e intercambios posible en cada paso.
   • Ejemplo: Arreglo original ["2021005", "2021004", "2021003", "2021002", "2021001"]
   • Pasos:
   Primera pasada ⟶ ["2021004", "2021003", "2021002", "2021001", "2021005"] (intercambio "2021005" y "2021004")
   Segunda pasada ⟶ ["2021003", "2021002", "2021001", "2021004", "2021005"] (intercambio "2021004" y "2021003")
   Tercera pasada ⟶ ["2021002", "2021001", "2021003", "2021004", "2021005"] (intercambio "2021003" y "2021002")
   Cuarta pasada ⟶ ["2021001", "2021002", "2021003", "2021004", "2021005"] (intercambio "2021002" y "2021001")
   • Fin del Algoritmo

3. Caso Promedio
   En el caso promedio, el algoritmo de burbuja optimizado realizará un número significativo de comparaciones e intercambios, especialmente en listas desordenadas aleatoriamente. La cantidad exacta de pasos puede variar según la implementación y el orden de los elementos en la lista.
   • Ejemplo: Arreglo original ["2021003", "2021002", "2021005", "2021004", "2021001"]
   • Pasos *(el número de pasos puede variar según la implementación):*
   Primera pasada ⟶ ["2021002", "2021003", "2021004", "2021001", "2021005"] (varios intercambios)

Segunda pasada ⟶ [”2021002”, ”2021003”, ”2021001”, ”2021004”, ”2021005”] (varios intercambios)
Tercera pasada ⟶ [”2021002”, ”2021001”, ”2021003”, ”2021004”, ”2021005”] (varios intercambios)
Cuarta pasada ⟶ [”2021001”, ”2021002”, ”2021003”, ”2021004”, ”2021005”] (varios intercambios)
• Fin del algoritmo

Conclusión

En conclusión, nos pudimos dar cuenta que la principal diferencia entre el algoritmo de burbuja y el algoritmo de burbuja optimizado es la inclusión de una bandera para evitar comparaciones innecesarias en el último. Esto mejora la eficiencia en ciertos casos, pero ambos algoritmos siguen siendo relativamente ineficientes en comparación con otros algoritmos de ordenamiento más avanzados, especialmente para grandes conjuntos de datos.

Referencias:

Ordenamientodeburbuja|DelftStack
Algoritmos|Ordenacindeburbujas(lwh-21.github.io)
OrdenamientodeBurbujaenPython-DiegoAmorin