Universidad de La Serena

Diseño y Análisis de Algoritmos
Profesor: Erick Luciano Castillo Bastias
Fecha : Miércoles 29 de mayo
Equipo 2

Integrantes :
Fernando López
Rodrigo Mamani
Bryan Townsend

Índice

1. Introducción

En el siguiente informe se detallara la actividad realizada durante el desarrollo de este. Como sabemos los métodos de ordenamiento cumplen un factor relevante en la computación, ya que el propósito principal por lo general es ordenar una secuencia para luego facilitar las búsquedas de los miembros del conjunto ordenado. Es tanto así que existen diferentes tipos de algoritmos de ordenamiento en los cuales una buena practica sería escoger cuidadosamente que algortimo nos será mas util al momento de tener una secuencia de datos por ejemplo. Un concepto importante a mencionar es la estabilidad del algortimo que escojamos, ya que será fundamental en el proceso de ordenamiento.

Figura 1: Tipos de Algortimos de Ordenamiento(Estables e Inestables)

Estables				
Nombre traducido	Nombre original	Complejidad	Memoria	Método
Ordenamiento de burbuja	Bubblesort	$O(n^2)$	$O(1)$	Intercambio
Ordenamiento de burbuja bidireccional	Cocktail sort	$O(n^2)$	$O(1)$	Intercambio
Ordenamiento por inserción	Insertion sort	$O(n^2)$("(en el peor de los casos)")	$O(1)$	Inserción
Ordenamiento por casilleros	Bucket sort	$O(n)$	$O(n)$	No comparativo
Ordenamiento por cuentas	Counting sort	$O(n+k)$	$O(n+k)$	No comparativo
Ordenamiento por mezcla	Merge sort	$O(n \log n)$	$O(n)$	Mezcla
Ordenamiento con árbol binario	Binary tree sort	$O(n \log n)$	$O(n)$	Inserción
	Pigeonhole sort	$O(n+k)$	$O(k)$	
Ordenamiento Radix	Radix sort	$O(nk)$	$O(n)$	No comparativo
	Distribution sort	$O(n^2)$ versión recursiva	$O(n^2)$	
	Gnome sort	$O(n^2)$	$O(1)$	
Inestables				
Nombre traducido	Nombre original	Complejidad	Memoria	Método
Ordenamiento Shell	Shell sort	$O(n^{1.25})$	$O(1)$	Inserción
	Comb sort	$O(n \log n)$	$O(1)$	Intercambio
Ordenamiento por selección	Selection sort	$O(n^2)$	$O(1)$	Selección
Ordenamiento por montículos	Heapsort	$O(n \log n)$	$O(1)$	Selección
	Smoothsort	$O(n \log n)$	$O(1)$	Selección
Ordenamiento rápido	Quicksort	Promedio: $O(n \log n)$, peor caso: $O(n^2)$	$O(\log n)$	Partición
	Several Unique Sort	Promedio: $O(n\ u)$, peor caso: $O(n^2)$; u=n; u = número único de registros		

es por eso que a partir de la Figura 1,surgen diferentes preguntas considerando a cada uno de los tipos de ordenamientos,una de ellas es ¿Cuándo conviene usar uno o otro método de ordenamiento?

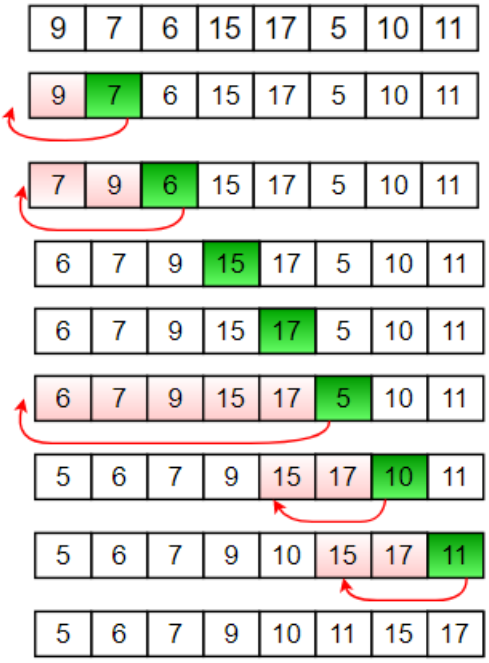
Y la respuesta más rapida es considerar cuando se requiere hacer una cantidad considerable de búsquedas y es importante el factor tiempo.

Una vez contextualizado el tema que estaremos constantemente abordando y aterrrizando con la actividad planteada propongo dar una breve definicón muy por encima de los algoritmos que estaremos estudiando y poniendo a prueba durante el desarrollo de este informe:

1.1. Orden por Inserción (InsertionSort)

Dado un array A de n-elementos, la estrategia consiste en realizar n-recorridos por el array. En el recorrido i-ésimo, el elemento almacenado en A[i-1] es trasladado a su lugar apropiado en A[0..i - 1]. De este modo, después del recorrido i-ésimo, los elementos en A[0...i -1] están en orden, lo que significa que después de n-recorridos, el array A completo estará ordenado.

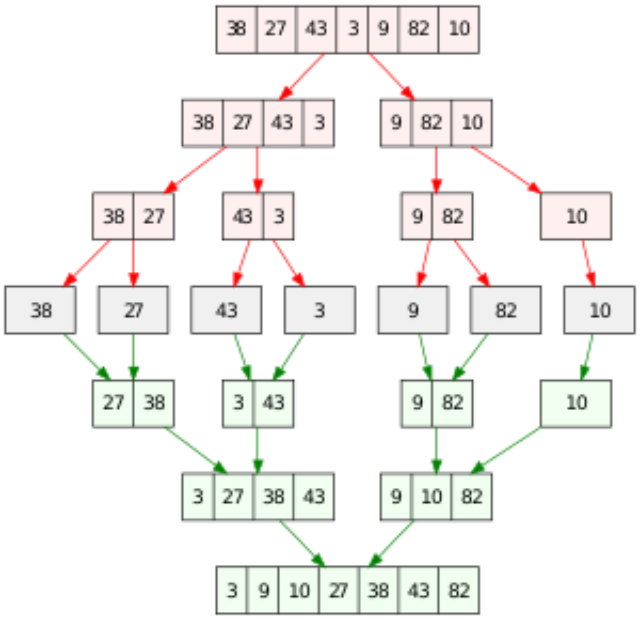
Figura 2: Ejemplo de Ordenamiento por InsertionSort.



1.2. Ordenación por Mezcla (MergeSort)

Dado un array A de n-elementos, la estrategia esta basada en la metodología Dividir para Conquistar, en donde el array es particionado en 2 subarrays, y así sucesivamente hasta Finalmente lograr los subarrays ya ordenados para luego combinarlos para producir el array ordenado final.

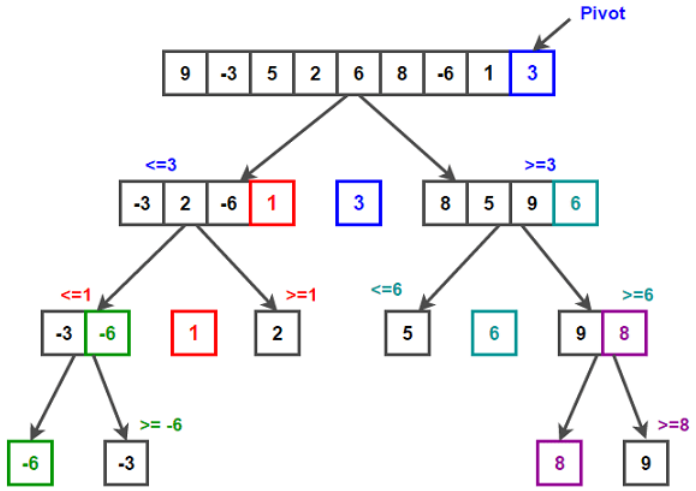
Figura 3: Ejemplo de Ordenamiento por MergeSort.



1.3. Orden Rápido (QuickSort)

Dado un array A de n-elementos, la estrategia también esta basada en la metodología Dividir para Conquistar. Sin embargo, a diferencia del algoritmo de ordenación por mezcla, la parte no recursiva del algoritmo de ordenación rápida conlleva la construcción de subejemplares por medio de una técnica de partición, Se selecciona un elemento especifico del array A[p] llamado pivote, seguidamente se recorre el array haciendo una serie de operaciones considerando en todo momento este elemento que escogeremos, hasta lograr así el ordenamiento.

Figura 4: Ejemplo de Ordenamiento por QuickSort.



2. Desarollo de Software

En la siguiente sección se describirá el progreso que realizamos en conjunto con los desarrolladores durante el progreso de la implementación de los metodos de ordenamiento, para luego poder andalizar y diferenciar cada uno de los algoritmos como así tambien sus métodos utilizados.

2.1. Algoritmos desarrollados y métodos utilizados

RIGO,BRYAN

2.2. Diferencias entre algoritmos y métodos

blablabla

3. Conclusión

Para concluir la actividad realizada Como pudimos observar

4. Referencias Web

Referencias

[1] Diseño y Análisis de Algoritmos. (DAA-2009)–Dr. Eric Jeltsch F. *PDF Diseño y Análisis de Algoritmos*, Segunda Clase, DAA 2019.

[2] E78. INGENIERÍA DEL SOFTWARE 5º CURSO DE INGENIERÍA INFORMÁTICA 2000-2001, *Especificación de Requisitos Software según el estándar de IEEE 830*, Departament de Informática Universitario Jaume I.

[3] Bazaraa, M.S., J.J. Jarvis y H.D. Sherali, *Programación lineal y flujo en redes*, segunda edición, Limusa, México, DF, 2004.

[4] Dantzig, G.B. y P. Wolfe, «Decomposition principle for linear programs», *Operations Research*, **8**, págs. 101–111, 1960.

Appendices

Anexo I: Diagramas UML (casos de uso y de clases)

Anexo II: Manual de usuarios