Tecnológico de Monterrey
TC1031.11: Programación de estructuras de datos y algoritmos fundamentales (GPO 11)
Edgar Cruz Vázquez (A01730577)

Act. 1.3 – Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

Realiza una investigación y reflexión en forma individual de la importancia y eficiencia del uso de los diferentes algoritmos de ordenamiento y búsqueda en una situación problema de esta naturaleza, generando un documento llamado “ReflexAct1.3.pdf”.

El uso de ambos algoritmos (ordenamiento y búsqueda) es importante y esencial, el uso adecuado y eficiente son útiles para poner datos en forma canónica y para generar resultados legibles por humanos.

Los procesos de búsqueda involucran recorrer un arreglo completo con el fin de encontrar algo. Lo más común es buscar el menor o mayor elemento (cuando se puede establecer un orden), o buscar el índice de un elemento determinado.

- Búsqueda Secuencial.
  Consiste en ir comparando el elemento que se busca con cada elemento del arreglo hasta cuándo se encuentra.

  - Eficiencia y Complejidad:

    Mejor caso:
    O(1): El elemento buscado está en la primera posición. Es decir, se hace una sola comparación.

    Peor caso:
    O(n): El elemento buscado está en la última posición. Necesitando igual cantidad de comparaciones que de elementos el arreglo.

    En promedio:
    O(n/2): El elemento buscado estará cerca de la mitad. Necesitando en promedio, la mitad de comparaciones que de elementos.

- Búsqueda Binaria.
  Compara si el valor buscado está en la mitad inferior o inferior. En la que esté, subdivido nuevamente, y así sucesivamente hasta encontrar el valor.

  - Eficiencia y Complejidad:

    Mejor caso:

Tecnológico de Monterrey
TC1031.11: Programación de estructuras de datos y algoritmos fundamentales (GPO 11)
Edgar Cruz Vázquez (A01730577)

Act. 1.3 – Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

O(1): El elemento buscado está en el centro. Por lo tanto, se hace una sola comparación.

Peor caso:
O(log(n)): El elemento buscado está en una esquina. Necesitadno log2(n) cantidad de comparaciones.

En promedio:
O(log(n/2)): Serán algo como log2(n/2).

Ordenar un conjunto de elementos de una lista es una tarea que se presenta con frecuencia en la programación. Un programa de computadora debe seguir una secuencia de instrucciones exactas para lograrlo. Esta secuencia de instrucciones se llama algoritmo. Un algoritmo de ordenamiento es un método que puede utilizarse para colocar una lista de elementos de una secuencia ordenada. La secuencia de ordenamiento está determinada por una clave. Existen varios algoritmos de ordenamiento y difieren en cuanto a su eficiencia y rendimiento.

Los algoritmos se clasifican por:

- Lugar donde se realice la ordenación:
    - Algoritmo de ordenamiento interno.
      En la memoria del ordenador.
    - Algoritmos de ordenamiento externo.
      En un lugar externo como un disco duro.
- Por el tiempo que tardan en realizar la ordenación, dadas entradas ya ordenadas o inversamente ordenadas:
    - Algoritmos de ordenación natural.
      Tarda lo mínimo posible cuando la entrada está ordenada.
    - Algoritmos de ordenación no natural.
      Tarda lo mínimo posible cuando la entrada está inversamente ordenada.
- Por estabilidad.

Tecnológico de Monterrey
TC1031.11: Programación de estructuras de datos y algoritmos fundamentales (GPO 11)
Edgar Cruz Vázquez (A01730577)

Act. 1.3 – Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

- Complejidad computacional.

- Uso de memoria y otros recursos computacionales.

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)

REFLEXIÓN:

En esta evidencia pude desarrollar las competencias necesarias para poder realizar el trabajo, durante la elaboración de esta tuve que aplicar habilidades de observación, lógica y experiencia ya que fue un gran reto descifrar la forma en la cual debía ordenar los datos con la utilización de vectores y conceptos como el de tokenizar, para esta evidencia usamos el ordenamiento BubbleSort por la simplicidad en la aplicación en el código, esta tiene una complejidad de tiempo

Tecnológico de Monterrey
TC1031.11: Programación de estructuras de datos y algoritmos fundamentales (GPO 11)
Edgar Cruz Vázquez (A01730577)

Act. 1.3 – Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)

de O(n^2) la cual la hace ineficiente pero con una complejidad espacial de O(1) el mejor de todos, para la búsqueda utilizamos la secuencial en la cual su peor caso es O(n), en el momento de su finalización logre ver la importancia de uso de ambos algoritmos ya que al correrlo el tiempo que toma ordenarlo va desde 3-5 min en la cual es demasiado, seria rápido en caso serían pocos datos pero aquí estamos acomodando aprox 16,000 líneas, si quisiéramos ordenar mas líneas y buscarlas (como en posiciones donde usen bastante información, como Amazon) tendríamos que buscar y aplicar mejores algoritmos para la manipulación de datos.