Resumen de Pruebas realizadas con Algoritmos de Ordenamiento y Búsqueda en lenguaje de programación C++.

Andrés Norberto Cáceres Becerra - 1152074

Octubre 25 2022

Universidad Francisco de Paula Santander

Norte de Santander

Análisis de Algoritmos

En esta tarea decidí utilizar 3 algoritmos de ordenamiento que me llamaron mucho la atención. Estos algoritmos fueron:

* **SelectionSort:**

Este algoritmo es muy interesante cómo funciona ya que en cada iteración lo que hace al arreglo es comparar elemento mínimo con el valor del índice. Hace esto para saber cual es el menor de esa iteración y posteriormente cuando llega al final lo ordena y el mínimo ahora será el siguiente del numero ordenado anteriormente. Se repite este proceso hasta que ordena el arreglo dado. También cabe resaltar que este algoritmo tiene un tiempo algorítmico de .

* **BubbleSort:**

Este algoritmo es muy bueno al momento de ordenar arreglos y es muy peculiar como lo hace ya que recorre todo el arreglo en cada iteración comparando 2 elementos y ordenándolos de menor a mayor. Hace esto en cada iteración hasta que lo recorre todo y posteriormente el arreglo queda ordenado. También cabe resaltar que este algoritmo tiene un tiempo algorítmico de .

* **InsertionSort:**

La funcionalidad de este algoritmo es muy peculiar ya que en cada iteración saca el elemento menor del arreglo y posteriormente lo ordena teniendo en cuenta los elementos en el arreglo. Hace esto en cada iteración hasta que termina de recorrer todo el arreglo y ordenarlo. También cabe resaltar que este algoritmo tiene un tiempo algorítmico de .

También en esa tarea se utilizaron dos algoritmos de búsqueda, en mi caso los que elegí fueron:

* **LinearSearch:**

Este algoritmo de búsqueda recorre todo el arreglo comenzando desde la posición 0 hasta la posición n-1 comparando en cada iteración el elemento que está en el índice del arreglo con el que se esta buscando. Si lo encuentra termina la búsqueda y devuelve la posición donde se encuentra el elemento encontrado. Por otro lado, si el elemento no está en el arreglo este hace una búsqueda en todo el arreglo y devuelve un mensaje donde dice que no se encuentra. También cabe resaltar que este algoritmo tiene un tiempo algorítmico de .

* **BinarySearch:**

Este algoritmo de búsqueda es muy interesante como funciona ya que toma el arreglo y se posiciona en la mitad de este y pregunta si el numero que se está buscando es menor a el elemento en el arreglo en la posición del índice y también pregunta si es igual. Si es igual retorna la posición donde lo encontró de lo contrario si es menor toma la parte izquierda de la mitad del arreglo o si es mayor hace lo mismo, pero tomando la parte derecha de la mitad del arreglo. Hace esto hasta recorrer todo el arreglo dividiendo cada vez el pedazo que va tomando. También cabe resaltar que este algoritmo tiene un tiempo algorítmico de .

**Conclusión**

Las pruebas con los algoritmos de ordenamiento las pude realizar con arreglos de tamaño 10^2 y 10^3 ya que cuando intenté hacerla con 10^6 mi IDE no respondía y se quedaba procesando. En el caso de 10^9 no lo pude hacer porque se me quedaba pegado el IDE y no respondía. Pero en los casos que si pude probar (10^2 y 10^3) pude sacar los tiempos de diferentes casos y pudiendo ver cierta diferencia entre ellos. Por ejemplo, pude notar que es más rápido cuando el arreglo inicial está totalmente desordenado a comparación cuando esta parcialmente ordenado. También pude notar que no hay mucha diferencia en ejecución de los tres algoritmos que probé. También realicé las pruebas a los algoritmos de búsqueda y en estos pude darme cuenta de que es más rápido en ciertos casos. Por ejemplo, en BinarySearch es más rápido si el elemento a buscar se encuentra en toda la mitad del arreglo. Y en la búsqueda lineal me fijé que puede tomar mucho más tiempo si el elemento no se encuentra o se encuentra al final del arreglo. También me gusto mucho utilizar el lenguaje de programación C++.

Las especificaciones de computador portátil donde realicé las pruebas son:

Modelo del sistema Nitro AN515-52

Procesador Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, 2304 Mhz, 4 procesadores principales, 8 procesadores lógicos

Memoria física instalada (RAM) 8,00 GB

# Lista de referencias

Programiz. (s.f.). Selection Sort Algorithm. <https://www.programiz.com/dsa/selection-sort>

Programiz. (s.f.). Bubble Sort. <https://www.programiz.com/dsa/bubble-sort>

Programiz. (s.f.). Insertion Sort Algorithm. <https://www.programiz.com/dsa/insertion-sort>

geeksforgeeks. (s.f.). Linear Search Algorithm. <https://www.geeksforgeeks.org/linear-search/>

geeksforgeeks. (s.f.). Binary Search. <https://www.geeksforgeeks.org/binary-search/>