Estructuras de datos y algoritmos avanzados

Laboratorio 1

Rodrigo Alexander Richards Valenzuela

Matrícula: 2016404813

El problema dado para este laboratorio es implementar algoritmos de búsqueda para encontrar la posición en la que se encuentra un elemento pedido, dentro de un arreglo de tamaño n. Para cada algoritmo se buscó el último elemento del arreglo para forzar el peor caso.

Complejidades de algoritmos

1. Búsqueda lineal/secuencial

   Complejidad: $O(p)$

   Este algoritmo se basa en encontrar el elemento pedido, iterando desde la primera posición del elemento, hasta encontrar el elemento pedido, de esta forma, en el peor caso, se iterará desde el primer elemento, hasta el último elemento, vale decir se realizarán n iteraciones, de esta forma la complejidad dada por este algoritmo estará en función de la posición del elemento buscado, es decir $O(p)$.

2. Búsqueda binaria

   Complejidad: $O(log(n))$

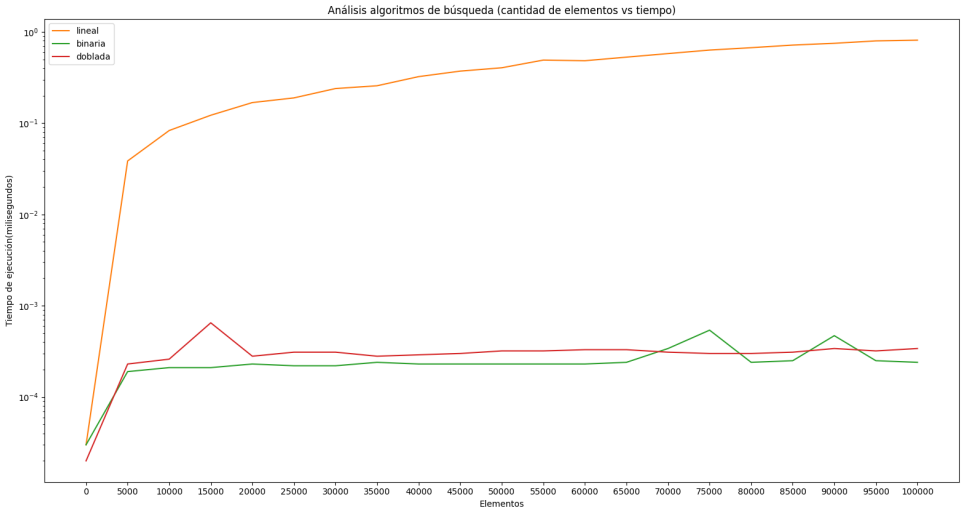
   Este algoritmo divide el arreglo en la mitad, vale decir se toma como pivote la posición $\frac{n}{2}$ y se verifica si el elemento está en esa posición, si no se encuentra, se debe verificar si el elemento de esa mitad es mayor o menor al elemento buscado, si es mayor se toma como pivote inicial la posición que se tomó como pivote anteriormente y se debe dividir nuevamente entre el nuevo pivote inicial y el pivote final (para un primer caso será n), en caso contrario se toma como pivote final la posición pivote dada anteriormente y se procede a realizar la división entre el pivote inicial (en un primer caso será 0) y el final, de esta forma se irán descartando los elementos que vayan siendo mayor o menor al elemento buscado, dividiendo cada subproblema de cada iteración a la mitad, de esta manera se obtiene una complejidad de $O(log(n))$

3. Búsqueda doblada

   Complejidad: $O(log(p))$

   Este algoritmo es un tipo de combinación de los algoritmos anteriores, dado que realiza una búsqueda secuencial, pero en cada búsqueda dobla la posición a buscar, una vez que se encuentra una posición donde el elemento en esa posición sea **mayor** al elemento buscado, se deja de doblar la posición y se procede a utilizar búsqueda binaria para encontrar el elemento. Al ir doblando la posición se obtiene una secuencia exponencial para encontrar el elemento, de esta forma se obtiene complejidad $O(log(p))$ y cuando se realiza la búsqueda binaria a través de ese límite encontrado, se tiene que la complejidad es de $O(log(n))$ donde n es el tamaño del intervalo de búsqueda, en este caso el intervalo está dado por el límite encontrado el cuál está en función de p, de esta forma la complejidad para realizar la búsqueda binaria será de $O(log(p))$ y por lo tanto la complejidad del algoritmo completo será $O(log(p))$

Gráfico comparativo



Del gráfico observado, vemos que la búsqueda lineal es la que demora más tiempo, ya que se buscará sobre todos los elementos del arreglo y la posición final será el resultado final ya que se forzó el peor caso para comparar estos algoritmos, también vemos que la búsqueda binaria posee un tiempo de ejecución menor que la búsqueda doblada, esto es debido a que la búsqueda doblada resulta ser mejor solo en los casos cuando el elemento buscado se encuentra cerca del principio del arreglo o bien cuando se encuentra el elemento en la primera fase del algoritmo.