Carlo Ángel Luján García

A01639847

Programación de estructuras de datos y algoritmos fundamentales (Gpo 13)

Profesor Luis Ricardo Peña Llamas

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

12/09/2021

Tecnológico de Monterrey

Realizar 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"

Algoritmos de ordenamiento

METODOS ITERATIVOS

Estos métodos son simples de entender y de programar ya que son iterativos, simples ciclos y sentencias que hacen que el vector pueda ser ordenado. Dentro de los Algoritmos iterativos encontramos:

– Burbuja:

El método de la burbuja es uno de los más simples, es tan fácil como comparar todos los elementos de una lista contra todos, si se cumple que uno es mayor o menor a otro, entonces los intercambia de posición.

– Inserción:

El bucle principal de la ordenación por inserción va examinando sucesivamente todos los elementos de la matriz desde el segundo hasta el n-ésimo, e inserta cada uno en el lugar adecuado entre sus predecesores dentro de la matriz.

– Selección

La ordenación por selección funciona seleccionando el menor elemento de la matriz y llevándolo al principio; a continuación, selecciona el siguiente menor y lo pone en la segunda posición de la matriz y así sucesivamente.

METODOS RECURSIVOS

Estos métodos son aún más complejos, requieren de mayor atención y conocimiento para ser entendidos. Son rápidos y efectivos, utilizan generalmente la técnica Divide y vencerás, que consiste en dividir un problema grande en varios pequeños para que sea más fácil resolverlos. Mediante llamadas recursivas a sí mismos, es posible que el tiempo de ejecución y de ordenación sea más optimo. Dentro de los algoritmos recursivos encontramos:

– Ordenamiento por Mezclas (merge)

Este algoritmo consiste básicamente en dividir en partes iguales la lista de números y luego mezclarlos comparándolos, dejándolos ordenados. Si se piensa en este algoritmo recursivamente, podemos imaginar que dividirá la lista hasta tener un elemento en cada lista, luego lo compara con el que está a su lado y según corresponda, lo sitúa donde corresponde.

– Ordenamiento Rápido (quick)

Lo que hace este algoritmo es dividir recursivamente el vector en partes iguales, indicando un elemento de inicio, fin y un pivote (o comodín) que nos permite segmentar nuestra lista. Una vez dividida, lo que hace, es dejar todos los mayores que el pivote a su derecha y todos los menores a su izq. Al finalizar el algoritmo, nuestros elementos están ordenados.

Algoritmos de búsqueda

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

ALGORITMOS DE BÚSQUEDA:

– Búsqueda Secuencial:

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

– Búsqueda Binaria

La Búsqueda Binaria, compara si el valor buscado está en la mitad superior o inferior. En la que esté, subdivido nuevamente, y así sucesivamente hasta encontrar el valor. Cabe destacar que en este metodo de búsqueda la lista debe estar previamente ordenada

Reflexión:

La ordenación y las búsquedas son aplicaciones fundamentales en la computación, durante todo momento mientras estamos programando y sea necesario desplegar datos o recibir datos de alguna forma, y se espera que estos datos puedan ser leídos por el usuario, o bien necesiten ser ordenados para hacer búsquedas con mayor facilidad. Supongamos que un conjunto de datos esta desordenado, en caso de que nosotros queramos mostrar al usuario la información, esta persona tardará más tiempo en encontrarlo en comparación si los datos tuvieran algún patrón de ordenamiento útil para su visualización. En cualquiera de los casos si nuestro usuario deseara buscar un dato dentro una lista de datos si bien ordenados o desordenados, una computadora podría encontrarlo mucho más rápido que un humano y su ojo, he aquí donde reside la importancia de los métodos de búsqueda en la computación. Ambas acciones ordenación y búsqueda están directamente relacionadas, por que por un lado una ordenación a fin de cuentas es indispensable para que en la ejecución la computadora pueda ahorrar tiempo, esfuerzo y el programador pueda pensar en algún sistema que de igual forma sea optimizado para un sistema en específico. Mientras que la búsqueda solo tiene un propósito, buscar datos lo más rápido y eficiente para que el sistema despliegue lo encontrado, y en conjunto usando un set de datos ordenados por un patrón, permitirá que la computadora pueda ahorrar tiempo en caso de que se programase el método correspondiente, por lo que a fin de cuentas el programador tiene el poder de hacer un sistema más eficiente o decidir que el usuario batalle para ver los datos.

Una vez explicada esta pequeña reflexión del porque son importantes los métodos de ordenamiento y de búsqueda me gustaría explicar ahora por que es importante que estos métodos sean eficientes en problemas de esta naturaleza, en donde los sets de datos son extremadamente grandes (En nuestra actividad se tiene un número de entradas que es algo elevado), o ya no sean tan sencillos de observar tanto por el usuario ni por la computadora. En empresas con inmensos volúmenes de datos como Facebook, Twitter, Instagram, WhatsApp, Amazon, y en general empresas que almacenan cantidades descomunales de datos, debido a su gran cantidad de usuarios, el hecho de tener métodos de búsqueda y ordenamiento lentos los haría molestos y por ende el usuario podría dejar de usarlos buscando algo mejor y que no le haga perder tiempo. Es aquí donde entra la eficiencia en los algoritmos. Supongamos que un usuario quiere comprar una bicicleta en Amazon.com, bueno… para poder realizar esa acción el usuario de iniciar sesión, esta función que parece despreciable, implica que se haga una búsqueda entre millones y millones de datos de otros usuarios, por lo que el simple hecho de tener que iniciar sesión implica que alguien en el back end tuvo que programar un método que fuera lo suficientemente eficaz para que esta cantidad desmesurable de datos que implicaría millones de operaciones computacionales, se pudiera hacer en solo algunos milisegundos o segundos. Ahora una vez que el usuario entro para comprar su bicicleta iniciando sesión, en algún momento atrás en la parte de la programación se tuvo que hacer una búsqueda ¨Bicicleta¨, pero estos datos no los mostrara de forma random sino especializada usando un método de ordenamiento, que permita que el usuario pueda ver lo que el necesita y le conviene más, es aquí donde entran los algoritmos de ordenamiento específicos, que si recordamos a fin de cuentas todos estas acciones son para que un usuario obtenga su información de la forma más rápida.

Explicando un poco del trasfondo de la complejidad computacional podemos usar la notación Big O, que nos habla de que forma crecerá el tiempo al ejecutarse un algoritmo en la computadora, esto implica operaciones dentro de ella y una mayor cantidad de operaciones implica más tiempo de procesamiento, si bien esto explicado hay distintos métodos de ordenamiento conocidos y de búsqueda. Se puede considerar que un algoritmo es no tan eficaz si este tiene una complejidad n^2 o mayor, debido a que si por ejemplo

en el caso de la actividad realizada en bitácora teníamos una entrada de 16700 datos y queríamos ejecutar un ordenamiento que usara una complejidad n^2 implicaría que la computadora debería realizar 278,890,000 operaciones para solo ordenarlos, que en práctica pudimos notar que

Usando método de ordenamiento por Bubble Sort:

```
Sorting entries...
Entries sorted
Time difference (sec) = 93.7404
Saving sorted entries...
Sorted entries saved.
```

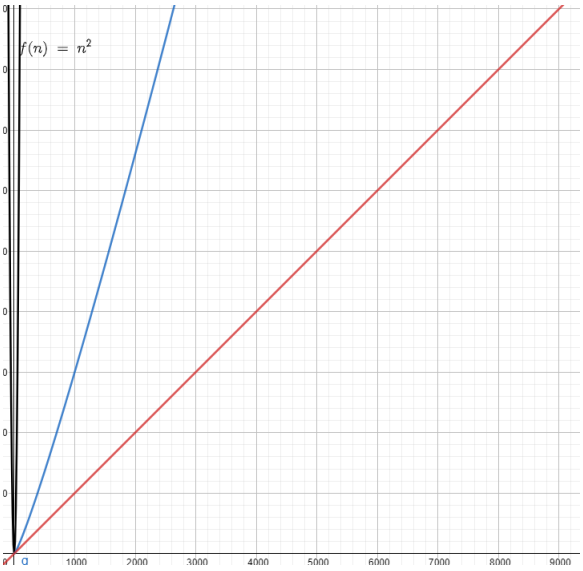
La computadora tardó 1:33 min en solo realizar el ordenamiento, recordemos que el método de ordenamiento por bubble Sort tiene una complejidad computacional de n^2 lo que implica lo anteriormente mencionado, una entrada como la que se tenia de 16700 datos hizo que la computadora tuviera que ejecutar operaciones 278 millones de veces.

Usando método de ordenamiento por Merge Sort:

```
Loading entries...
Entries loaded (16807).

Sorting entries...
Entries sorted
Time difference (sec) = 0.184
Saving all entries...
Sorted entries saved successfully, check your folder
```

Mientras que al usar merge sort el cual tiene una complejidad computacional de n*log(n), que con una entrada de 16700 implicaría que tuvo que realizar 70519.36 operaciones significando que este algoritmo se pudo haber ejecutado x3942 veces para llegar al mismo numero de operaciones por la misma entrada! demorándose .184 segundos, ósea x510 veces más rápido que con bubble sort.



Comparando sus comportamientos visualmente podemos ver como el algoritmo nlogn(azul) se comporta casi igual que al lineal n(rojo), mientras que el n^2(negro) incrementa casi como una línea recta vertical.

Una vez comparando visualmente y con números podemos confirmar que la complejidad computacional de los algoritmos es fundamental al querer ahorrar tiempo.

Referencias:

Universidad técnica Federico Santa María (2010) Algoritmos de Búsqueda y Ordenamiento Programación de Computadores Recuperado de: https://www.inf.utfsm.cl/~noell/IWI-131-p1/Tema8b.pdf