Reflexión Actividad 1.3

Pablo Navarro Zepeda
A01284116

Reflexión Individual Pablo

Durante esta actividad pusimos en practica el verdadero uso y eficiencia de diferentes algoritmos. Hablando específicamente de la situación planteada, podemos decir que el tomar en cuenta si el algoritmo será **recursivo o iterativo fue fundamental** ya que en este caso contábamos con una bitácora de 16,807 casos. Debido a lo aprendido en clase y lo que investigamos , nos dimos cuenta de que un algoritmo de Búsqueda Binaria que es recursivo no seria de gran utilidad, ya que este tipo de algoritmos no son tan buenos con muchos datos. La importancia de la eficiencia se vuelve cada mas importante a medida que el numero de datos crece por eso fue importante decidir usar una búsqueda secuencial. El haber analizado los diferentes algoritmos en la actividad 1.2 nos dio un amplio panorama para decidir que algoritmos de ordenamiento y búsqueda se amoldaban mejor a nuestra situación problema. Según la tercera edición "**Introduction to Algorithms", Capitulo 1 Pagina 5**, los algoritmos de ordenamiento y búsqueda son fundamentales ya que cumplen con un rol inicial o como parte de muchos programas que hacen procedimientos mucho mas complejos. Es por esto la importancia de conocer los diferentes que hay y saber analizarlos y usarlos en casos específicos

Jorge del Barco
A01284234

Reflexión Individual Jorge

Después de haber realizado la actividad y de habernos enfrentado a diferentes retos como lo fue comprender en su totalidad los algoritmos que estamos utilizando, su complejidad, y la manera de implementarlos utilizando los vectores en C++, se llegó a un aprendizaje valioso: elegir el algoritmo correcto es importante y conocer sobre la teoría de ellos también.

Un algoritmo de ordenamiento es un programa que nos permite ordenar información de una manera específica. Para lo que hemos visto en clase, nos permite ordenar valores numéricos de forma ascendente o descendente. El motivo de utilizar estos algoritmos es ordenar los datos con los que se trabaja para habilitar un acceso a la información más rápido y posiblemente hasta seguro. Por ejemplo, si realizamos cambios a un arreglo que contiene información en el lugar equivocado, estaríamos dañando nuestra base de datos. Por lo que tener organizada información es importante sea cual sea el área o trabajo.

Aplicado al desarrollo del proyecto entregado, al inicio, se quiso utilizar el algoritmo Quicksort para ordenar por fecha los más de 16 mil elementos que se tenían que clasificar. Sin embargo, este demostró ser muy lento, pues tiene una complejidad de O(n^2). Después de seguir buscando un algoritmo más eficiente de los que hemos visto en clase nos topamos con merge sort. Este es un algoritmo recursivo que tiene una complejidad de O(n logn). Al implementarlo, la superioridad del algoritmo con este volumen de datos se vio de manera casi inmediata. Con Quicksort nunca alcanzamos a

ver el resultado final sorteado, y con merge sort, bastaron unos segundos para tener el resultado esperadoy así poder proceder a un acceso de la información.
En el caso de los algoritmos de búsqueda, se decidió optar por la búsqueda secuencial para facilitar el desarrollo de una función que se utilizaría más adelante en el proyecto. Es un algoritmo rápido y desde mi punto de vista tiene una complejidad aceptable de O(n). Como se mencionó anteriormente con el merge sort, el ordenamiento permite un acceso más fácil a la información y simplifica la dificultad que existe para manipular los datos.
Fue interesante darse cuenta de que el factor que nos hizo decidir por cual algoritmo de búsqueda usar fue nuestro tiempo. El tiempo de buscar un algoritmo más eficiente hacía que el desarrollo del proyecto se frenara, por lo que decidimos optar por el más adecuado a nuestras necesidades y capacidades en ese momento. Me gusta pensar que parte del trabajo lo hace la computadora una vez tiene ya cargado el programa, pero la investigación, desarrollo e implementación del algoritmo es lo que más consume tiempo y cuesta al proyecto.

Ana Lucía Cárdenas
A01284090

Reflexión Individual Ana
Los algoritmos son herramientas un poco complejas a primera vista, pero podemos ver que con pocas líneas de código se puede lograr demasiado. Logramos ordenar un archivo con más de 16 mil líneas de datos para que al utilizar el método de búsqueda, este fuera preciso y más sencillo de que nos imprimiera la información que nosotros como usuarios pidiéramos. Es impresionante como estos algoritmos se pueden utilizar para muchas cosas diferentes, solo es saber darles un valor para que el sistema pueda reconocerlo y así ordenarlo.

Para este sistema decidimos utilizar el método Merge Sort el cual tiene una complejidad de O(nlogn) y el método de búsqueda secuencial el cual tiene una complejidad de O(n). La complejidad de estos métodos se vio eficaz ya que hicieron su proceso relativamente rápido. Al inicio habíamos intentado utilizar el método de quick sort el cual tiene una complejidad de O(n^2) y vimos que este se tardaba en llevar a cabo el proceso, y como buscamos eficiencia en este tipo de programas, optamos por usar Merge Sort. El saber varios algoritmos de ordenamiento es bueno ya que como la mayoría tienen la misma base, se pueden hacer pruebas con cada uno para ver cual es el más eficaz para cada caso que se pueda presentar, ya sea solamente ordenar números, o letras con un valor dado, o como en este caso, meses y días. También la manera en la que el código está formado nos va a poder ayudar a seguir ordenando el archivo en el caso de que si en algún futuro se agrega más información, este no tendrá un límite predeterminado, simplemente va a leer el archivo y darle un valor a n que corresponda al archivo actualizado, es decir, no es un código de uso único, también se puede implementar en el código que este abra más de un (1) archivo y que uno por uno este lo vaya ordenando, claro, tiene que ser un archivo en el mismo formato que el original ya que este código está hecho específicamente para ese formato.
[7:39 p. m., 10/9/2021] +52 1 81 1910 5040: mi matricula es

Análisis Complejidades

Merge Sort

```
void mergeSort(vector<string> &logger, int low, int high)
{
    int mid = (low + high) / 2;     1

    if (low < high)                 2
    {
        mergeSort(logger, low, mid);  3
        mergeSort(logger, mid + 1, high);  4
        merge(logger, low, mid, high);  5
    }
}
```

C_1	1
C_2	0
C_3	$T(\frac{n}{1})$
C_4	$T(\frac{n}{2})$
C_5	n

$T(n) = C_1 + C_3\,T(\frac{n}{2}) + C_4\,T(\frac{n}{2}) + C_5\,n$

$T(n) = 2T(\frac{n}{2}) + n$

$$T(n) = \begin{cases} 1 & ; n=1 \\ 2T(\frac{n}{2}) + n & ; n>1 \end{cases}$$

$T(n) = 2T(\frac{n}{2}) + n$

$= 2\left[2T\left(\frac{\frac{n}{2}}{2}\right) + \frac{n}{2}\right] + n$

$= 2\left[2T\left(\frac{n}{2^2}\right) + n + n\right] = 2^2 T\left(\frac{n}{2^2}\right) + 2n$

$= 2^2\left[2T\left(\frac{n/2^2}{2}\right) + \frac{n}{2^2}\right] + 2n$

$= 2^3 T\left(\frac{n}{2^3}\right) + n + 2n = 2^3 T\left(\frac{n}{2^3}\right) + 3n$

$= 2^k T\left(\frac{n}{2^k}\right) + kn$

$\frac{n}{2^k} = 1 \rightarrow n = 2^k \rightarrow \log_2 n = k$

$= 2^{\log_2 n} T\left(\frac{n}{\log_2 n}\right) + n \log_2 n$

$= nT\left(\frac{n}{n}\right) + n \log_2 n$

$= nT(1) + n \log_2 n$

$T(n) = n + n \log_2 n$

$T(n) = O(n + n \log_2 n) = O(n \log_2 n)$

$O(n \log n)$

$\log_a b = b$

a

Busqueda Secuencial

```
222 int busquedaSecuencial(vector<string> &logger, string key){
223
224     for (int i = 0; i < logger.size(); i++){ //Recorremos vector C1
225
226         /* Checamos si el valor a buscar tiene solo un dia "Feb 2"y checamos que el valor del indice encontrado
227            tambien sea en formato de un dia "Feb 2" no "Feb 23"
228         */
229         //cout << key << "subtr 1: " << logger[i].substr(0,5) << "subtr2: " << logger[i][5] << "\n";
230         if (key.size() == 5 && key == logger[i].substr(0,5) && logger[i][5] == ' '){ C2
231             return i; C3
232         }
233         else if(key.size() == 6 && key == logger[i].substr(0,6) ){ C4
234             return i; C5
235         }
236     }
237     cout << "Es fecha no esta en el rango de datos" << endl;
238 }
239
```

C_1	$n+1$
C_2	n
C_3	1
C_4	1
C_5	1

$$T(n) = C_1(n+1) + C_2 n + C_3 + C_4 + C_5$$

$$T(n) = C_1 n + C_1 + C_2 n + C_3 + C_4 + C_5$$

$$T(n) = C_1 n + C_2 n + C_1 + C_3 + C_4 + C_5$$

$$T(n) = n\,[C_1 + C_2] + [C_1 + C_3 + C_4 + C_5]$$

$$T(n) = O(n)$$