

## Actividad 1.3

**TC1031.501**

**Programación de Estructuras de  
Datos y Algoritmos Fundamentales**

**Profesor Baldomero Olvera Villanueva**





**Integrantes:**

Matías Kochlowski – a01625364







Esteban

Michelle Andrea Arceo Solano - a01625268

# Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales

bitacora.txt			
			
06-01	0:14:39	895.33.752.33:5974	Failed password for illegal user test
06-01	0:17:32	901.18.919.12:5807	Failed password for illegal user guest
06-01	0:22:14	268.82.665.50:6202	Failed password for root
06-01	0:31:39	745.41.553.21:4925	Failed password for admin
06-01	0:42:50	497.97.988.31:6636	Illegal user
06-01	0:49:25	608.37.179.94:6715	Failed password for admin
06-01	0:55:42	335.95.645.32:6284	Illegal user
06-01	1:18:23	10.43.466.53:6937	Failed password for illegal user test
06-01	1:30:57	930.68.543.89:5825	Failed password for illegal user test
...	...	...	...

En equipos de tres personas, hacer una aplicación que:

-  **1** Abra el archivo de entrada llamado "bitacora.txt", lealo y almacene los datos en un vector.
-  **2** Ordene la información por fecha para la realización de las búsquedas.
-  **3** Solicite al usuario las fechas de inicio y fin de búsqueda de información.
-  **4** Despliegue los registros correspondientes a esas fechas.
-  **5** Almacenar en un archivo el resultado del ordenamiento.
-  **6** Realizar una investigación y reflexión \*nombrada "ReflexAct1.3.pdf") en forma individual sobre:
  - a.la importancia y eficiencia del uso de los diferentes algoritmos de ordenamiento
  - b.la importancia y eficiencia del uso de los diferentes algoritmos de búsqueda

en una situación problema de esta naturaleza.

- 1 Abra el archivo de entrada llamado "bitacora.txt", lealo y almacene los datos en un vector.

bitacora.txt

Jun-01	0:14:39	895.33.752.33:5974	Failed password for illegal user test
Jun-01	0:17:32	901.18.919.12:5807	Failed password for illegal user guest
Jun-01	0:22:14	268.82.665.50:6202	Failed password for root
Jun-01	0:31:39	745.41.553.21:4925	Failed password for admin
Jun-01	0:42:50	497.97.988.31:6636	Illegal user
Jun-01	0:49:25	608.37.179.94:6715	Failed password for admin
Jun-01	0:55:42	335.95.645.32:6284	Illegal user
...	...	...	...

```
$ ~ wc -l bitacora.txt
16806
```

16806 Registros

**N = 16806**

Este dato es útil para el cálculo de la complejidad tiempo de los algoritmos (de ordenamiento y búsqueda).

Nos permitirá tomar una decisión fundamentada sobre cuales usar y porqué.

txt - String

```
Oct-30 21:39:26 679.57.853.40:5668 Failed password for root
Oct-30 21:52:57 164.48.60.79:6597 Failed password for illegal user test
Oct-30 22:06:30 774.16.162.29:4184 Illegal user
```

Vector

```
{
  "10-30T21:39:26,679.57.853.40:5668,Failed password for root ",
  "10-30T21:52:57,164.48.60.79:6597,Failed password for illegal user test ",
  "10-30T22:06:30,774.16.162.29:4184,Illegal user "
}
```

```
std::string mesTextoANro(const std::string& mes)
{
    if (mes == "Jun ") return "06-";
    if (mes == "Jul ") return "07-";
    if (mes == "Aug ") return "08-";
    if (mes == "Sep ") return "09-";
    if (mes == "Oct ") return "10-";
    return nullptr;
}
```



## 2 Ordene la información por fecha para la realización de las búsquedas.

```
// Quicksort recursivo
void quickSort(std::vector<std::string> &str, int inicio, int fin) {
    int i = inicio, j = fin;
    // Variable para asistir en cambio de valores
    std::string buffer;
    // Valor medio en lista
    std::string mid = str[(inicio + fin) / 2];
    while (i <= j) {
        // Contador desde inicio hacia el centro
        while (str[i] < mid)
            i++;
        // Contador desde el final hacia el centro
        while (str[j] > mid)
            j--;
        // Comparador y cambiador de valores
        if (i <= j) {
            buffer = str[i];
            str[i] = str[j];
            str[j] = buffer;
            i++; j--;
        }
    }
    // Recursión - función se llama a si misma
    if (inicio < j)
        quickSort(str, inicio, j);
    if (i < fin)
        quickSort(str, i, fin);
}
```

bitacora\_AZ.csv

```
dateTime,ipAddress,errorMessage
06-01T00:14:39,895.33.752.33:5974,Failed password for illegal user test
06-01T00:17:32,901.18.919.12:5807,Failed password for illegal user guest
06-01T00:22:14,268.82.665.50:6202,Failed password for root
06-01T00:31:39,745.41.553.21:4925,Failed password for admin
06-01T00:42:50,497.97.988.31:6636,Illegal user
06-01T00:49:25,608.37.179.94:6715,Failed password for admin
06-01T00:55:42,335.95.645.32:6284,Illegal user
06-01T01:18:23,10.43.466.53:6937,Failed password for illegal user test
06-01T01:30:57,930.68.543.89:5825,Failed password for illegal user test
06-01T02:06:19,271.58.635.18:5758,Failed password for root
06-01T02:27:35,847.39.216.22:6665,Failed password for illegal user test
06-01T02:32:49,698.84.382.6:4362,Failed password for admin
06-01T02:59:15,828.85.492.15:6281,Illegal user
06-01T03:08:19,719.96.787.8:4897,Illegal user
06-01T03:08:35,112.65.132.41:4869,Failed password for admin
06-01T03:10:03,376.95.120.45:4335,Failed password for root
06-01T03:22:38,609.89.179.46:4444,Failed password for root
06-01T03:44:33,257.63.94.92:4928,Failed password for illegal user test
06-01T04:08:36,783.83.33.92:5184,Failed password for root
06-01T04:10:09,730.30.210.65:5167,Failed password for admin
06-01T04:32:33,698.13.580.48:4394,Failed password for admin
06-01T04:48:20,1.55.405.18:6734,Failed password for illegal user guest
```



- 3 Solicite al usuario las fechas de inicio y fin de búsqueda de información.

```
$ ~ ./bitacora
```

```
Esta base contiene 16,806 registros de intentos fallidos de ingreso al sistema desde  
01-Jun hasta 30-Oct.
```

```
Para realizar una búsqueda, por favor ingrese el rango de fechas deseado en formato  
MM-DD. Por ejemplo, el 10 de Junio se debe escribir como 06-10.
```

```
Ingrese Fecha (06-01). Desde: 06-01
```

```
Ingrese Fecha (10-30). Hasta: 06-01
```

```
Buscando resultados entre 06-01 y 06-01.
```



- 4 Despliegue los registros correspondientes a esas fechas.

```
$ ~ ./bitacora
```

```
Esta base contiene 16,806 registros de intentos fallidos de ingreso al sistema desde 01-Jun hasta 30-Oct.
```

```
Para realizar una búsqueda, por favor ingrese el rango de fechas deseado en formato MM-DD. Por ejemplo, el 10 de Junio se debe escribir como 06-10.
```

```
Ingrese Fecha (06-01). Desde: 06-01
```

```
Ingrese Fecha (10-30). Hasta: 06-01
```

```
Buscando resultados entre 06-01 y 06-01.
```

```
06-01T00:14:39,895.33.752.33:5974,Failed password for illegal user test
06-01T00:17:32,901.18.919.12:5807,Failed password for illegal user guest
06-01T00:22:14,268.82.665.50:6202,Failed password for root
06-01T00:31:39,745.41.553.21:4925,Failed password for admin
06-01T00:42:50,497.97.988.31:6636,Illegal user
06-01T00:49:25,608.37.179.94:6715,Failed password for admin
06-01T00:55:42,335.95.645.32:6284,Illegal user
06-01T01:18:23,10.43.466.53:6937,Failed password for illegal user test
```

```
...
```



## 5 Almacenar en un archivo el resultado del ordenamiento.

output/resultados06-13a06-15.csv

```
dateTime,ipAddress,errorMessage
06-13T00:04:26,9.61.622.72:4496,Failed password for root
06-13T00:15:35,288.24.857.85:5146,Failed password for root
06-13T00:51:27,386.94.526.71:5357,Illegal user
06-13T00:51:43,860.90.201.52:4231,Failed password for illegal user test
06-13T00:55:17,587.85.748.77:5962,Failed password for illegal user test
06-13T01:01:27,109.18.555.18:4026,Failed password for illegal user guest
06-13T01:24:28,533.7.248.53:6075,Failed password for illegal user guest
06-13T01:25:05,390.82.654.56:4135,Illegal user
06-13T01:31:54,612.5.910.47:4500,Failed password for illegal user test
06-13T01:35:49,536.94.836.3:5759,Failed password for root
06-13T01:38:23,537.83.851.40:6946,Failed password for root
06-13T01:48:26,859.45.721.16:6146,Failed password for illegal user test
06-13T01:50:43,713.78.547.18:6154,Failed password for illegal user test
06-13T01:56:34,521.51.536.9:6197,Failed password for illegal user guest
06-13T02:11:25,713.78.236.71:4469,Failed password for admin
06-13T03:01:19,336.54.577.44:5838,Failed password for admin
06-13T03:02:25,139.58.729.28:4663,Failed password for admin
06-13T03:36:36,414.83.781.8:5203,Failed password for illegal user test
06-13T04:11:25,484.62.4.48:6371,Failed password for illegal user guest
06-13T04:17:26,555.41.502.33:5343,Failed password for admin
06-13T04:28:35,117.74.289.35:4548,Failed password for admin
06-13T04:29:58,921.30.671.60:6568,Illegal user
06-13T04:35:39,208.68.922.10:5612,Failed password for illegal user guest
06-13T04:53:28,851.19.530.42:6076,Illegal user
06-13T05:02:29,44.12.270.91:5068,Illegal user
06-13T05:17:03,295.90.568.32:6917,Illegal user
06-13T05:24:24,104.56.331.68:4317,Failed password for illegal user test
...
```

output/bitacora\_AZ.csv

```
dateTime,ipAddress,errorMessage
06-01T00:14:39,895.33.752.33:5974,Failed password for illegal user test
06-01T00:17:32,901.18.919.12:5807,Failed password for illegal user guest
06-01T00:22:14,268.82.665.50:6202,Failed password for root
06-01T00:31:39,745.41.553.21:4925,Failed password for admin
06-01T00:42:50,497.97.988.31:6636,Illegal user
06-01T00:49:25,608.37.179.94:6715,Failed password for admin
06-01T00:55:42,335.95.645.32:6284,Illegal user
06-01T01:18:23,10.43.466.53:6937,Failed password for illegal user test
06-01T01:30:57,930.68.543.89:5825,Failed password for illegal user test
06-01T02:06:19,271.58.635.18:5758,Failed password for root
06-01T02:27:35,847.39.216.22:6665,Failed password for illegal user test
06-01T02:32:49,698.84.382.6:4362,Failed password for admin
06-01T02:59:15,828.85.492.15:6281,Illegal user
06-01T03:08:19,719.96.787.8:4897,Illegal user
06-01T03:08:35,112.65.132.41:4869,Failed password for admin
06-01T03:10:03,376.95.120.45:4335,Failed password for root
06-01T03:22:38,609.89.179.46:4444,Failed password for root
06-01T03:44:33,257.63.94.92:4928,Failed password for illegal user test
06-01T04:08:36,783.83.33.92:5184,Failed password for root
06-01T04:10:09,730.30.210.65:5167,Failed password for admin
06-01T04:32:33,698.13.580.48:4394,Failed password for admin
06-01T04:48:20,1.55.405.18:6734,Failed password for illegal user guest
...
```



- 6
- Considerando una situación problema de esta naturaleza, realizar una investigación y reflexión sobre:
- a. la importancia y eficiencia del uso de los diferentes algoritmos de ordenamiento

Ordenamiento Tipo	Complejidad Temporal			Comentarios/Observaciones
	Best	Average	Worst	
Burbuja (Bubble)	$O(N)$	$O(N^2)$	$O(N^2)$	Best case: ya estaba ordenado y requirió una sola pasada."
Merge	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$	En cuanto espacio auxiliar, esa es peor que ordenamiento burbuja, pero en cuanto eficiencia es mejor.
Quicksort	$O(N \log N)$	$O(N \log N)$	$O(N^2)$	Se elige el último nº (no fijado) como el ancla y se compara con el primero y el penúltimo elemento de la lista.  Worst case: El final de la lista ya esta organizada (los últimos 2 elementos - termina comparando todo con todo)  Se puede correr análisis asincrónico y elegir otro nº como ancla para hacerlo más eficiente.  Espacio auxiliar constante.
Heap	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$	Parte ordenada vs parte no ordenada.

Para esta situación problema en particular, serían aceptables usar cualquiera de estos métodos ya que:

- la lista pesa solo 1.1 mB y el espacio de memoria no es problema
- Cualquier computadora moderna puede ordenar y navegar esta lista con mucha facilidad (son solo +16K de registros con 3 campos cortos e importantes)

Cuando son millones de registros y mucho mas complejos, ya se deben considerar algoritmos con complejidades temporales logarítmicas, después lineales pero no exponenciales. Es importante verse cómo están organizados los datos de input para ver si se aplican cambios de anclas en un mergeSort, cual es la capacidad del hardware, etc.





6 Considerando una situación problema de esta naturaleza, realizar una investigación y reflexión sobre:

b. la importancia y eficiencia del uso de los diferentes algoritmos de búsqueda

Búsqueda Tipo	Big 'O' Notation
Binary (Intervalo)	$O(\log N)$
Linear (Secuencial)	$O(N)$

# Bibliografía

---



<https://www.geeksforgeeks.org/why-quick-sort-preferred-for-arrays-and-merge-sort-for-linked-lists/>

<https://www.geeksforgeeks.org/analysis-of-different-sorting-techniques/>

<https://www.geeksforgeeks.org/iterative-quick-sort/>

<https://www.geeksforgeeks.org/difference-between-vector-and-list/>

<https://www.geeksforgeeks.org/external-sorting/>

<https://codereview.stackexchange.com/questions/166313/sorting-large-1gb-file-with-100-millions-numbers-using-merge-sort>