# **Resultado de imagen para tec de monterrey logo png**

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

# **Actividad 1.3 | *Actividad Integral de Conceptos Básicos y Algoritmos Fundamentales (Evidencia Competencia)***

**Alumno:**

Carlos Antonio Buendia López A01379471

Luis Ángel Terrazas García A01377440

**Profesor(a):**

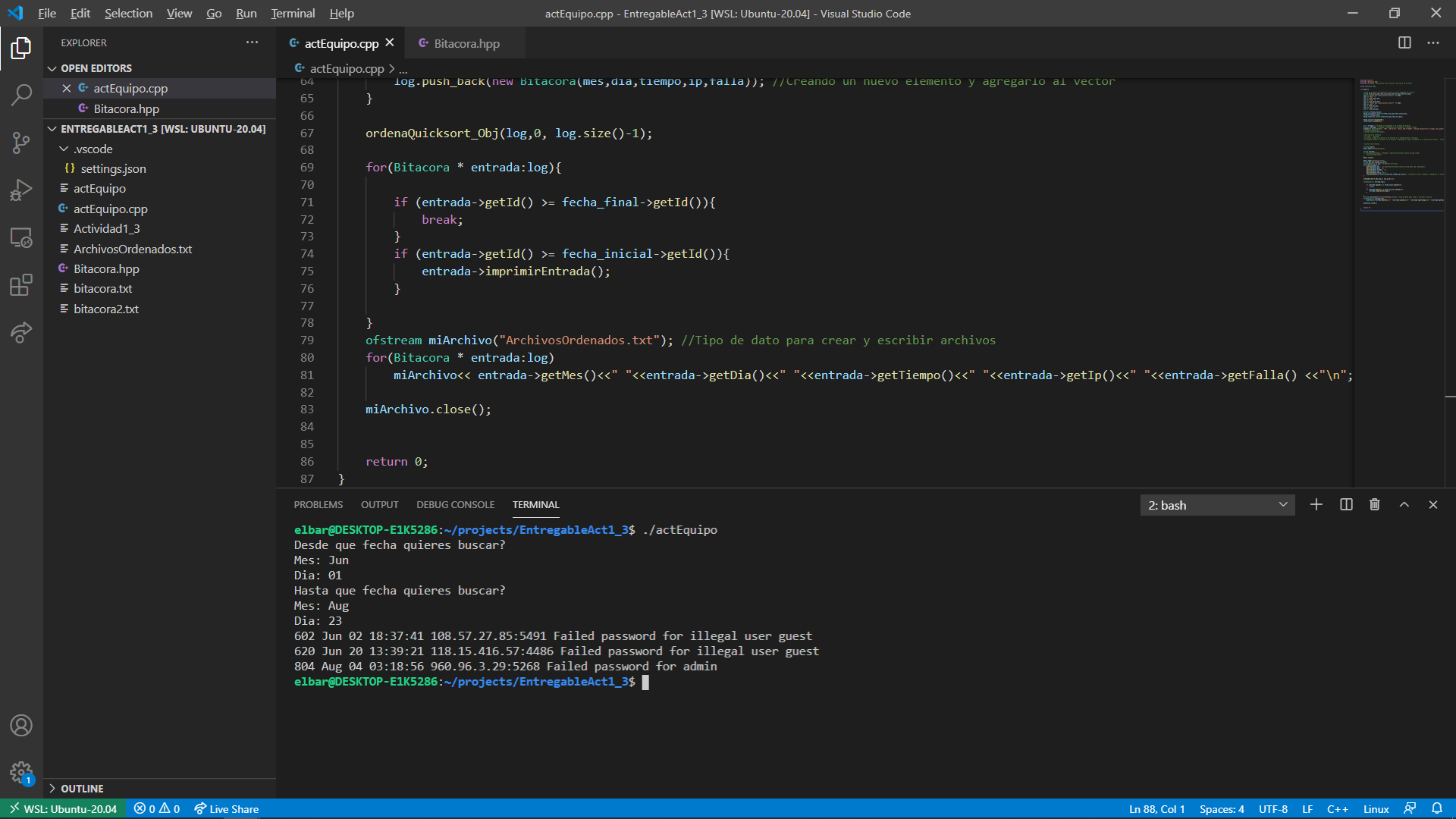
### **Víctor Adrián Sosa Hernández**

07/10/2020

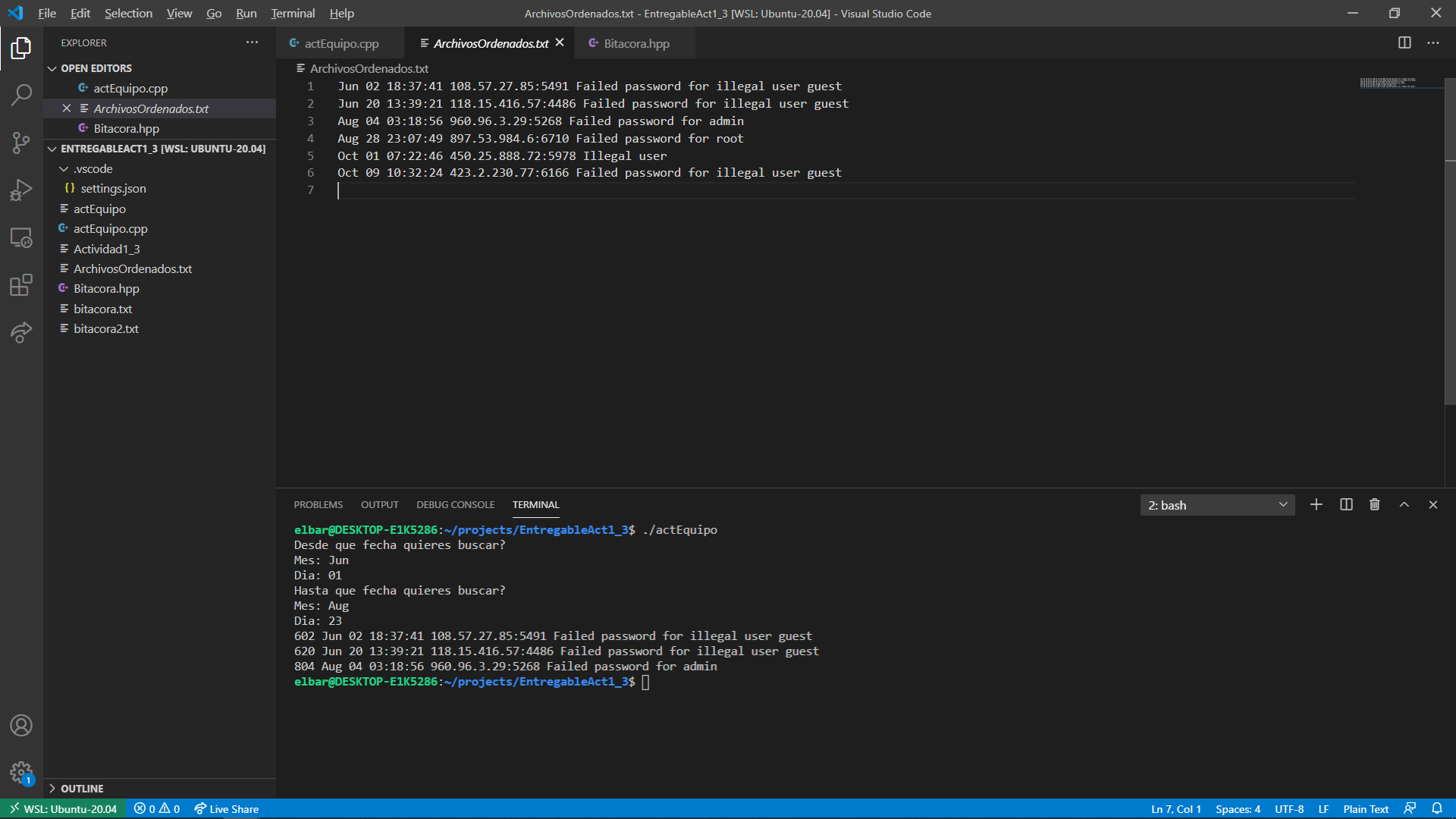
Investigación:

La importancia y eficiencia de los algoritmos de ordenamiento y búsqueda tiene que ver con cuantas operaciones tiene que hacer un algoritmo para realizar la tarea que es mandada a hacer. En términos de Big O podemos determinar qué algoritmo es más eficiente, teniendo en cuenta el número de operaciones realizadas y su dificultad (del mejor caso posible al peor caso posible).

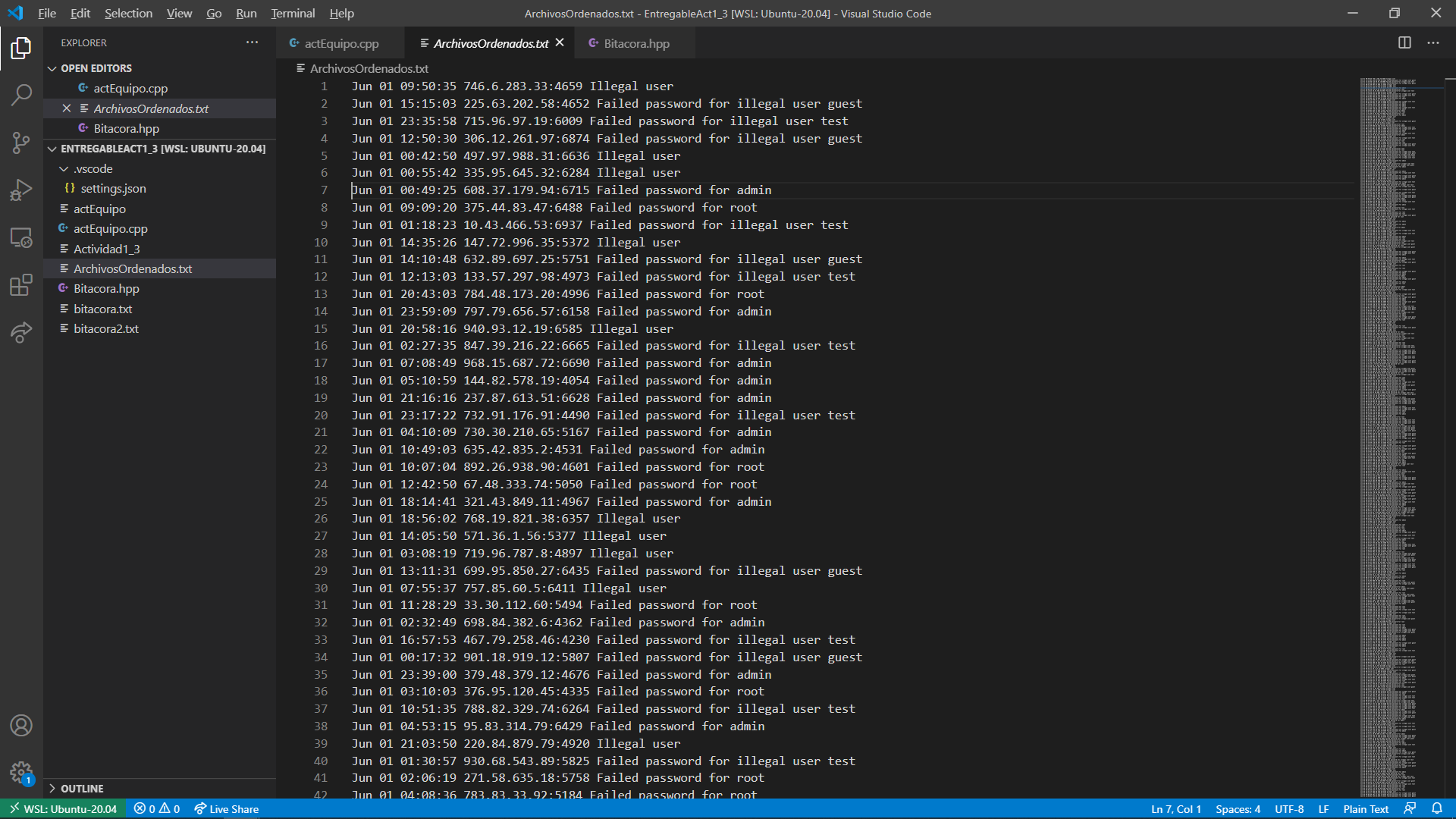
Existen diferentes algoritmos de ordenamiento y de búsqueda. Los algoritmos de ordenamiento: por selección y reemplazo con un , Burbuja , Quicksort , entre otros. Entonces la eficiencia de los algoritmos está directamente vinculado a las operaciones definidas por el .

Prueba de buscar fechas

Archivo nuevo ordenado(bitácora.txt).



Archivo nuevo ordenado(bitácora2.txt).



**Complejidad de código**

La complejidad de este código parte de la propia complejidad del algoritmo de búsqueda Quicksort el cual tiene como complejidad BigO(n^2), para conseguir encontrar la complejidad de este código se tiene que tener en cuenta la complejidad de la clase Bitácora así como las recursiones que están en el código principal. La una parte del código que le agrega complejidad al algoritmo de quicksort son las recursiones de “while(datos.good())” la cual tendría una complejidad de BigO(n) al igual que la otra recursion “for(Bitacora \* entrada:log)”. Al tomar en cuenta estas complejidades y sumarlas a la complejidad de Quicksort nos terminaria dando un BigO(n^2) debido a que esta sigue teniendo el mayor exponente y las otras complejidades no suponen un incremento de complejidad en términos de BigO para cambiar la complejidad de Quicksort.

**Reflexion individual:**

*Carlos Antonio Buendia López:*

Esta es una actividad que me ayudó a comprender diferentes algoritmos y métodos de busca y ordenamiento que existen. Ayudó a comprender de una mejor manera lo visto en clase. Ya que tanto mi compañero Luis Ángel Terrazas García y yo modificamos el código de quicksort visto en las clases anteriores. El modificar el código fue un reto para ambos ya que necesitamos investigar más a fondo sobre cómo solicitar un valor de un objeto específico en el vector de objetos, para que nuestro algoritmo funcionará únicamente con valores de números enteros ya que el lenguaje de programación C++ es altamente tipado.

*Luis Ángel Terrazas García:*

Esta actividad fue una gran oportunidad para conseguir implementar los algoritmos vistos en clase de una manera práctica al ordenar archivos de texto. Esta actividad fue un reto debido a que necesitábamos encontrar la forma de modificar el código anterior de Quicksort para que pudiera recibir un vector de objetos y ordenarlo, debido a que nuestro anterior código de quicksort no tenía en cuenta este cambio se necesito aprender como se puede pedir un valor de un objeto específico en un vector de objetos y así conseguir extraer ese valor y utilizarlo en nuestro algoritmo de quicksort el cual funciona sólo con enteros.

Referencias:

1. Michael T. Goodrich and Roberto Tamassia y Ing Estevan Gomez,MSc. (N/A). *Algoritmos de ordenación y búsqueda*. 07/10/2020, de GoogleSitio web: <https://sites.google.com/a/espe.edu.ec/programacion-ii/home/a1-arreglos/algoritmos-de-ordenacion-y-busqueda>
2. Anónimo. (N/A). *Ordenamiento*. 07/10/2020, de N/A Sitio web: <https://users.dcc.uchile.cl/~lmateu/CC10A/Apuntes/ordenamiento/index.html>
3. Teddy Alfaro Olave. (N/A). *1 Algoritmos de Búsqueda y Ordenamiento*. 07/10/2020, de Departamento de Informática Universidad Técnica Federico Santa María Sitio web: <https://www.inf.utfsm.cl/~noell/IWI-131-p1/Tema8b.pdf>