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

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

# ***Act 3.2 - Actividad Integral de BST (Evidencia Competencia)***

**Alumno:**

Carlos Antonio Buendia López A01379471

Luis Ángel Terrazas García A01377440

**Profesor(a):**

### **Víctor Adrián Sosa Hernández**

27/11/2020

Investigación:

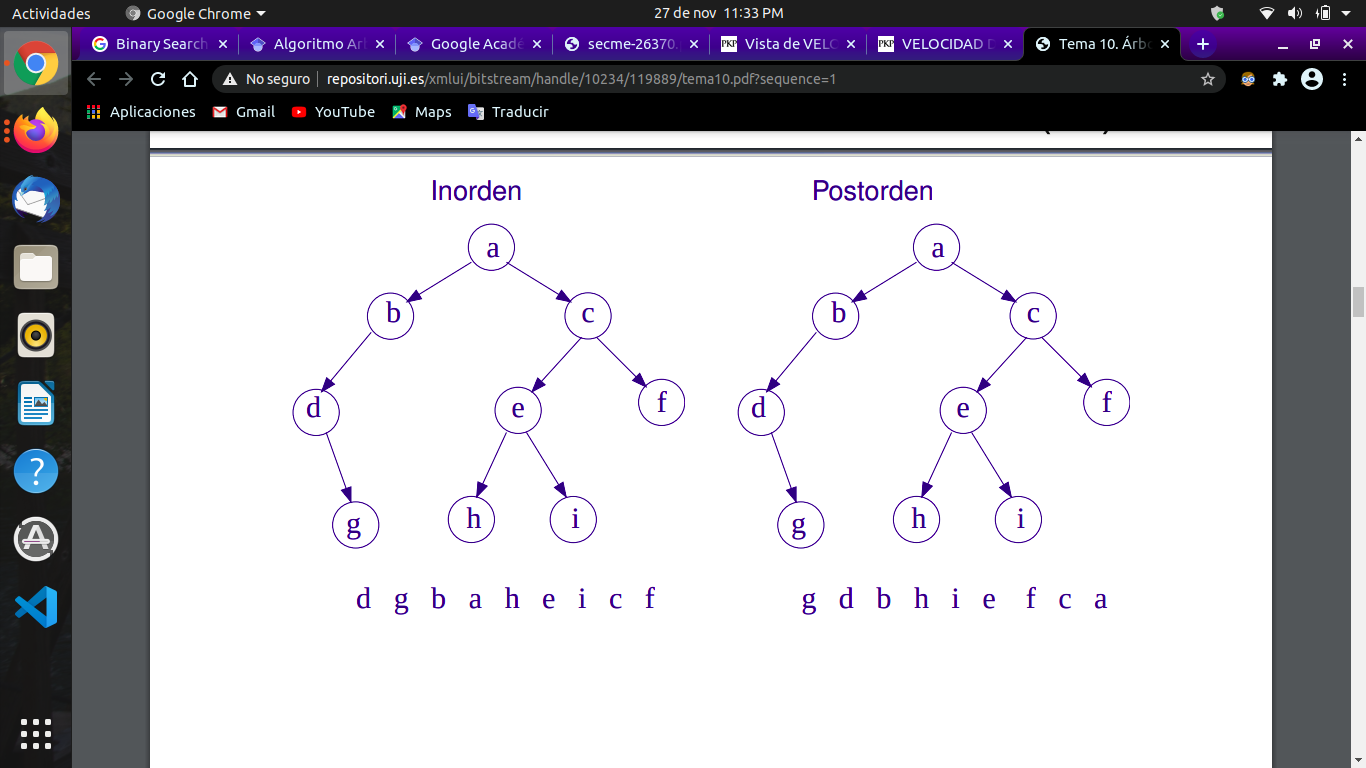
El algoritmo de *BST (Binary Search Tree)* o árbol de búsqueda binaria cuya eficiencia en términos de Big O es de en el peor de los casos. La gestión de una enorme cantidad de datos mediante el acceso lineal deja de ser eficiente. Siendo que los árboles tienen una estructura no lineal y cuya organización de los datos es jerárquica a lista enlazadas(padre, hijo, hermanos).

Un árbol es un conjunto limitado de nodos y a su vez un conjunto limitado de ramas; de tal forma que el nodo principal se denomina nodo raíz, los otros nodos se dividen en de conjuntos disjuntos, donde cada es a la vez un subárbol de la raíz. Las ramas se conectan a un par de nodos. Un nodo se conforma de información y las ramas son las responsables de unirlos con otros nodos. Cualquier árbol de nodos tiene ramas. Si un nodo cuenta con una rama hacia otro nodo , entonces se dice que este es el hijo (h) de por lo tanto este es el padre de Cada nodo cuenta con un único padre a excepción del nodo raíz que no tiene y dos nodos son hermanos si tienen el mismo padre. Un nodo hoja es aquel que no tiene hijos. Un camino está definido porque si un nodo al nodo por lo que se le denomina una secuencia de nodos distintos entre sí de tal manera que existe una rama conectada a cada par de nodos consecutivos cuya longitud es definida por el número de ramas que contiene.

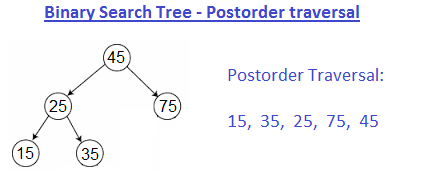
Existen diferentes tipos de árboles binarios los cuales son el equilibrado porque para todos sus nodos se cumple ; extendido porque todos sus nodos tienen 0 ó 2 hijos no vacíos; completo de profundidad , todos los nodos que son hojas se encuentran en un nivel y además el números de nodos es de lo cual representa el número máximo de nodos para un árbol con profundidad

Los recorridos de los árboles binarios son:

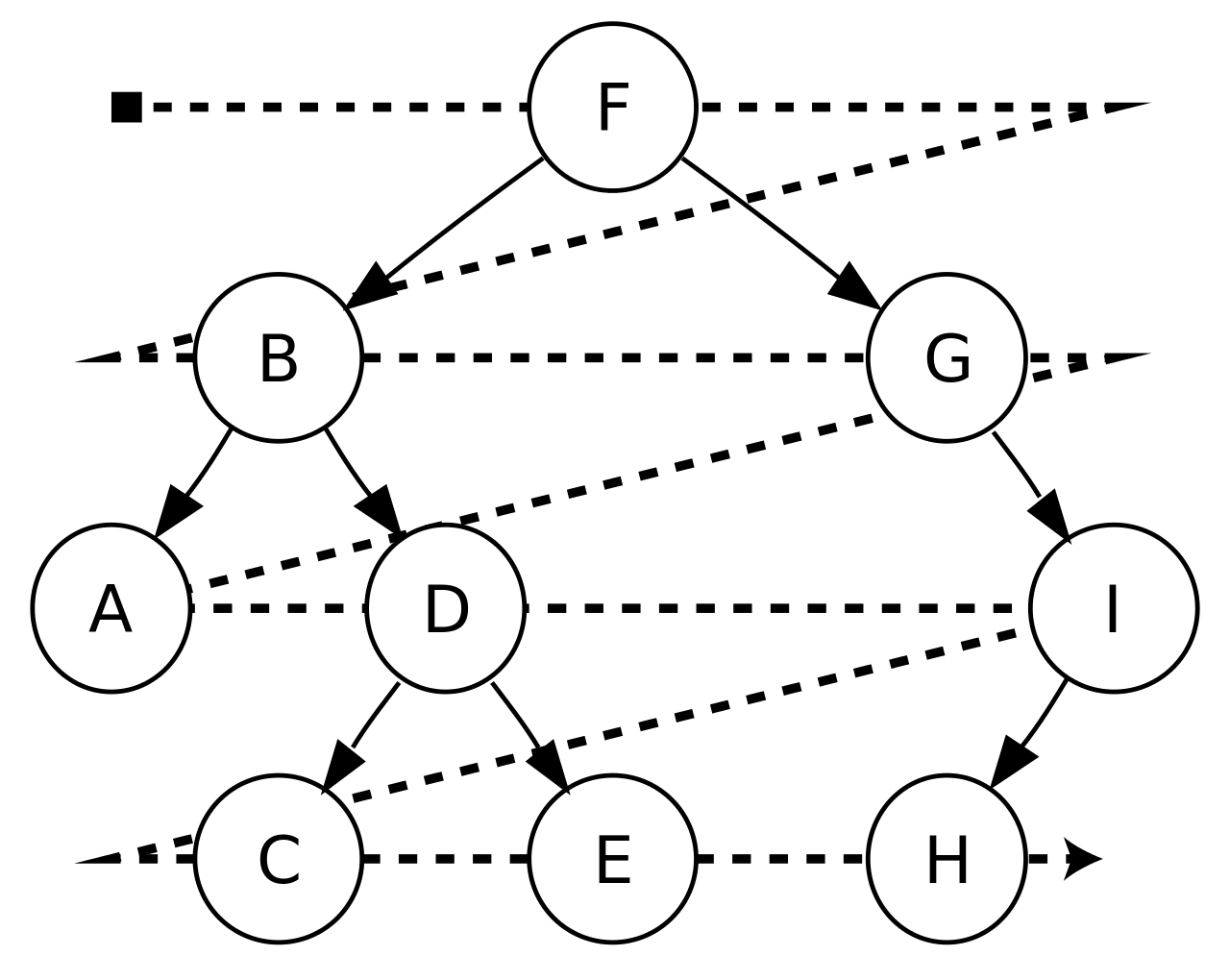
* Preorden (presenta en la raíz, pasa por el preorden del subárbol izq; y el preorden del subárbol der.)
* Inorden(pasa en inorder del subárbol izq; se presenta en la raíz, pasa en inorder del subárbol der.)



* Postorden (pasa en postorden del subárbol izq; pasa en postorden del subárbol der; presenta en la raíz).

1. 

* Por niveles (los nodos del nivel se presentan antes que los nodos del sig. nivel , este no es recursivo ya que se hace uso de una cola aux.)

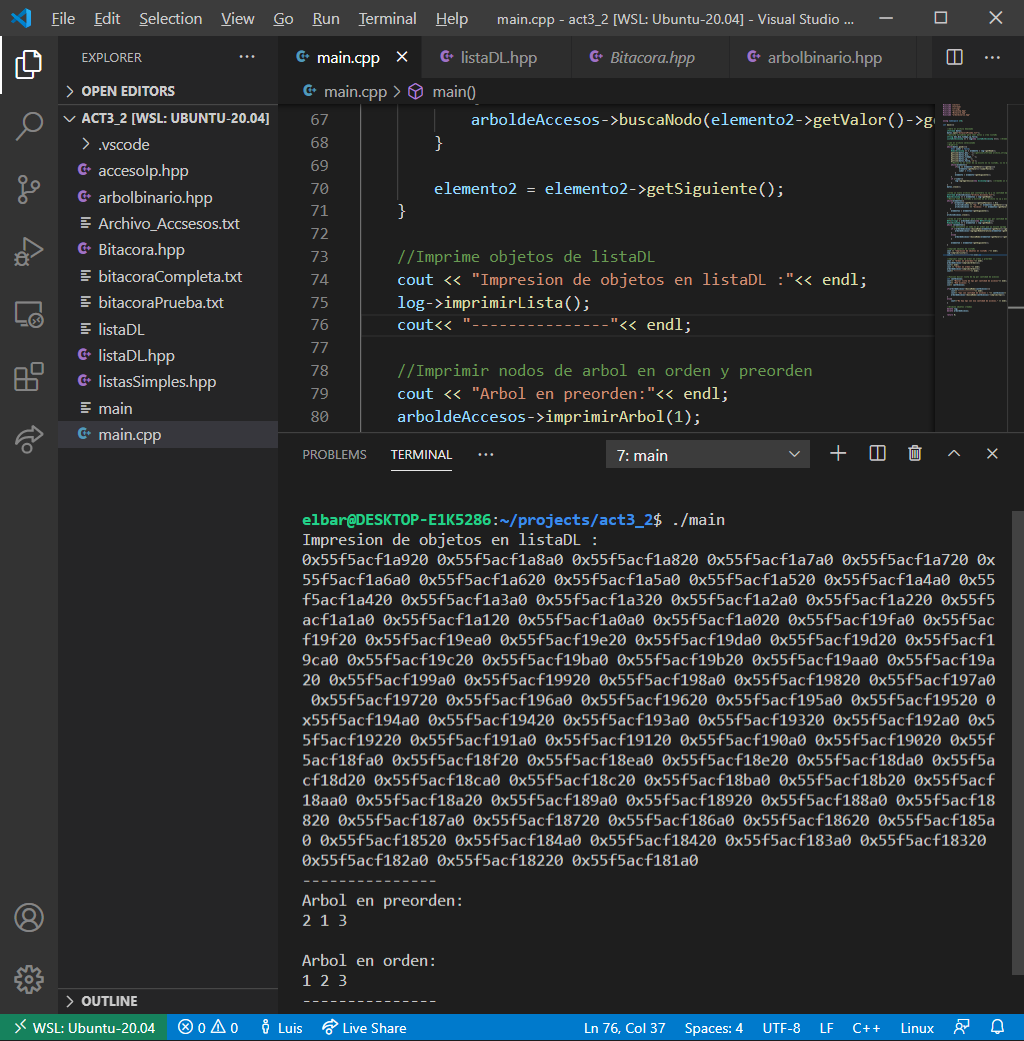


Un montículo o *heap* en inglés es un árbol binario cuyas características son: es un árbol binario semicompleto; para cada nodo de un montículo, el valor que almacena es mayor o igual que el valor almacenado en sus nodos hijos, etc. Cuyas aplicaciones son la implementación de las colas de prioridad y el ordenamiento de los elementos. Y las operaciones que realizan son insertar, leer (elemento situado en la raíz) o borrar el elemento mayor o menor.

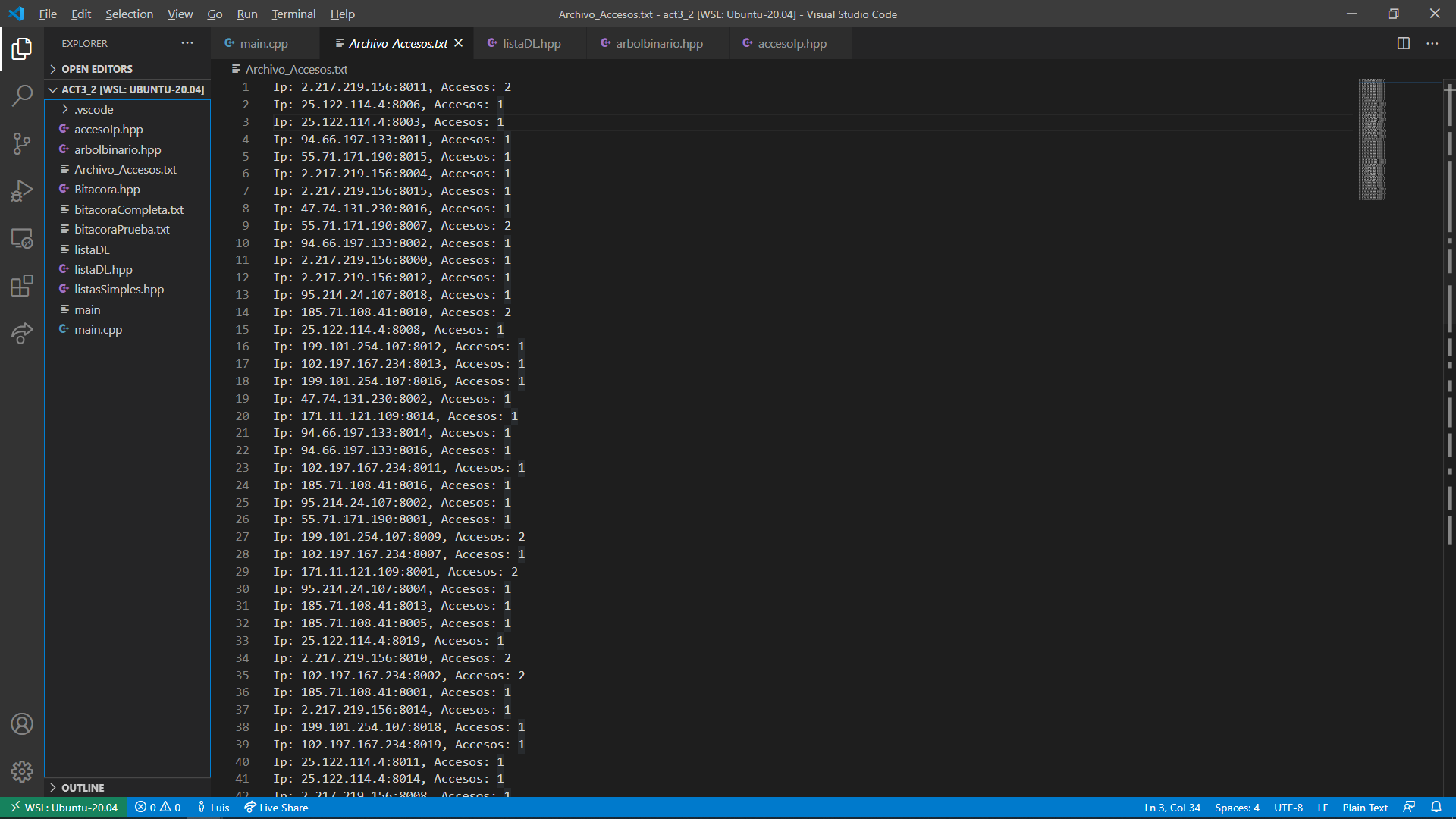
La importancia de el uso de los árboles de búsqueda binaria es grande cuando se necesita hacer algún tipo de búsqueda por jerarquía, también son bastante útiles para conseguir un tiempo de búsqueda bastante bueno, en promedio estos tiene una complejidad de O(log2N). Por lo general los BST son muy utilizados por bases de datos, diccionarios, caches, etc.. Siempre existirán una base de datos donde sea más óptimo su uso.

Generalmente una red puede ser identificada como una red infectada sabiendo la proveniencia de las Ips, cuando se intenta infectar una red generalmente es atacada por un número de dispositivos los cuales envían de manera continua señales al servidor. Se puede determinar si una red está infectada si esta tiene una gran cantidad de accesos provenientes de la misma ip ya que esta es la manera más común de hacer un “raid” a un servidor.

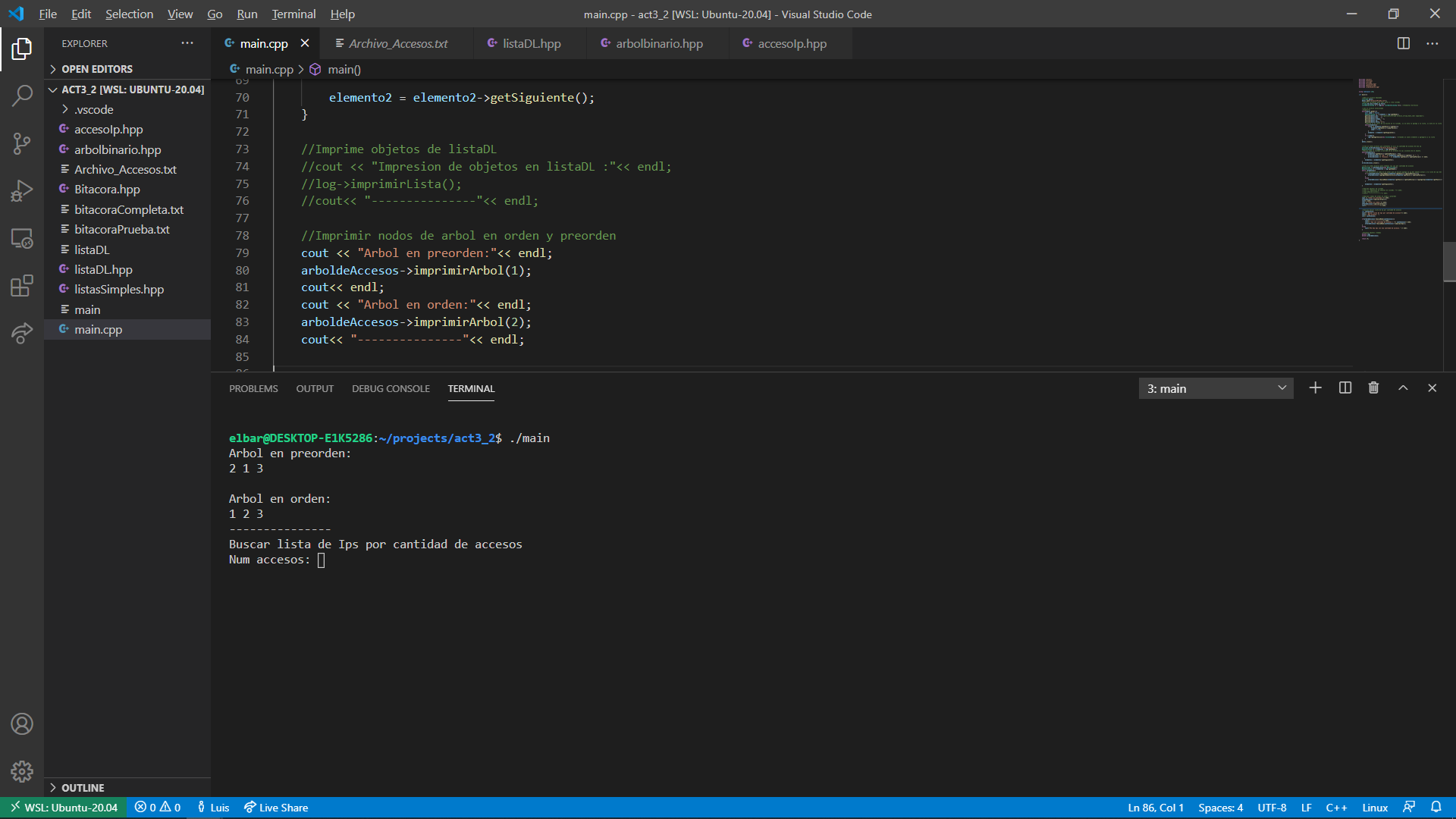
Captura de Pantalla de la Impresión de Lista Simple:



Archivo generado con ip y cantidad de accesos

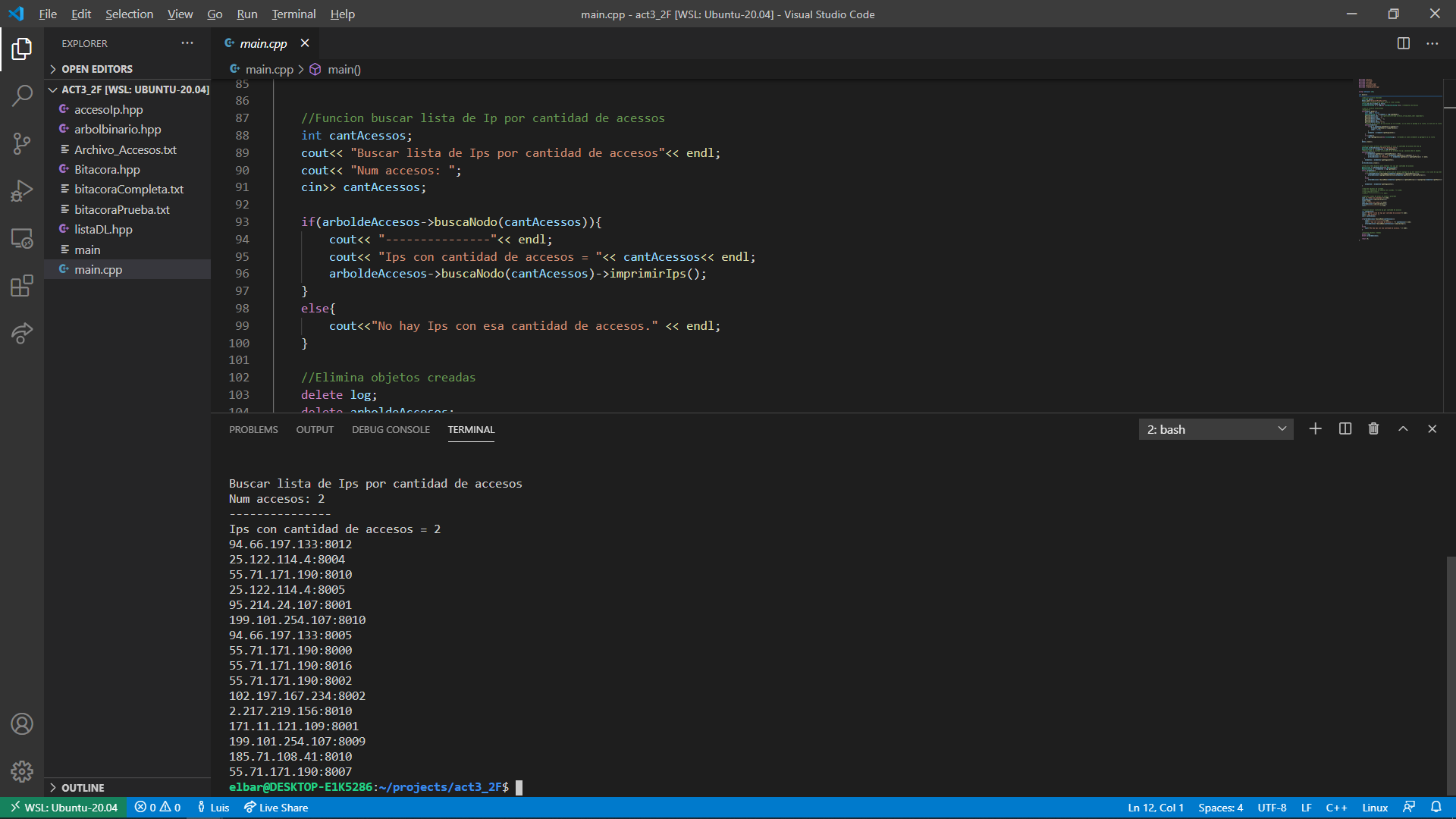


El recorrido en preorden y orden de los números de accesos únicamente:

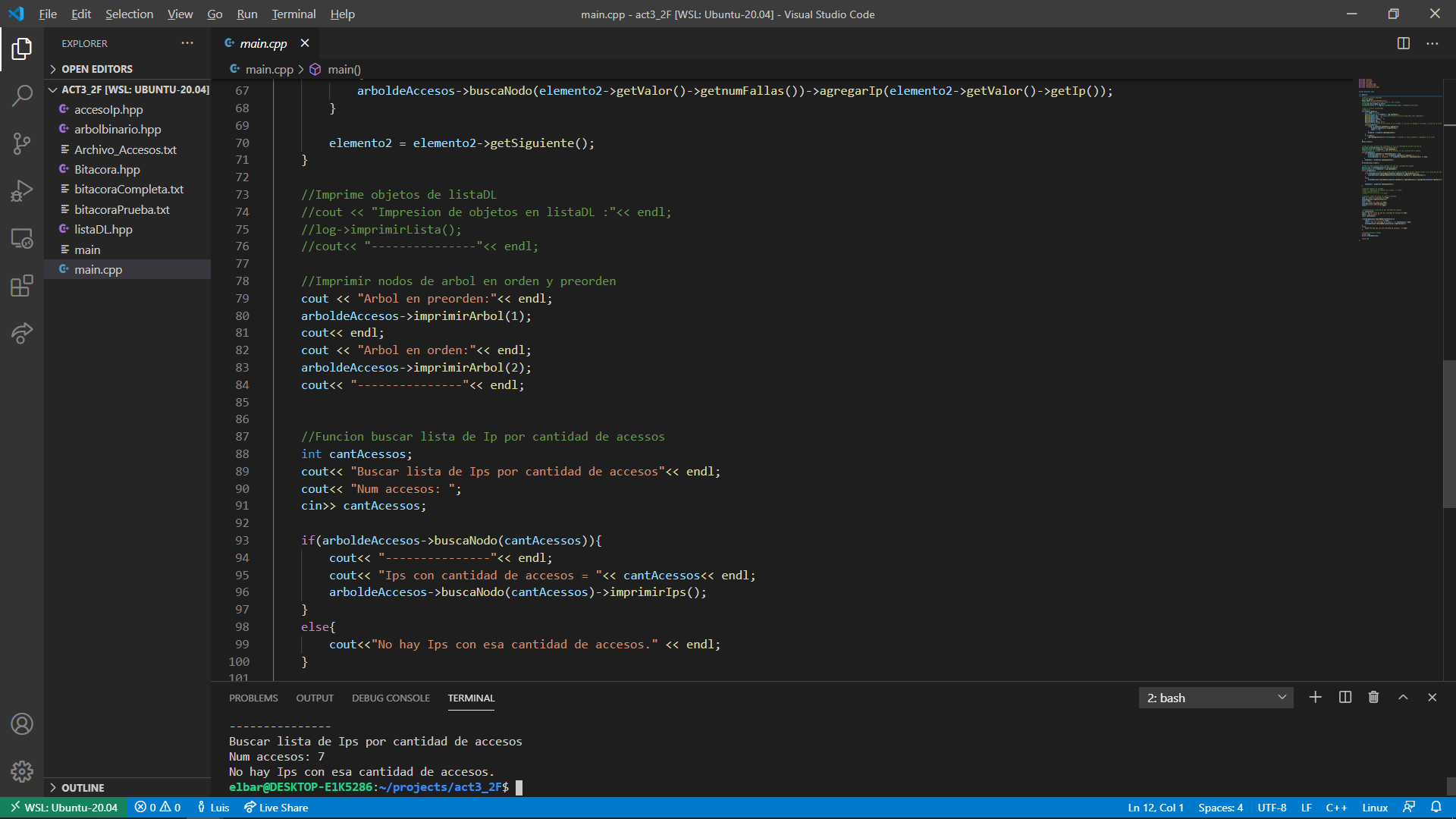


2 Ejemplos de realizar la búsqueda por número de accesos utilizando el BST y si se encuentra la llave o identificador deberá imprimir la lista de dirección IP's con esa cantidad de accesos:

Ejemplo 1:

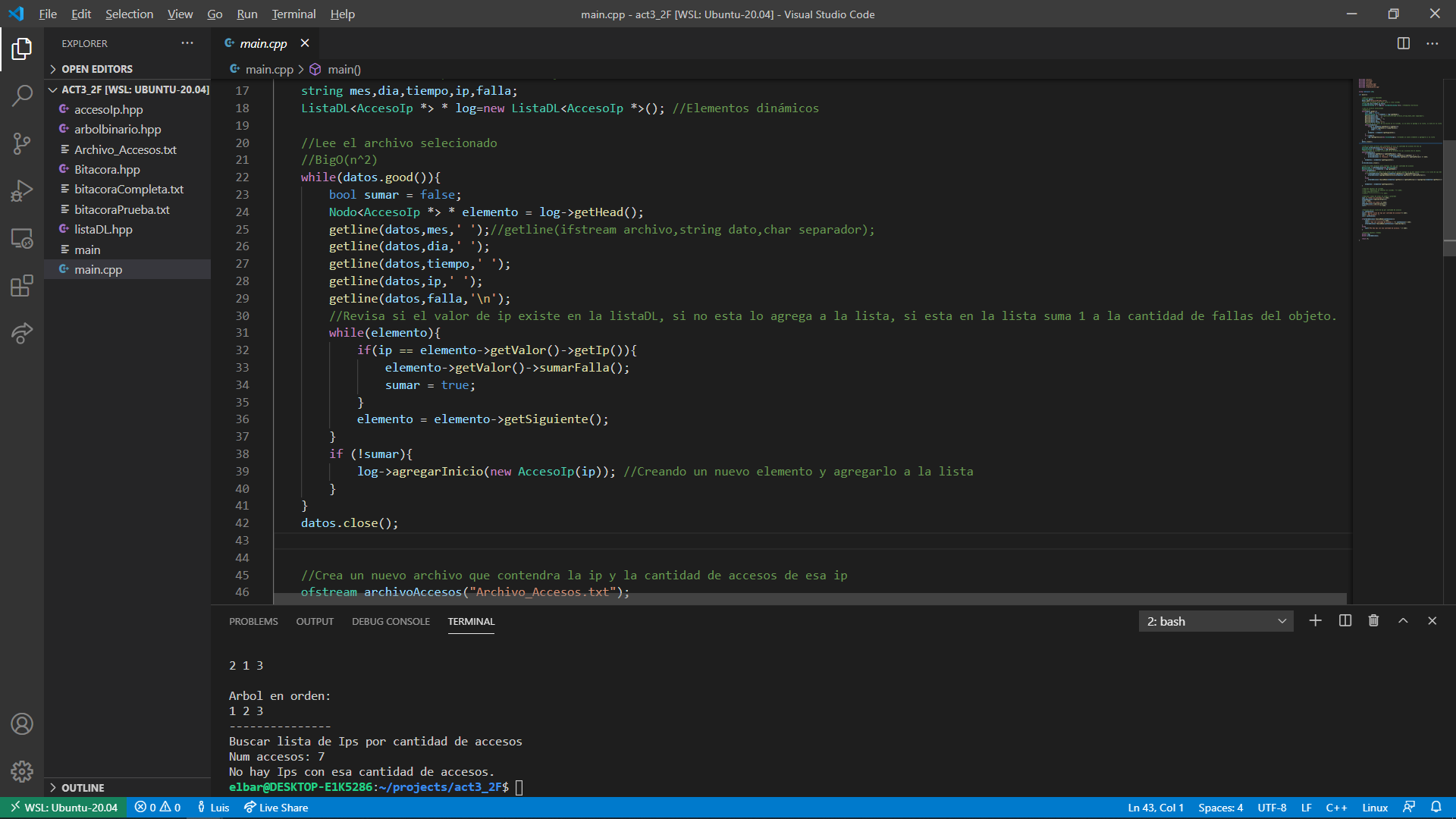


Ejemplo2:



Complejidad del código:

Este código en general no utiliza funciones que lleguen a ser más grandes que BigO(n), los procesos de búsqueda de nodo en el bst son BigO(n), así como las funciones de agregarNodo, getValor, imprimirLista, getSiguiente, etc.. Este código tiene una complejidad de BigO(n^2) a causa de la lectura de datos y agregación de objetos a la listaDL.



Esta parte del código le da la complejidad de Big O(n) debido a que por cada lectura de línea en el archivo seleccionado este código tiene que revisar toda la listaDL para revisar si la ip se repite. A causa de esta parte del código tenemos dos loops while que en el peor de los casos tiene complejidad de Big , debido a esto nuestro código tiene la complejidad de Big .

Conclusión:

* Luis Ángel Terrazas García:

En esta entrega tuvimos que implementar todos los conocimientos previos de anteriores entregas y utilizarlo en conjunto con un árbol de búsqueda binaria. Si tuviera que decir el reto más grande al que nos tuvimos que enfrentar sería la parte del manejo de una lista doblemente ligada dentro del nodo árbol. La implementación de este programa me pareció más fácil que las anteriores, al ser solo una parte del código la creación de un bst me pareció que conseguimos hacer todos los otros puntos de esta actividad bastante más rápido.

* Carlos Antonio Buendia López:

Es una actividad que con la guia del profesor Victor, los trabajos, e investigación que anteriormente hemos realizado mi compañero y su servidor. Logramos completar los puntos de esta entrega de una forma más eficiente. La complicación de esta entrega fue la implementación de la lista doblemente ligada adentro del nodo árbol. También algo interesante sobre esto es como estos algoritmos ayudan a determinar si una red está o no infectada.

**Referencias**:

1. Ramos, E. A. H., & Vizcarra, H. M. B. (2017). *VELOCIDAD DE RESPUESTA DE LOS ALGORITMOS DE BÚSQUEDA DE DATOS CONTENIDOS EN ESTRUCTURAS ESTÁTICAS Y DINÁMICAS*. *Ciencia & Desarrollo*, (21), 65-72.
2. Badía Contelles, J. M., Martínez Salvador, B., Morales Escrig, A., & Sanchiz Martí, J. M. (2012). *Tema 10. Árboles*. Se encuentra en el sitio web: [Tema 10. Árboles](http://repositori.uji.es/xmlui/bitstream/handle/10234/119889/tema10.pdf?sequence=1).
3. E. Horowitz, S. Sahni, D. Mehta. (1995). *Fundamentals of Data Structures in C++*. Computer Science Press 1995. Capítulos 5, 7.6, 9, 10.1 y 10.2.
4. Hinojosa Ramos, E. A. (2014). Velocidad de respuesta de los algoritmos de búsqueda de datos contenidos en estructuras estáticas y dinámicas. Se encuentra en el sitio web: <http://www.revistas.unjbg.edu.pe/index.php/cyd/article/view/733/745>