1. **Titulo.**

   Covid-19 case tracker Colombia

2. **Integrantes.**
   - Sebastián Ibarra Méndez.
   - Sebastián Suárez García.
   - Camilo Andrés Tejada Merchán.

3. **Repositorio.**

   https://github.com/sebastianssg/Covid-19_case_tracker_America-

4. **Resumen ejecutivo.**

Teniendo en cuenta la pandemia que se vive actualmente a nivel mundial a causa del Covid-19 se decide realizar una base de datos que dé cuenta de los casos registrados en Colombia en la cual se puedan hacer búsquedas filtrando los datos, teniendo en cuenta tres características importantes, a saber, el género, la edad y la localización de los individuos. Es importante destacar que la localización se divide en 33 partes correspondientes a los 32 departamentos de Colombia y Bogotá D.C.

5. **Funcionalidad y descripción de la herramienta.**

El usuario ingresa los datos de búsqueda, es decir, el género (true para femenino y false para masculino), el rango de edad y la localización. Luego con los criterios de búsqueda ingresados se filtra la información que se encuentra en el archivo llamado datasetfinal.txt de la siguiente manera: 5929,27,F,Boyacá,15 donde 5929 es el ID único, 27 es la edad, F es el género (en este caso femenino y M para masculino), Boyacá es la locación y 15 es el ID del departamento. Los datos se encuentran separados por comas de forma que al recorrer las líneas de información se pueda obtener el respectivo dato.

Una vez hecho esto se hace uso de GenderMap donde se genera un vector de números enteros que contiene los IDs que satisfacen la condición dada por el usuario. Seguidamente, haciendo uso del árbol binario BinLoc donde la llave es la edad y el valor es el ID único se usa el vector “rango” cuyos valores [Desde, Hasta] están dados por el usuario donde se buscan las edades que se encentren en dicho rango, acá se hace uso de la constante NEXT que permite seguir buscando en el vector en caso en que el valor inicial o final del rango no se encuentren; al terminar la búsqueda se retorna el ID de la persona.

Luego en DptoSet se tiene un vector de tipo set de tipo entero en el cual se guardan las 33 ubicaciones. Para ingresar un elemento a un set en especial se usa el ID del departamento. Ahora bien, si el usuario requiere información referente a más de un departamento se debe hacer la unión de los datos, la cual se almacena en un vector padre el cual se recorre al final para generar un vector de retorno.

Posteriormente teniendo los tres vectores resultantes, es decir, el del género, la edad y la locación se procede a ordenar el vector del género haciendo uso de bubble sort, cabe resaltar que los otros dos vectores ya están organizados. Así que, empleando binary search se realiza la intersección entre el vector de las edades y el del género, guardando estos valores de la intersección en otro vector el cual se interseca con el vector de las ubicaciones.

Finalmente, los datos obtenidos se guardan en un archivo de texto llamado vis.txt el cual se usará para visualizar los datos con un código de Python haciendo uso de matplotlib.

6. Algoritmos y estructuras de datos.

- Árbol binario: Llamado BinLoc. Esta estructura contiene los métodos convencionales tales como: empty e insert_node. Se seleccionó esta estructura por sus propiedades de árbol binario que permiten hacer la búsqueda de elementos más sencilla y óptima. Otra característica importante es que se usaron los métodos predecesor y sucesor (que a su vez usan minimun y maximun) con el fin de desarrollar el método rango, que es el que al final hace la búsqueda con respecto al filtro de edad.
- Mapa tipo hash table: Llamado GenderMap, tiene los métodos convencionales.
- Clase DptoSet: Consta de un total de 33 sets referentes a los 32 departamentos de Colombia y Bogotá. Estos se guardan en un vector general para poder hacer la inserción a departamentos de manera más fácil sin necesidad de tener que llamar a cada uno de estos. Adicionalmente, se utilizó esta estructura con el fin de poder hacer el filtro de manera más sencilla.
- Algoritmo de búsqueda binaria: Al final para poder hacer una intersección de todos los elementos de cada uno de los vectores que se generan de cada una de las estructuras (BinLoc, GenderMap y DptoSet) se optó por utilizar el binary search con el fin de tener una complejidad de log(n).
- Algoritmo de ordenamiento: Para poder implementar la búsqueda binaria hubo ciertos vectores que no estaban organizados como tal luego de su generación, éste fue el caso del vector generado por el mapa de géneros para esto se usó el bubble_sort para así proceder a intersecarlos entre sí. Por otro lado, también se usó el bubble_sort al final, para poder generar el archivo de tipo texto con las locaciones de las personas que satisfacen los parámetros dados por el usuario.