TRABAJO PRACTICO 2º

Algoritmos y Programación I – 75.40 - FIUBA

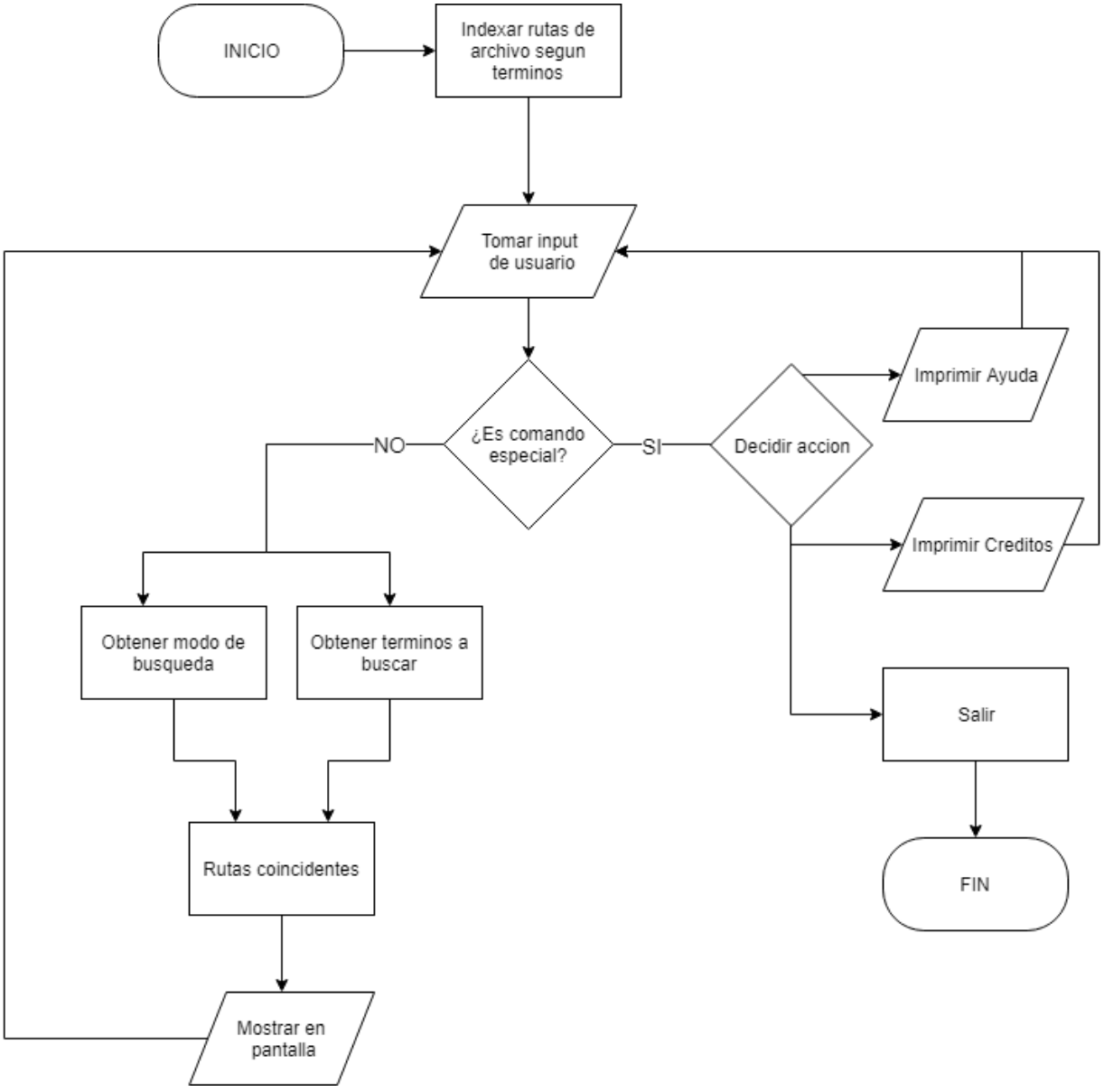
2º cuatrimestre - 2017

Franco Giordano - 100608

Bajo la consigna de implementar un motor de búsqueda de archivos, se consideraron las siguientes condiciones:

- Ofrecer un prompt capaz de interactuar con el usuario
- Disponer de un motor de búsqueda, capaz de utilizarse en tres modos distintos: OR, AND y NOT.
- Agilizar el proceso mediante un índice invertido, dispuesto de términos con sus respectivas rutas.
- Ignorar mayúsculas/minúsculas, tanto en el input como en el funcionamiento interno del programa.
- Analizar contenido de archivos .py, .md, .txt o .c
- Realizar la indexación de términos UNA SOLA VEZ, minimizando la utilización de recursos.

A grandes rasgos, el flujo del programa será el siguiente: **(A)**

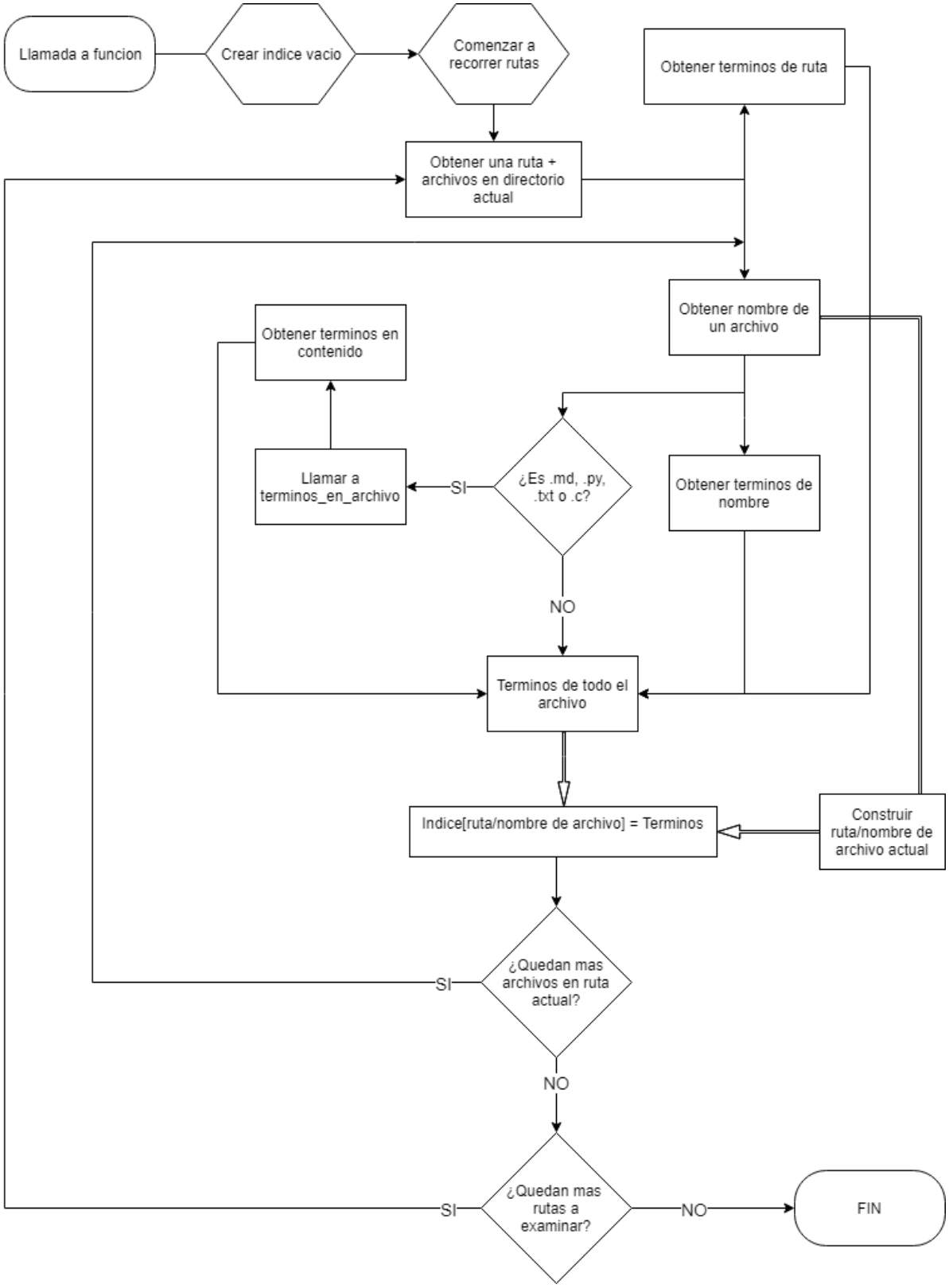


La construcción del índice invertido fue uno de los primeros diseños a realizar. Para ello se pensó en la utilización de un ciclo definido, donde se recorrerán las *rutas al alcance del algoritmo* (1) y se indexarán las mismas. Para agilizar el proceso, se construirá un índice con las rutas y cada una con sus respectivos términos hallados en su dirección, nombre y, de ser de la extensión adecuada, en su contenido.

(1): "rutas al alcance del algoritmo" serán aquellos directorios y todos los subdirectorios que se encuentren en la ubicación desde donde se ejecuta el programa.

Esto es de suma importancia para la futura implementación ya que se estarán recorriendo rutas de archivos, no términos, por lo que **su construcción en el momento debe ser lo más intuitiva y rápida posible**. Para luego interactuar con el usuario, **se utilizará un algoritmo capaz de invertir índices**, utilizando este nuevo índice invertido para las búsquedas.

Para la indexación de archivos, se utilizará un algoritmo como el siguiente: **(B)**



Como se puede observar en (B), la lectura de archivos ocurre SÓLO al llamar términos_en_archivo, dándose que sea de la extensión adecuada. La lectura de "datos" del sistema operativo (rutas, nombres de archivo) ocurre a lo largo de toda la función. Conociendo esto, y volviendo a (A), se encuentra que la lectura de archivos ocurre solo en el caso mencionado con anterioridad.

Con la intención de ignorar mayúsculas y minúsculas, se transformarán todos los datos obtenidos a minúsculas en cuanto sean obtenidos. Así, se evitarán posibles disparidades que puedan surgir con el desarrollo del programa, ya que se comparará activamente los datos indexados con el input del usuario.

Para la recolección de términos *dentro* de los archivos, se leerá línea por línea con la intención de poder manejar archivos que no entren en memoria. Por otro lado, para representar estos datos en memoria utilizando Python 3, se utilizarán diccionarios para los índices (por ruta e invertido) y listas en casos en los que se necesite, como las rutas asociadas a un término o rutas coincidentes con una búsqueda dada.

Luego de construir el índice e invertirlo, comienza la interacción con el usuario. Se le presentará un prompt ("> ") donde podrá detallar sus términos de búsqueda o incluso elegir un modo especifico. De esta forma, comienza la búsqueda en el índice invertido previamente construido.

Como cada modo posee su particularidad, se deberá implementar cada uno por separado con algoritmos fundamentalmente distintos. En todos los casos se añadirán todas las rutas coincidentes correspondientes, eliminando luego aquellas repetidas. Esto se hará con la intención de minimizar la repetición de código.

Asumiendo como caso general la búsqueda OR, se encontrarán los términos pedidos en el índice invertido añadiendo sus rutas a una lista. En caso de no encontrarse el termino, simplemente se lo ignora.

De haberse elegido la búsqueda AND, primero se chequeará que cada término a buscar se encuentre en el índice invertido. De no ser así, se devolverá que no hay coincidencias. Asumiendo ahora que se encuentran todos los términos, para la primera iteración, se copiarán las rutas del primer término y estas serán "candidatas" a rutas coincidentes. Así, por cada iteración, se buscarán aquellas rutas que sean tanto candidatas anteriores como rutas encontradas en el término actual, y se descartarán las rutas que no cumplan.

Finalmente, para la búsqueda NOT, se encontrarán las rutas del termino dado siendo éstas las "no válidas". A continuación, se tomarán todas las rutas disponibles en el índice invertido como "candidatas" y se procederá a descartar aquellas que no sean válidas.