**Algorithm Analyze**

Manual de usuario

**Brenes Maleaño Andrés Ottón**

**Fernández Jiménez Axel Alejandro**

**López Saborío Iván**

**Prefacio**

Este documento es una guía para que el usuario utilice de manera correcta la aplicación creada. Se incluirán imágenes ilustrativas con las que el lector podrá relacionar la información de una manera más sencilla. Es importante saber que todo lo que se indica es este manual es con fines académicos. El sistema en el cual se basa el manual fue realizado por Iván López Saborío, Axel Fernández Jiménez y Andrés Brenes Maleaño, para el proyecto 1 del curso de Análisis de Algoritmos de la Escuela de Ingeniería en Computación.

**Contenido**

[1. Funciones principales 4](#_Toc496122434)

[1.1. Instalación Python 3.4 4](#_Toc496122435)

[1.2. Abrir la interfaz de la aplicación 5](#_Toc496122436)

[1.3. Utilizar los algoritmos 6](#_Toc496122437)

[1.4. Datos a introducir 7](#_Toc496122438)

[1.5. Salir del programa 8](#_Toc496122439)

[1.5. Reutilizar un algoritmo 8](#_Toc496122440)

[2. Preguntas Frecuentes 9](#_Toc496122441)

[3.1 ¿Se puede utilizar un algoritmo más de una vez? 9](#_Toc496122442)

[3.2 ¿Se pueden subir mas fotos por avistamiento? **¡Error! Marcador no definido.**](#_Toc496122443)

[3.3 ¿Cuántos avistamientos se pueden realizar por día? **¡Error! Marcador no definido.**](#_Toc496122444)

[3.4 ¿Se pueden hacer varios avistamientos del mismo ave? **¡Error! Marcador no definido.**](#_Toc496122445)

[3.5 ¿Se pueden agregar especies nuevas de aves? **¡Error! Marcador no definido.**](#_Toc496122446)

[3.6 ¿Cómo puntuar otros avistamientos? **¡Error! Marcador no definido.**](#_Toc496122447)

[3.7 ¿Pueden dos personas realizar el mismo avistamiento? **¡Error! Marcador no definido.**](#_Toc496122448)

[3. Glosario **¡Error! Marcador no definido.**](#_Toc496122449)

[4. Créditos **¡Error! Marcador no definido.**](#_Toc496122450)

# **1. Funciones principales**

## 

## **1.1. Instalación Python 3.4**

Para hacer uso de la aplicación es necesario tener instalado el intérprete de Python en la versión 3.4 como mínimo. Para descargar este importante elemento, vaya a la página oficial de Python: <https://www.python.org/downloads/>

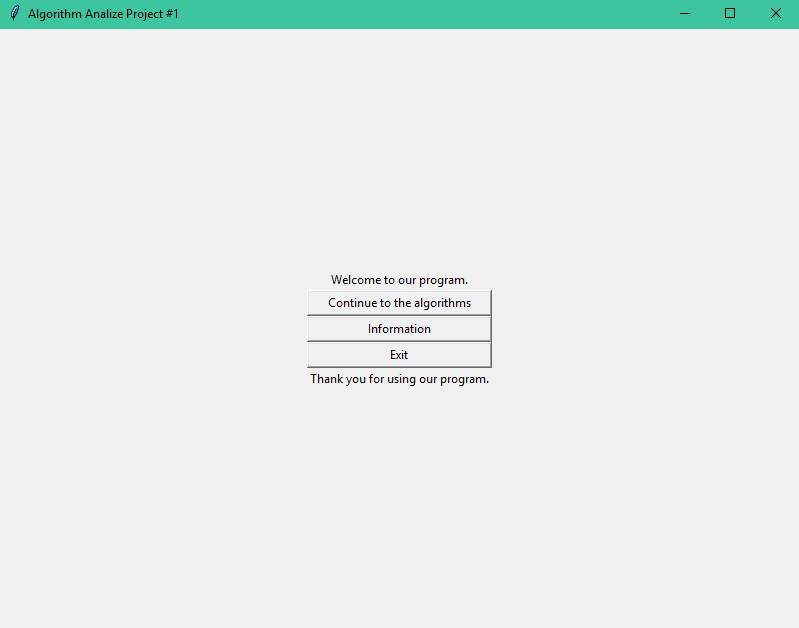


## **1.2. Abrir la interfaz de la aplicación**

Para utilizar la aplicación solo debe interactuar con el archivo llamado “Interface.py”, no nos hacemos responsables si altera algún otro archivo.

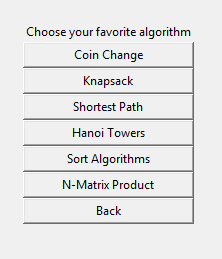
Los pasos para entrar a la interfaz de la aplicación son los siguientes:

* Click derecho en el archivo “Interface.py”.
* Buscar opción “Edit with IDLE”.
* Aparecerá una ventana con el código y documentación del mismo.
* Presionar F5.
* La interfaz aparecerá inmediatamente.



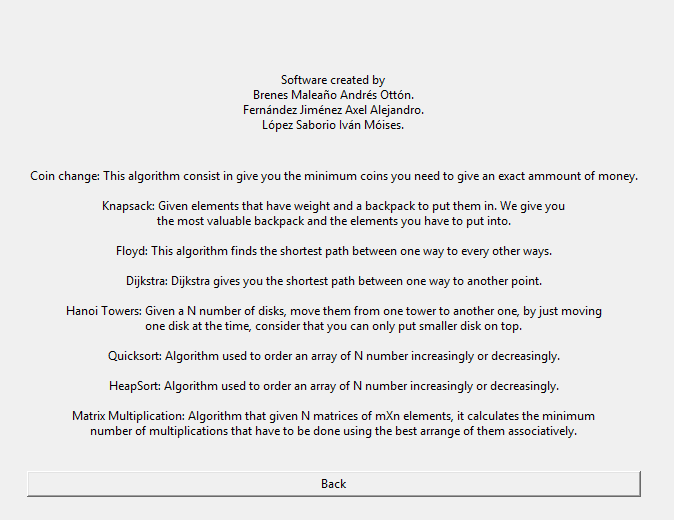
## **1.3. Utilizar los algoritmos**

El botón “Continue to the algorithms” lo lleva a una ventana donde podrá seleccionar el algoritmo que desea utilizar.



## **1.4. Datos a introducir**

Cada algoritmo va a recibir ciertos datos, estos dependerán de lo que el algoritmo requiera. Se recomienda antes de seleccionar los algoritmos ir a la pestaña de información, a la cual se puede acceder en la primera ventana presionando el botón de “Information”.

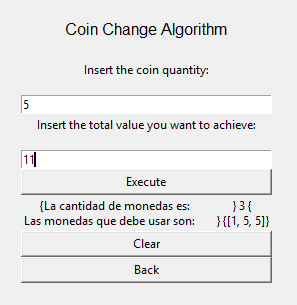


## **1.5. Salir del programa**

Para salir de la aplicación es tan simple como presionar la “X” en la esquina superior derecha, o presionar el botón “Exit” que está presente en la primera ventana de la aplicación.

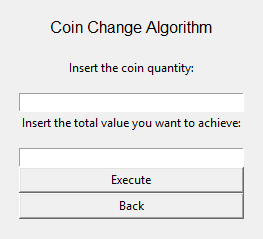
## **1.5. Reutilizar un algoritmo**

Si desea utilizar un algoritmo más de una vez, presione el botón “Clear” presente en la ventana una vez que utiliza el algoritmo.

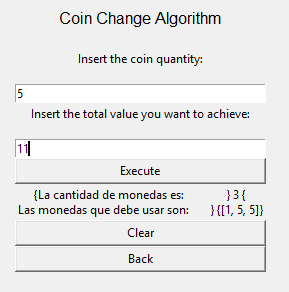


# **2. Descripción de algoritmos**

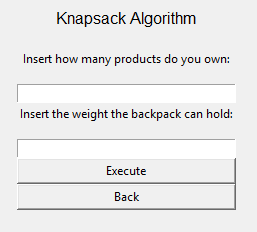
## **2.1 Coin Change**



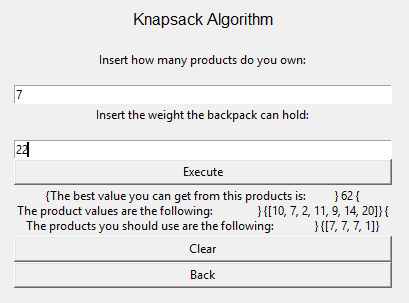
Recibe dos valores, la cantidad de monedas y el vuelto a recibir. Si el usuario pone que tiene 5 monedas, el valor de estas irá de 1 a 5. Y el valor de vuelto será el valor a conseguir con la menor cantidad de monedas posible.



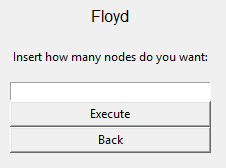
**2.2 Knapsack**



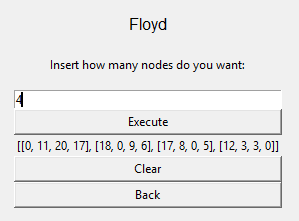
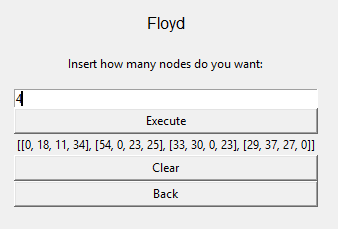
Recibe dos valores, la cantidad de productos que posee el usuario y el máximo peso que el bulto soporta. El valor asignado a cada producto es aleatorio, por lo que si el usuario posee 7 productos y el valor máximo del bulto es de 22, los pesos de cada producto irán de 1 a 7, los valores de cada uno serán aleatorios al presionar “Execute” mostrará que productos debe tener el bulto para tener el mejor valor.



## **2.3 Floyd**

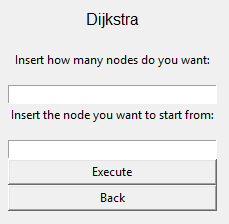


Floyd es más simple, pues solo pide una cantidad de nodos, donde los nodos es la cantidad de vértices en una matriz nxm. Los valores asignados a cada arista son aleatorios, por lo que se aplicará el algoritmo con distancias asignadas aleatoriamente. Por ejemplo, ejecutemos el algoritmo con el mismo número, dos veces para hacer notar que los valores son aleatorios.

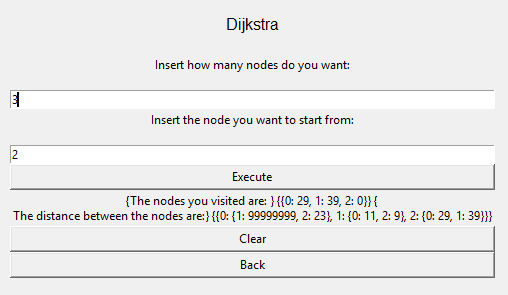


Como podemos notar, utilizando el valor de 4 nodos, nos retorna valores diferentes, queda demostrado que los valores son aleatorios.

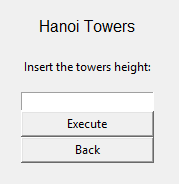
## **2.4 Dijkstra**



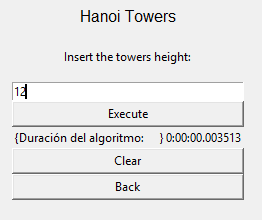
Dijkstra tiene dos entradas, la cantidad de nodos, donde los nodos es la cantidad de vértices en una matriz nxm. Los valores asignados a cada arista son aleatorios, por lo que se aplicará el algoritmo con distancias asignadas aleatoriamente. Y también tenemos el nodo donde queremos iniciar que va a ir desde 0 hasta n-1.



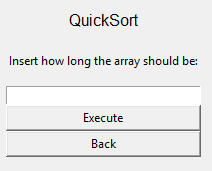
## **2.5 Hanoi Towers**



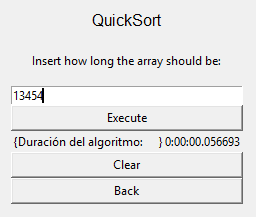
Este algoritmo recibe una única entrada que es el tamaño de las torres, entre más altas sean las torres, más va a durar ejecutando. Al ejecutar nos va a mostrar el tiempo que duró resolviendo el problema.



## **2.6 QuickSort**



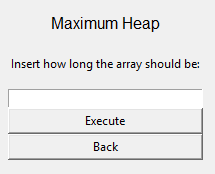
El algoritmo de quickSort recibe un n, donde n es el largo de la lista que tiene que ordenar, lo hace de menor a mayor. Al momento de ejecutar, nos retorna el tiempo que duró ejecutando hasta que ordenó toda la lista,



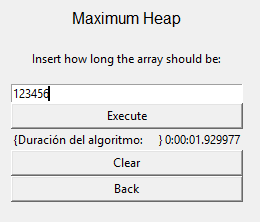
## **2.7 HeapSort**

El algoritmo de HeapSort lo dividimos entre ordenamiento de mayor a menor, y ordenamiento de menor a mayor. El “Maximum HeapSort” ordena de menor a mayor, caso contrario con el “Minimum HeapSort”.

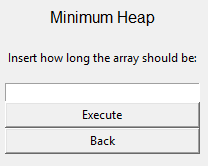
## **2.7.1 Maximum HeapSort**



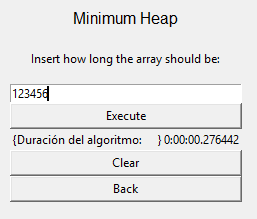
El algoritmo recibe un n, donde n es la cantidad de elementos de la lista que debe ordenar de menor a mayor. Entre más grande el n, más dura el algoritmo.



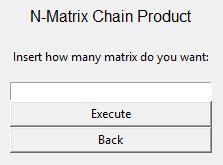
## **2.7.2 Minimum HeapSort**



El algoritmo recibe un n, donde n es la cantidad de elementos de la lista que debe ordenar de menor a mayor. Entre más grande el n, más dura el algoritmo.



## **2.8 N-Matrix Product**



El algoritmo recibe la cantidad de matrices que se desean multiplicar. Las dimensiones de cada matriz son dadas aleatoriamente. Al ejecutar nos va a mostrar la cantidad de operaciones necesarias en su mejor caso, las dimensiones de las matrices y la asociatividad a seguir.

