

# 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 en 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.

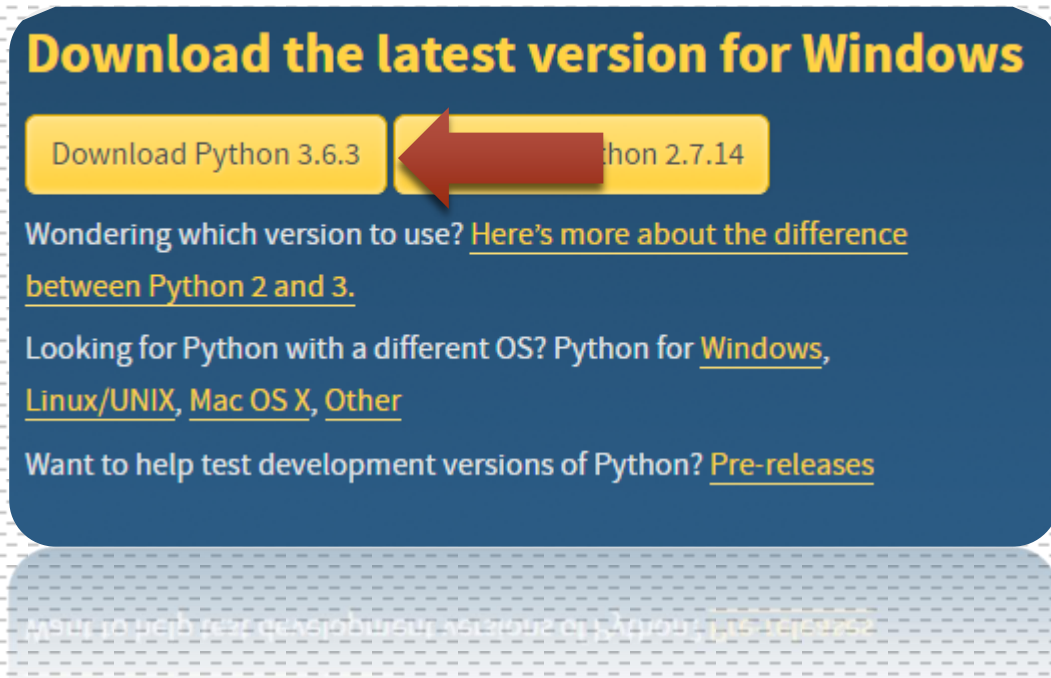
# Contenido

1. Funciones principales .....	4
1.1. Instalación Python 3.4.....	4
1.2. Abrir la interfaz de la aplicación.....	5
1.3. Utilizar los algoritmos .....	6
1.4. Datos a introducir.....	7
1.5. Salir del programa.....	8
1.5. Reutilizar un algoritmo .....	8
2. Descripción de algoritmos.....	9
2.1 Coin Change .....	9
2.3 Floyd .....	11
2.4 Dijkstra .....	12
2.5 Hanoi Towers .....	13
2.6 QuickSort .....	14
2.7 HeapSort.....	15
2.7.1 Maximum HeapSort .....	15
2.7.2 Minimum HeapSort.....	16
2.8 N-Matrix Product .....	17

# 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/>

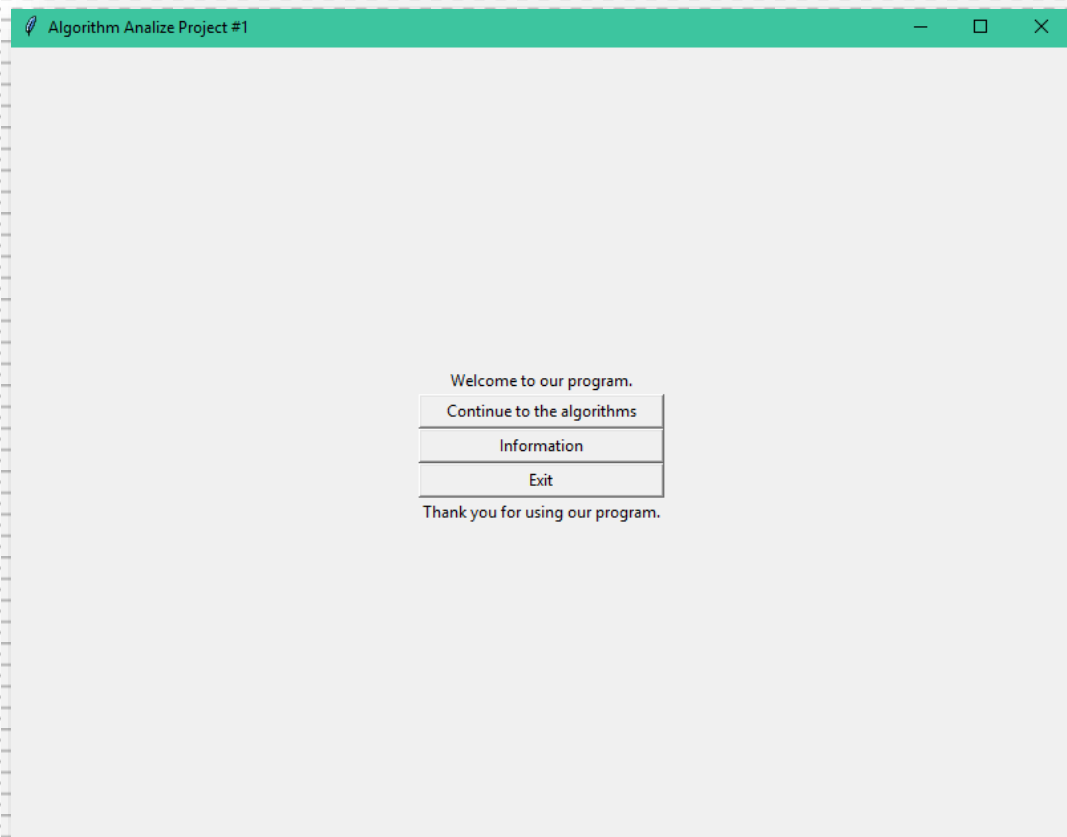


## 1.2. Abrir la interfaz de la aplicación

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

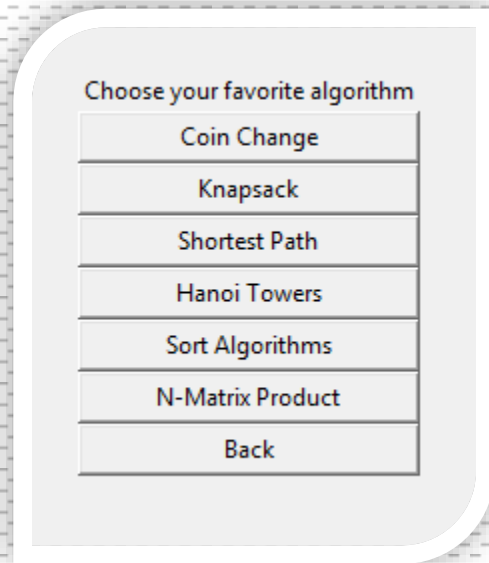
Los pasos para entrar a la interfaz de la aplicación son los siguientes:

- 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.



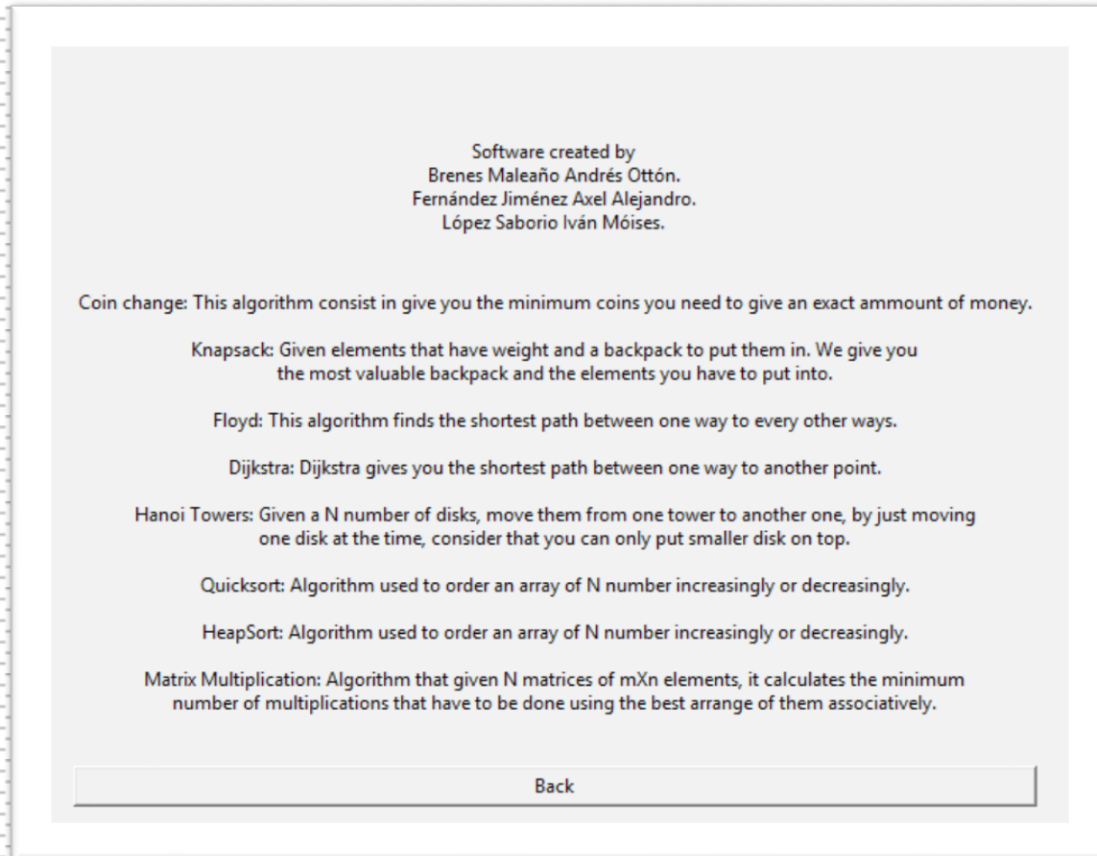
### 1.3. Utilizar los algoritmos

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".

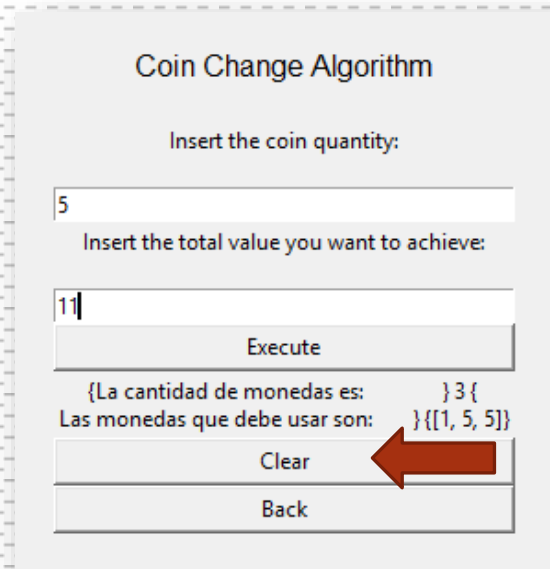


## 1.5. Salir del programa

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.

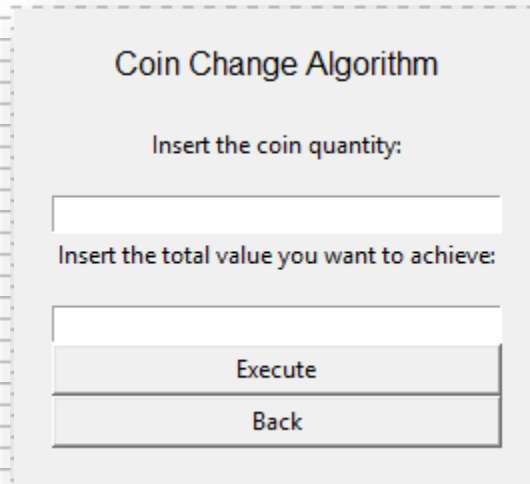


The screenshot shows a window titled "Coin Change Algorithm". It contains two input fields: "Insert the coin quantity:" with the value "5" and "Insert the total value you want to achieve:" with the value "11". Below these fields is an "Execute" button. Under the "Execute" button, the results are displayed: "{La cantidad de monedas es: } 3 {" and "Las monedas que debe usar son: } {[1, 5, 5]}". At the bottom of the window are two buttons: "Clear" and "Back". A red arrow points to the "Clear" button.



## 2. Descripción de algoritmos

### 2.1 Coin Change



Coin Change Algorithm

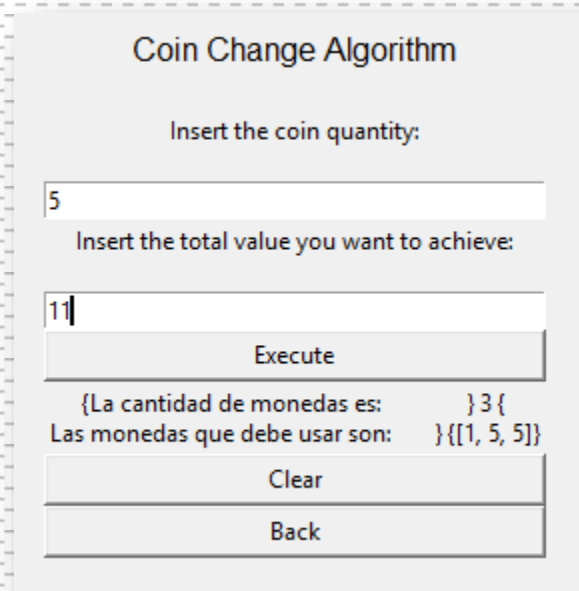
Insert the coin quantity:

Insert the total value you want to achieve:

Execute

Back

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.



Coin Change Algorithm

Insert the coin quantity:

Insert the total value you want to achieve:

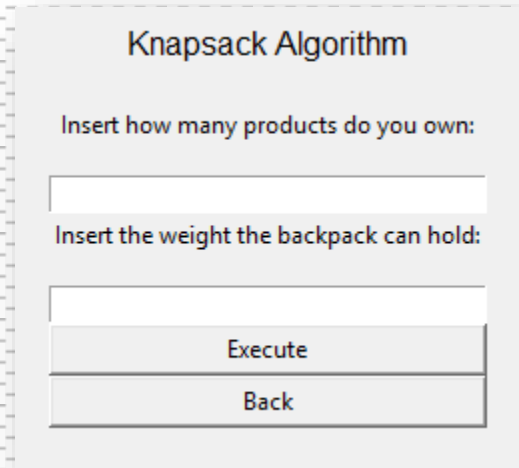
Execute

{La cantidad de monedas es: } 3 {  
Las monedas que debe usar son: } {[1, 5, 5]}

Clear

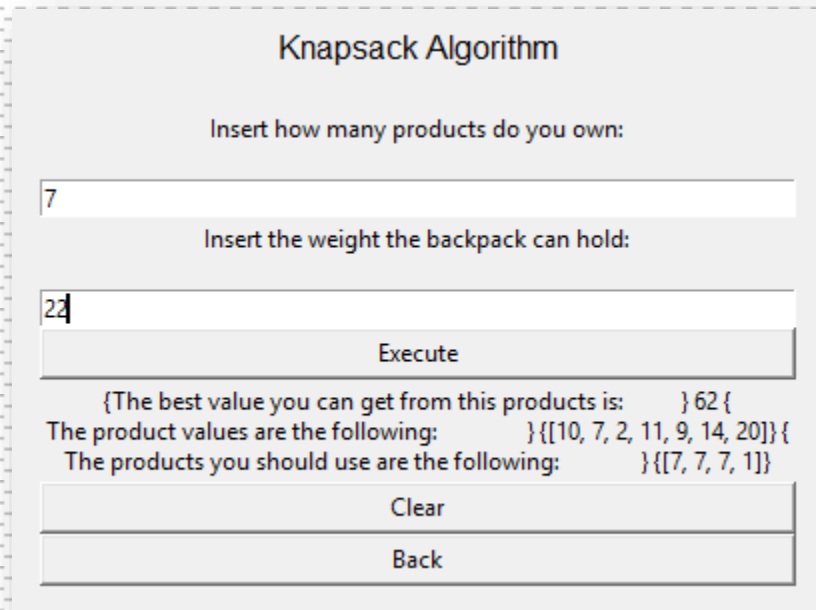
Back

## 2.2 Knapsack



A screenshot of a software window titled "Knapsack Algorithm". It contains two input fields: "Insert how many products do you own:" and "Insert the weight the backpack can hold:". Below these fields are two buttons: "Execute" and "Back".

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.



A screenshot of the same "Knapsack Algorithm" window after execution. The input fields now contain the values "7" and "22". The "Execute" button is still present. Below it, the results are displayed:

{The best value you can get from this products is: } 62 {  
The product values are the following: } {[10, 7, 2, 11, 9, 14, 20]} {  
The products you should use are the following: } {[7, 7, 7, 1]}

Below the results, there are two buttons: "Clear" and "Back".

## 2.3 Floyd

### Floyd

Insert how many nodes do you want:

Execute

Back

Floyd es más simple, pues solo pide una cantidad de nodos, donde los nodos es la cantidad de vértices en una matriz  $n \times m$ . 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.

### Floyd

Insert how many nodes do you want:

Execute

[[0, 18, 11, 34], [54, 0, 23, 25], [33, 30, 0, 23], [29, 37, 27, 0]]

Clear

Back

### Floyd

Insert how many nodes do you want:

Execute

[[0, 11, 20, 17], [18, 0, 9, 6], [17, 8, 0, 5], [12, 3, 3, 0]]

Clear

Back

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

## 2.4 Dijkstra

### Dijkstra

Insert how many nodes do you want:

Insert the node you want to start from:

Execute

Back

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.

### Dijkstra

Insert how many nodes do you want:

Insert the node you want to start from:


Execute

{The nodes you visited are: }{{0: 29, 1: 39, 2: 0}}{  
The distance between the nodes are:}{{0: {1: 99999999, 2: 23}, 1: {0: 11, 2: 9}, 2: {0: 29, 1: 39}}}

Clear

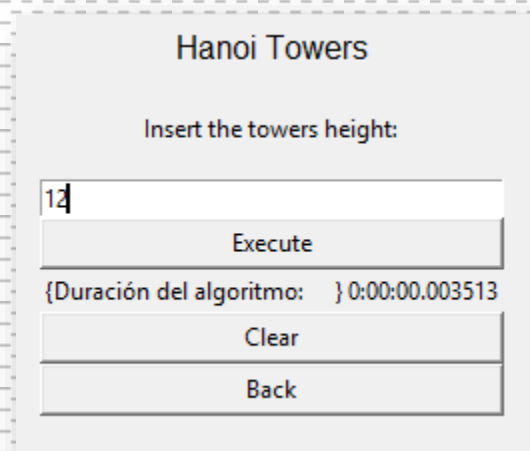
Back

## 2.5 Hanoi Towers



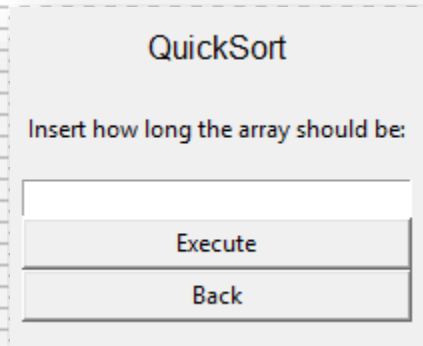
A screenshot of a software window titled "Hanoi Towers". Below the title bar, the text "Insert the towers height:" is displayed. Underneath this text is a text input field. Below the input field are two buttons: "Execute" and "Back".

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.



A screenshot of the same "Hanoi Towers" software window. The text "Insert the towers height:" is still present. The text input field now contains the number "12". Below the input field, the text "{Duración del algoritmo: } 0:00:00.003513" is displayed. Below this text are three buttons: "Execute", "Clear", and "Back".

## 2.6 QuickSort



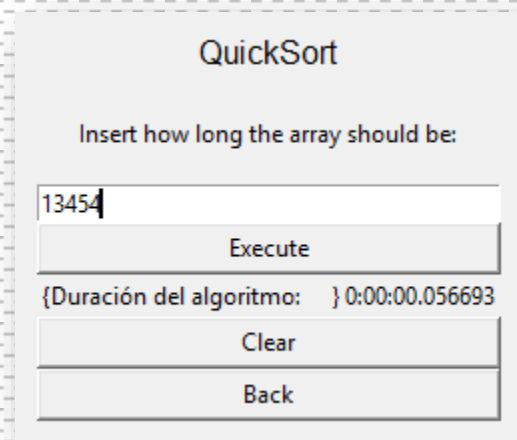
QuickSort

Insert how long the array should be:

Execute

Back

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,



QuickSort

Insert how long the array should be:

Execute

{Duración del algoritmo: } 0:00:00.056693

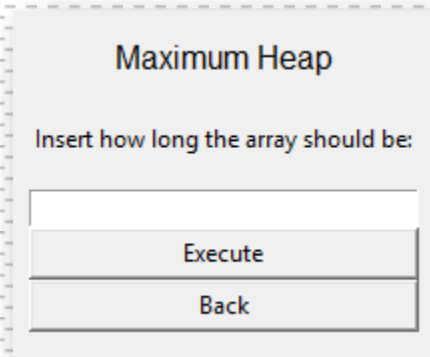
Clear

Back

## 2.7 HeapSort

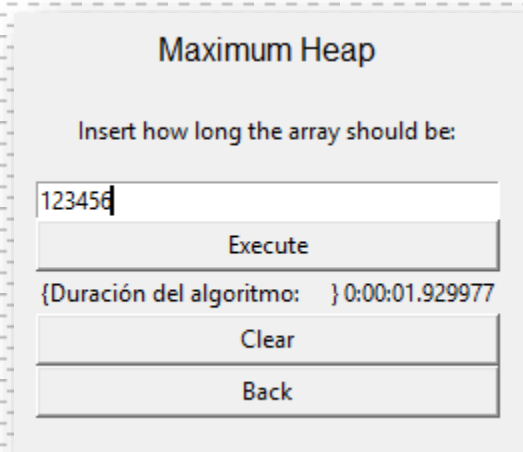
El algoritmo de HeapSort lo dividimos entre ordenamiento de mayor a menor, y ordenamiento de menor a mayor. El "Maximum HeapSort" ordena de menor a mayor, caso contrario con el "Minimum HeapSort".

### 2.7.1 Maximum HeapSort



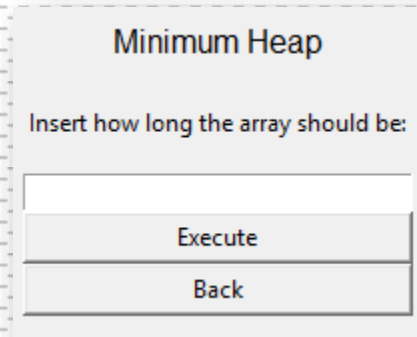
A screenshot of a software interface titled "Maximum Heap". Below the title is a label "Insert how long the array should be:" followed by an empty text input field. Below the input field are two buttons: "Execute" and "Back".

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.



A screenshot of the same "Maximum Heap" software interface. The text input field now contains the number "123456". Below the input field, the "Execute" button is highlighted. Below the "Execute" button, the text "{Duración del algoritmo: } 0:00:01.929977" is displayed. Below this text are two buttons: "Clear" and "Back".

## 2.7.2 Minimum HeapSort



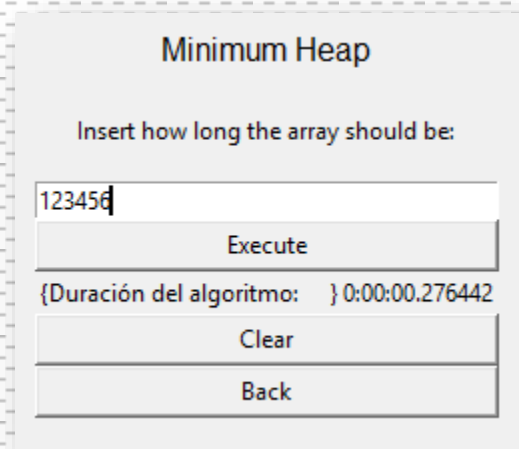
Minimum Heap

Insert how long the array should be:

Execute

Back

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.



Minimum Heap

Insert how long the array should be:

Execute

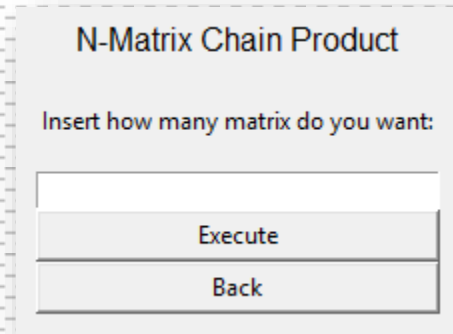
{Duración del algoritmo: } 0:00:00.276442

Clear

Back



## 2.8 N-Matrix Product



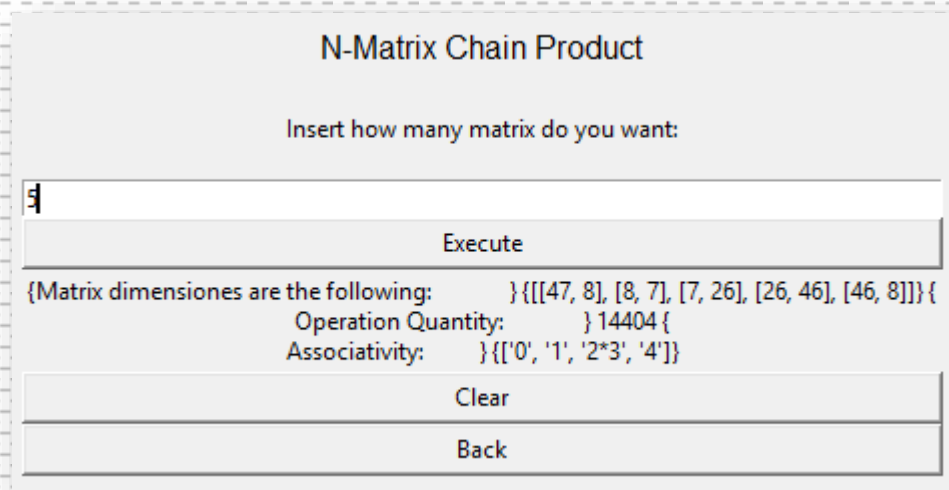
N-Matrix Chain Product

Insert how many matrix do you want:

Execute

Back

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.



N-Matrix Chain Product

Insert how many matrix do you want:

Execute

{Matrix dimensiones are the following: } {[47, 8], [8, 7], [7, 26], [26, 46], [46, 8]] {

Operation Quantity: } 14404 {

Associativity: } {[0, 1, 2\*3, 4]}

Clear

Back