Algoritmos (I)

Concepto de algoritmo

Un algoritmo es un procedmiento paso a paso que define un conjunto de instrucciones a ejecutar en un orden determinado para conseguir una determinada salida. Los algoritmos se crean, normalmente de forma independiente al lenguaje en el que van a ser ejecutados.

Problemas típicos

El propósito de un algoritmo y de un programa en general es resolver un problema. En general muchos de los problemas a resolver son variaciones de problemas típicos qué se suelen presentar. Si conocemos técnicas (algoritmos) que resuelven problemas típicos estas, luego las podemos adaptar para resolver nuestro problema particular.

En este documento presentaremos problemas típicos y plantear formas de solucionarlos. El objetivo principal es ver formas de abordar, desde un punto de vista computacional, diferentes formas de abordar problemas de programación y adquirir técnicas y destrezas que nos ayuden a aprender a programar.

Problemas de búsqueda

Un problema típico es buscar, siguiedo algún tipo de criterio, un valor concreto en un conjunto de datos. Esos datos pueden estar almacenados de alguna forma, o simplemente se van leyendo de uno en uno.

Localizar el valor máximo de un conjunto de datos

Vamos a suponer que tenemos que hacer un programa en el que se lee primero la cantidad de números que queremos leer y luego debemos calcular el valor máximo de los mismos.

El programa para leer los valores podría tener la siguiente forma:

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
for i in range(n_valores):
    valor = int(input(f"Introduce el valor {i + 1}: "))
```

Nota: partimos de la base de que todos los valores que se van a leer son correctos. El número de valores es un entero >= 0 y los valores leídos en el bucle son todos enteros.

Si queremos obtener el valor máximo de los valores leídos, una primera aproximación podría ser inicializar una variable `max` a 0 y luego en cada paso del bucle comprobar si el valor leído es mayor que el almacenado con lo que pasaría a ser el nuevo valor máximo:

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
max = 0
for i in range(n_valores):
    valor = int(input(f"Introduce el valor {i + 1}: "))
    if valor > max:
        max = valor
print(f"El valor mayor leido es {max}")
```

Si ejecutamos este programa con valores de ejemplo podemos obtener algo como:

```
¿Cuántos valores quieres leer? 3
Introduce el valor 1: 45
Introduce el valor 2: 3
Introduce el valor 3: 11
El valor mayor leido es 45
```

Parece que funciona, pero ¿Qué pasa si todos los valores que leemos son negativos?

```
¿Cuántos valores quieres leer? 4
Introduce el valor 1: -45
Introduce el valor 2: -2
Introduce el valor 3: -56
Introduce el valor 4: -14
El valor mayor leido es 0
```

Todos los valores que hemos leído son menores que 0 que es el valor con el que inicializamos la variable. En ningún paso del bucle se encuentra un valor mayor que 0 y por tanto este es el valor que se muestra como máximo, cuando en el ejemplo anterior el valor máximo es -2.

Para solucionarlo, una posible opción es que el primer valor lo leamos antes del bucle y sea tomado como referencia y que luego leamos el resto de valores en el resto del bucle. Cómo uno de los valores lo leemos antes de inicializar el bucle ya el bucle no se repite `n_valores` veces, sino una menos. Una versión inicial del problema podría ser:

```python
n_valores = int(input("¿Cuántos valores quieres leer? "))
max = int(input("Introduce el valor 1: "))
for i in range(n_valores - 1):
    valor = int(input(f"Introduce el valor {i + 1}: "))
    if valor > max:
        max = valor
print(f"El valor mayor leido es {max}")
```

La salida ahora si solo usamos valores negativos podría ser:

```
¿Cuántos valores quieres leer? 4
Introduce el valor 1: -4
Introduce el valor 1: -6
Introduce el valor 2: -1
Introduce el valor 3: -45
El valor mayor leido es -1
```

El valor mayor lo muestra correctamente, pero los índices dentro del bucle no se muestran correctamente (el valor 1 se lee dos veces y luego va de 2 a 3). Esto es porque `range(nvalores - 1)` genera la secuencia `0, 1, 2, ..., n_valores-2)`, pero el primer valor que leemos en el bucle es el 2º. Lo podríamos arreglar usando `i + 2` en el mensaje de lectura:

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
max = int(input("Introduce el valor 1: "))
for i in range(n_valores - 1):
    valor = int(input(f"Introduce el valor {i + 2}: "))
    if valor > max:
        max = valor
print(f"El valor mayor leido es {max}")
```

Ahora si que la salida parece ofrecer el resultado esperado:

```
¿Cuántos valores quieres leer? 3
Introduce el valor 1: -5
Introduce el valor 2: -23
Introduce el valor 3: -2
El valor mayor leido es -2
```

Bueno, aparentemente hemos finalizado parece que nuestro problema da solución correcta sea cual sea los datos que le proporcionesmos. ¿Seguro que es así?.¿Qué pasa si no leemos ningún valor.

```
¿Cuántos valores quieres leer? 0
Introduce el valor 1: 4
El valor mayor leido es 4
```

Si introducimos `0` para indicar que no queremos leer ningún valor nuestro programa, pese a todo nos leerá el valor de inicialización. Este caso lo podemos contemplar, comprobando el valor de `n_valores` justo después de leerlo modificando la salida si el valor es `0`

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
if n_valores == 0:
    print("No se lee ningún valor. No hay valor mayor")
else:
    max = int(input("Introduce el valor 1: "))
    for i in range(n_valores - 1):
        valor = int(input(f"Introduce el valor {i + 2}: "))
        if valor > max:
            max = valor
    print(f"El valor mayor leido es {max}")
```

Ahora si, la salida es correcta en todos los casos:

```
¿Cuántos valores quieres leer? 0
No se lee ningún valor. No hay valor mayor
```

```
¿Cuántos valores quieres leer? 3
Introduce el valor 1: 5
Introduce el valor 2: -3
Introduce el valor 3: 45
El valor mayor leido es 45
```

```
¿Cuántos valores quieres leer? 3
Introduce el valor 1: 5
Introduce el valor 2: -3
Introduce el valor 3: 45
El valor mayor leido es 45
```

En cualquier caso, lo que debería quedar claro, es que no hay soluciones únicas y que la forma de abordar la resución de este tipo de problemas no es de una forma memorística sino entendiendo lo que necesitamos y lo que vamos a obtener con nuestro problema. Si el resultado no es exactamente el esperado vamos refinando el programa hasta obtener el resultado que queremos.

A continuación te muestro otras posibles alternativas válidas. ¿Ves las diferencias?¿Las entiendes? ¿Se te ocurren alternativas?

Alternativa 1: modificando `range` y el mensaje para leer el valor en el bucle:

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
if n_valores == 0:
    print("No se lee ningún valor. No hay valor mayor")
else:
    max = int(input("Introduce el valor 1: "))
    for i in range(2, n_valores +  1):
        valor = int(input(f"Introduce el valor {i}: "))
        if valor > max:
            max = valor
    print(f"El valor mayor leido es {max}")
```

Alternativa 2: usando como referencia el primer valor leído dentro del bucle:

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
if n_valores == 0:
    print("No se lee ningún valor. No hay valor mayor")
else:
    for i in range(0, n_valores):
        valor = int(input(f"Introduce el valor {i}: "))
        if i == 0:
            max = valor
        elif valor > max:
            max = valor
    print(f"El valor mayor leido es {max}")
```

Alternativa 3: comprobando si se van a leer valores, en lugar de si no se va a leer ninguno:

```
n_valores = int(input("¿Cuántos valores quieres leer? "))
if n_valores > 0:
    max = int(input("Introduce el valor 1: "))
    for i in range(2, n_valores +  1):
        valor = int(input(f"Introduce el valor {i}: "))
        if valor > max:
            max = valor
    print(f"El valor mayor leido es {max}")
else:
    print("No se lee ningún valor. No hay valor mayor")
```

Actividad 1

Elige una o varias de las alternativas anteriores y modifícalas para que obtenga el valor más próximo a 1000 de todos los valores que se leen.

Recursos

- Tutorialspoint: Diseño de algorítmos en Python