Universidad Nacional Autónoma de México

Análisis de Algoritmos
Semanal 3

Luis Sebastián Arrieta Mancera (318174116)
Zuriel Enrique Martínez Hernández (318056423)

19 de febrero de 2023

Ejercicio

Se tiene un arreglo A de n enteros cuyos valores están entre entre 0 y $n-1$. Algunos de los valores están repetidos. El arreglo A no está ordenado necesariamente.

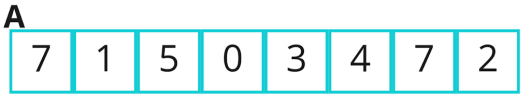


Figura 1: Arreglo de 8 números con valores entre 0 y 7 con un valor repetido.

Preguntas

1. Si hay un único valor r que está repetido en A, y este aparece 2 veces en el arreglo, por consecuencia, hay un valor p que falta en A.

   - Brinda un algoritmo que encuentre a r y p en tiempo $O(n)$ y espacio extra $O(n)$.

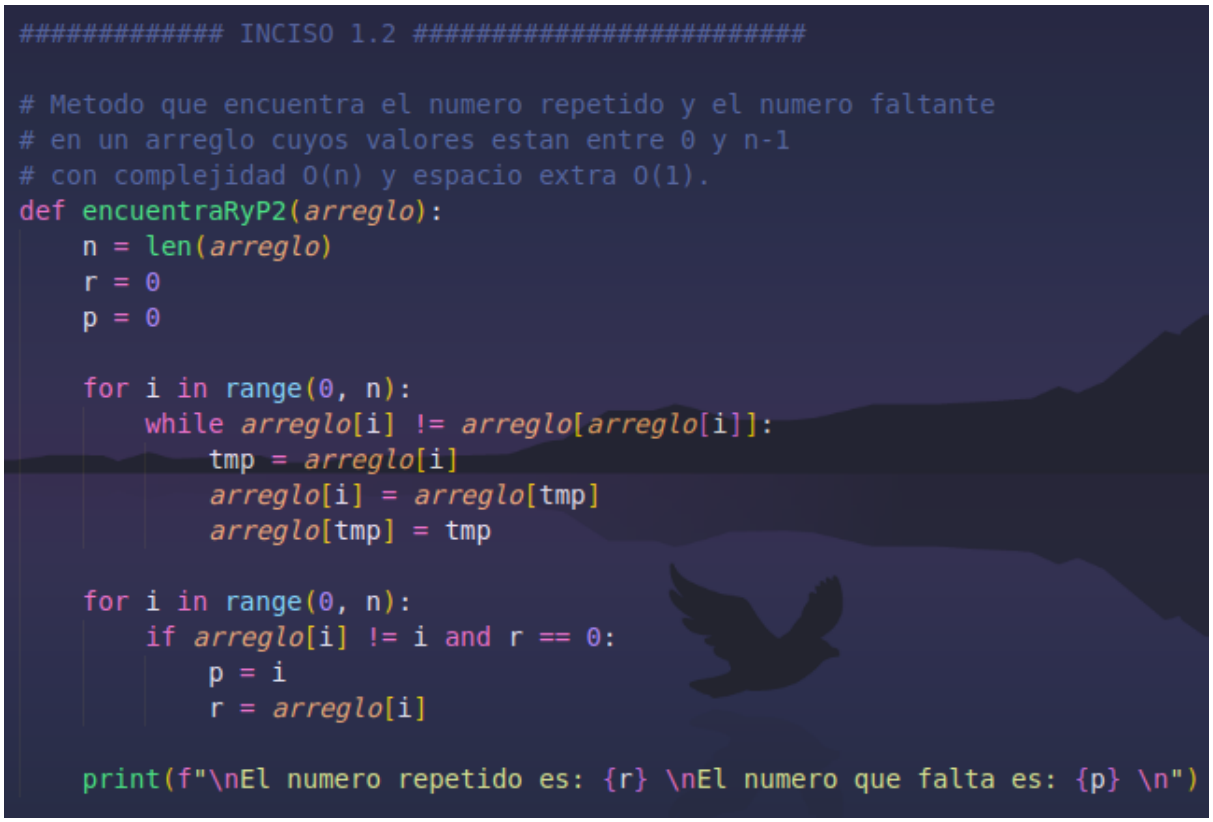
```python
# Metodo que encuentra el numero faltante en un arreglo
# cuyos valores estan entre entre 0 y n - 1.
def numeroFaltante(arreglo):
    n = len(arreglo)
    total = ((n-1) + 1)*(((n-1) + 2)/2)
    suma = sum(arreglo)
    return int(total - suma)

# Metodo que encuentra el numero repetido en un arreglo
# cuyos valores estan entre entre 0 y n - 1.
def numeroRepetido(arreglo):
    visitado = set()
    duplicado = [x for x in arreglo if x in visitado or (visitado.add(x) or False)]
    repetido = duplicado[0]
    return repetido

# Metodo que encuentra el numero repetido y el numnero faltante
# en un arreglo cuyos valores estan entre entre 0 y n - 1.
# con complejidad O(n) y espacio extra O(n)
def encuentraRyP(arreglo):
    r = numeroRepetido(arreglo)
    arreglo.remove(r)
    p = numeroFaltante(arreglo)
    print(f"\nEl numero repetido es: {r} \nEl numero que falta es: {p} \n")
```

Tenemos un método que encuentra el numero faltante de un arreglo cuyos valores están entre 0 y $n - 1$. Para ello, se hace uso de una variable que almacena la longitud del arreglo, para después, haciendo uso de la formula y la variable poder calcular la suma esperada si no faltara dicho elemento. Posteriormente, con otra variable calculamos la suma real, sumando los elementos del arreglo. Finalmente hacemos la resta y ese es nuestro numero faltante.

Después, tenemos un método que encuentra el numero repetido de un arreglo. Para ello, creamos un arreglo auxiliar que almacena los elementos repetidos de un arreglo haciendo uso de set, la cual no permite los elementos repetidos. Como suponemos que solo habrá un elemento repetido, entonces obtenemos el primer elemento del arreglo auxiliar y obtenemos el numero repetido.

Finalmente, en el método final, obtenemos el numero repetido, haciendo uso del método que lo encuentra, eliminamos ese numero repetido del arreglo y obtenemos el numero faltante con el método que lo hace. Como llamamos a los métodos de forma secuencial, la complejidad de este método es O(n), y por hacer uso de un arreglo auxiliar, nuestro espacio extra igual es O(n).

- Brinda un algoritmo que encuentre a r y p en tiempo $O(n)$ y espacio extra $O(1)$.



```
############## INCISO 1.2 ##########################

# Metodo que encuentra el numero repetido y el numero faltante
# en un arreglo cuyos valores estan entre 0 y n-1
# con complejidad O(n) y espacio extra O(1).
def encuentraRyP2(arreglo):
    n = len(arreglo)
    r = 0
    p = 0

    for i in range(0, n):
        while arreglo[i] != arreglo[arreglo[i]]:
            tmp = arreglo[i]
            arreglo[i] = arreglo[tmp]
            arreglo[tmp] = tmp

    for i in range(0, n):
        if arreglo[i] != i and r == 0:
            p = i
            r = arreglo[i]

    print(f"\nEl numero repetido es: {r} \nEl numero que falta es: {p} \n")
```

Tenemos el siguiente algoritmo en el que empezamos por definir a n, r y p, siendo n la longitud del arreglo y r y p inicializadas en 0.

Primeramente, recorremos el arreglo y lo vamos ordenando de manera que cuando volvamos a recorrer el arreglo una segunda vez solo validemos que el arreglo en la posición i sea diferente de la iteración i actual y aparte r sea igual a 0. Si esto se cumple, le asignamos a p el valor de la iteración actual y a r el valor del arreglo en la posición i.

2. Si hay un único valor r que está repetido en A, y este aparece 3 veces en el arreglo, por consecuencia, hay 2 valores p y s que faltan en A.

   - Brinda un algoritmo que encuentre a r, p y s en tiempo $O(n)$ y espacio extra $O(1)$.

```python
############ INCISO 2 ############################

# Metodo que encuentra el numero repetido y los numeros faltantes
# en un arreglo cuyos valores estan entre 0 y n-1
# con complejidad O(n) y espacio extra O(1).
def encuentraRPyS(arreglo):
    n = len(arreglo)
    r = 0
    p = 0
    s = 0

    for i in range(0, n):
        while arreglo[i] != arreglo[arreglo[i]]:
            tmp = arreglo[i]
            arreglo[i] = arreglo[tmp]
            arreglo[tmp] = tmp

    for i in range(0, n):
        if arreglo[i] != i and r == 0:
            p = i
            r = arreglo[i]
            continue
        if arreglo[i] != i:
            s = i
            r = arreglo[i]
            break

    print(f"\nEl numero repetido es: {r} \nLos numeros que faltan son: {p} y {s} \n")
```

Para este algoritmo, repetimos el proceso del inciso anterior, solo que esta vez inicializamos a s = 0 en un inicio y al momento de asignar los valores de p y r, con otra condicional validamos que el arreglo en la posición i sea diferente de i, si esto se cumple, asignamos a s el valor de la iteración actual, y a r le asignamos el valor del arreglo en la posición i.

3. Si hay un único valor r que está repetido en A, y este aparece $k+1$ veces en el arreglo donde $k \geq 1$, por consecuencia, hay k valores p_1, p_2, ..., p_k que faltan en A.

   - Brinda un algoritmo que encuentre a r y los valores p_1, p_2, ..., p_k en tiempo $O(n)$ y espacio extra $O(k)$.

   El algoritmo para este ejercicio es análogo al ejercicio 4 , con diferencia de que como solo nos interesan los números faltantes entonces solo los almacenamos en una lista. **Nota:** Para entender como funciona el algoritmo para este inciso revisar la respuesta del ejercicio 4. El algoritmo es el siguiente:

```python
def ejercicio4(arr: list) -> list:
    faltantes = list()

    # Iterador
    i = 0
    # Cantidad de elementos
    n = len(arr)

    # Variable auxiliar para hacer el swap
    swap = -1
    # print(arr)
    while i < n:

        # Elemento en la posicion actual
        elem = arr[i]

        # Si el elemento es un pivote de repeticion o ya esta ordenado
        if (elem == -2) or (elem == i):
            i+=1
            continue

        # Veamos si se repite
        if elem == arr[elem]:
            # Dejamos un pivote en su posicion correspondiente
            arr[elem] = -2
            # Dejamos un pivote en la posicion actual
            arr[i] = -1
            i+=1
        # En caso contrario son elementos distintos
        else:
            # Verificamos si ya sabiamos que estaba repetido
            if arr[elem] == -2:
                arr[i] = -1
                i+=1
            # En caso contrario hacemos el swap
            else:
                swap = arr[elem]
                arr[elem] = arr[i]
                arr[i] = swap
                # Verificamos si el swap fue con un pivote
                if swap == -1:
                    i+=1

    # Determinamos los numeros repetidos y los faltantes
    for j in range(n):
        valor = arr[j]
        if valor == -1:
            faltantes.append(j)
```

```
49          elif valor == -2:
50              print(j)
```

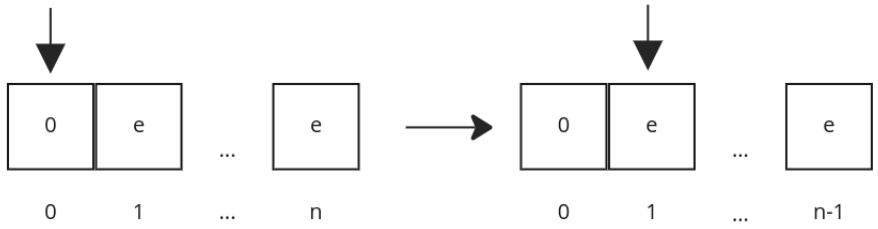
4. Si hay m valores r_1, r_2, ..., r_m que están repetidos en A y k valores p_1, p_2, ..., p_k que faltan en A donde $m \geq 1$ y $k \geq 1$.

    - Brinda un algoritmo que encuentre los valores r_1, r_2, ..., r_m y los valores p_1, p_2, ..., p_k en tiempo $O(n)$ y espacio extra $O(k+m)$.

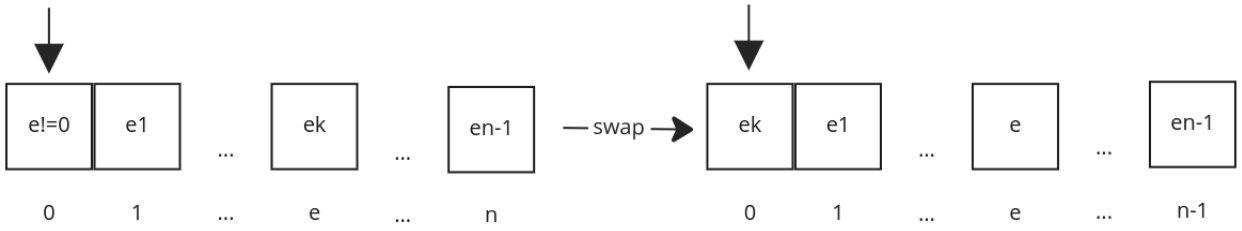
Hint: Para las preguntas 3 y 4 puedes suponer que solo te interesa la respuesta, no el contenido de A, así que puedes modificar su contenido durante tu algoritmo siempre que la repuesta que brindes corresponda con el arreglo original.

La idea principal del siguiente algoritmo es ordenar los elementos y encontrar los elementos faltantes. Sabemos que los elementos van desde 0 hasta n-1, por esto mismo podemos hacer función de su valor para ordenar y colocarlo en la posición que corresponde. Comenzamos revisando el primer elemento del arreglo, veamos los que pueden ocurrir los siguientes casos.

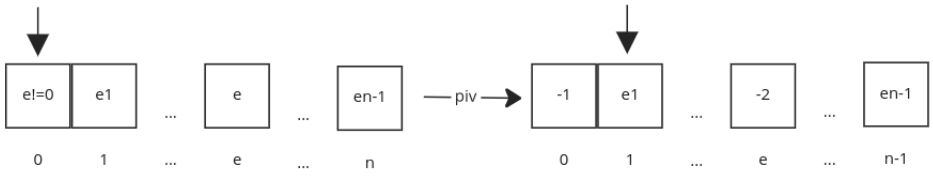
- Si el elemento ya esta en su posición correspondiente entonces pasamos el iterador se pasa al siguiente elemento.



- Si el elemento no corresponde con el índice, entonces hacemos swap con el elemento en su posición.

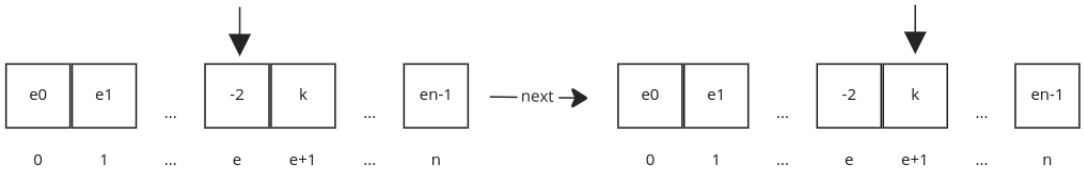


- Si el elemento no corresponde con el indice, y además en su posición está el mismo elemento, entonces en la posición actual dejamos un pivote con el valor -1 y en la posición del elemento dejamos un pivote con el valor -2 e iteramos con el siguiente elemento.

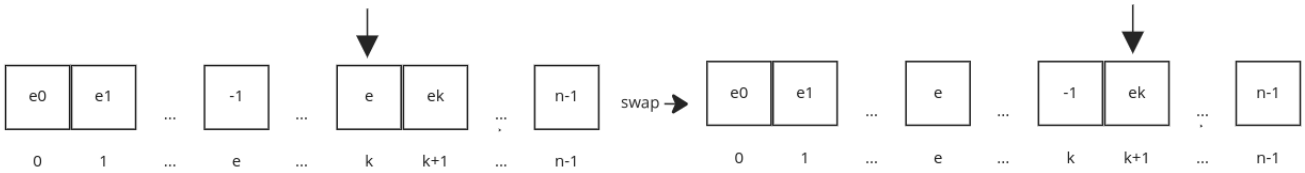


Nota: Para facilitar la abstracción el pivote -1 es un marcador para indicar que no hemos encontrado el elemento que va en esa posición, por ende nos pasamos con el siguiente elemento. El pivote -2 es un marcador para indicar que el elemento que se encuentra en esa posición ya determinamos que está repetido.

- Si el elemento en el que estamos es un pivote -2 entonces nos pasamos con el siguiente elemento.



- El ultimo caso es cuando hacemos un swap con un pivote -1, si esto pasa entonces nos pasamos con el siguiente elemento.



De esta manera al final tendremos un arreglo ordenado y con pivotes. Recorremos el arreglo y cuando encontremos un pivote -1 significa que ese índice corresponde a un número faltante, y cuando encontremos un pivote -2 significa que ese índice corresponde a un número repetido. Con respecto al espacio, podemos agregar a una lista el índice cuando encontremos un pivote -1, esto almacenaría los números faltantes con espacio $O(k)$. De igual manera podemos tener otra lista para guardar los elementos repetidos correspondiente a los índices que contengan el pivote -2 con espacio $O(m)$. De esta manera no se usa espacio extra mas que para almacenar los números, y el tiempo de ejecución del algoritmo (omitiendo constantes) permanece en $O(n)$. El código de este algoritmo es el siguiente:

```python
def ejercicio4(arr: list) -> list:
    repetidos = list()
    faltantes = list()

    # Iterador
    i = 0
    # Cantidad de elementos
    n = len(arr)

    # Variable auxiliar para hacer el swap
    swap = -1
    # print(arr)
    while i < n:

        # Elemento en la posicion actual
        elem = arr[i]

        # Si el elemento es un pivote de repeticion o ya esta ordenado
        if (elem == -2) or (elem == i):
            i+=1
            continue

        # Veamos si se repite
        if elem == arr[elem]:
            # Dejamos un pivote en su posicion correspondiente
            arr[elem] = -2
            # Dejamos un pivote en la posicion actual
            arr[i] = -1
            i+=1
        # En caso contrario son elementos distintos
        else:
            # Verificamos si ya sabiamos que estaba repetido
            if arr[elem] == -2:
                arr[i] = -1
                i+=1
            # En caso contrario hacemos el swap
            else:
                swap = arr[elem]
                arr[elem] = arr[i]
                arr[i] = swap
                # Verificamos si el swap fue con un pivote
                if swap == -1:
                    i+=1

    # Determinamos los numeros repetidos y los faltantes
    for j in range(n):
        valor = arr[j]
        if valor == -1:
            faltantes.append(j)
        elif valor == -2:
            repetidos.append(j)
```