# Instituto Politécnico Nacional

# Escuela Superior de Cómputo

Unidad académica: Análisis y Diseño de Algoritmos

Actividad:

“Índice Mágico”

Equipo:

Solares Velasco Arturo Misael

Solis Lugo Mayra

Grupo: 3CV1

Profesor: García Floriano Andrés

Fecha:

21 de Mayo de 2024

## Reporte de Práctica: Índice Mágico

### Descripción del Problema

Un "índice mágico" en un array AAA de longitud nnn se define como un índice tal que A[i]=iA[i] = iA[i]=i. Dado un array ordenado de enteros distintos, el objetivo es escribir un método para encontrar un índice mágico, si existe, en el array AAA. Además, se explora cómo encontrar un índice mágico cuando los valores no son distintos.

### Implementación de la Solución

El código proporcionado incluye tres métodos para encontrar el índice mágico, cada uno adaptado a diferentes condiciones del array. A continuación se presentan las implementaciones y una función `main` para ejecutar y demostrar el uso de estas funciones.

#### Función 1: `find_magic_index_simple`

Esta función busca un índice mágico en un array de enteros distintos de manera lineal.

```python
def find_magic_index_simple(arr):
    for i in range(len(arr)):
        if arr[i] == i:
            return i
    return -1  # No se encontró un índice mágico
```

**Explicación:** La función recorre el array completo comparando cada índice con su valor correspondiente. Si encuentra un índice mágico, lo retorna; si no, retorna -1.

#### Función 2: `find_magic_index_distinct`

Esta función usa búsqueda binaria para encontrar un índice mágico en un array de enteros distintos.

```python
def find_magic_index_distinct(arr):
    def binary_search(arr, start, end):
        if start > end:
            return -1
        mid = (start + end) // 2
        if arr[mid] == mid:
            return mid
        elif arr[mid] > mid:
            return binary_search(arr, start, mid - 1)
        else:
            return binary_search(arr, mid + 1, end)

    return binary_search(arr, 0, len(arr) - 1)
```

**Explicación:** Utiliza un enfoque de búsqueda binaria. Divide el array en mitades y compara el elemento medio con su índice. Dependiendo del resultado, se ajusta el rango de búsqueda hasta encontrar un índice mágico o concluir que no existe.

#### Función 3: `find_magic_index_non_distinct`

Esta función también usa búsqueda binaria, pero está adaptada para arrays donde los valores pueden no ser distintos.

**Explicación:** Adapta la búsqueda binaria para manejar arrays con valores repetidos. Compara el índice medio y si no es un índice mágico, busca recursivamente en ambos lados del array ajustando los límites con los valores mínimos y máximos posibles.

#### Función `main`

La función principal para demostrar el uso de las tres funciones.

```python
def main(arr):
    print(f'''
    Ejemplo de uso de las funcione de Magic Index:

        el arreglo es: {arr}

        Magic Index simple: {find_magic_index_simple(arr)}
        Magic Index distinct: {find_magic_index_distinct(arr)}
        Magic Index non distinct: {find_magic_index_non_distinct(arr)}
    ''')
```

**Explicación:** Crea un array de ejemplo y llama a cada una de las funciones de índice mágico. Imprime los resultados para mostrar cómo funcionan en la práctica.

## Resultados

Al ejecutar el código con el array de ejemplo `[-10, -5, 2, 2, 2, 3, 4, 7, 9, 12, 13]`, se obtienen los siguientes resultados:

- **Magic Index simple:** 2
- **Magic Index distinct:** 2
- **Magic Index non distinct:** 2

```
Ejemplo de uso de las funciones de Magic Index:

    El arreglo es: [-10, -5, 2, 2, 2, 3, 4, 7, 9, 12, 13]

    Magic Index simple: 2
    Magic Index distinct: 2
    Magic Index non distinct: 2
```

Esto demuestra que todas las funciones pueden encontrar un índice mágico cuando existe, y la implementación está adaptada correctamente para manejar arrays con valores distintos y no distintos.

## Conclusiones

La búsqueda lineal es simple pero ineficiente para arrays grandes, mientras que la búsqueda binaria optimiza la eficiencia para arrays de valores distintos, alcanzando una complejidad de O(log⁡n)O(\log n)O(logn). La versión adaptada de la búsqueda binaria maneja arrays con valores repetidos eficientemente. Las tres funciones demostraron ser efectivas para encontrar un índice mágico cuando existe, destacando la importancia de adaptar los algoritmos a las condiciones específicas del array para optimizar el rendimiento. Ejemplos prácticos confirman la precisión y eficiencia de los métodos propuestos.