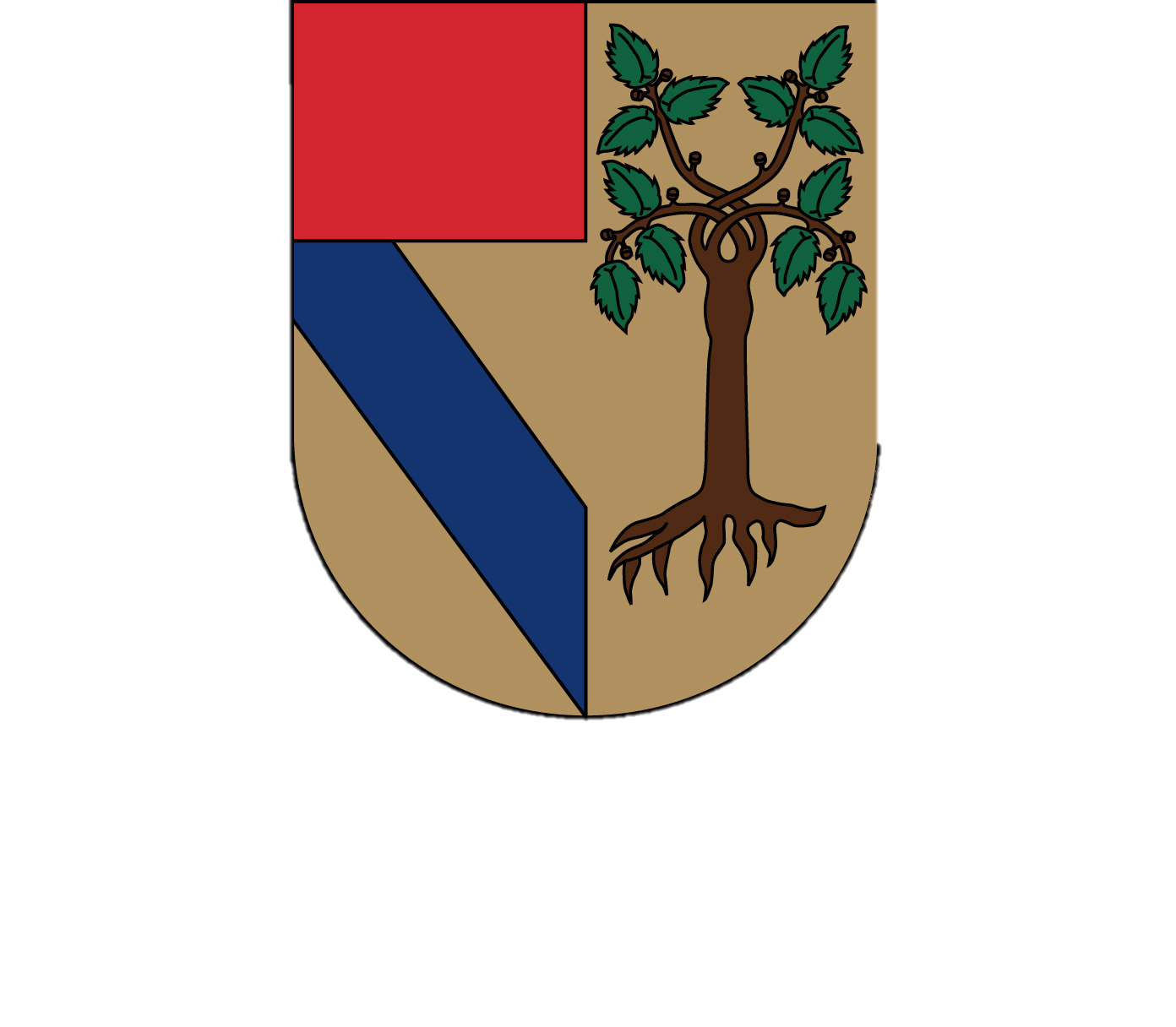
floyd’s algorithm: tortoise and hare

Reporte de Proyecto

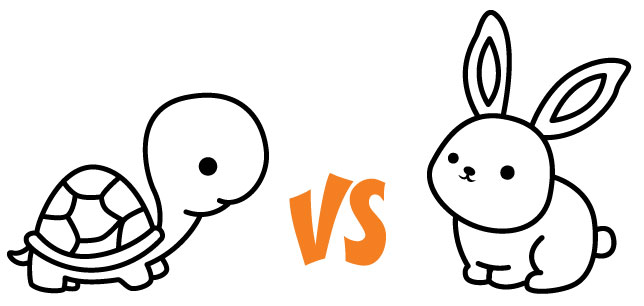
Universidad Panamericana Campus Bonaterra

Estructuras de datos y algoritmos



**Luis Eduardo Robles Jiménez.**

**0224969**



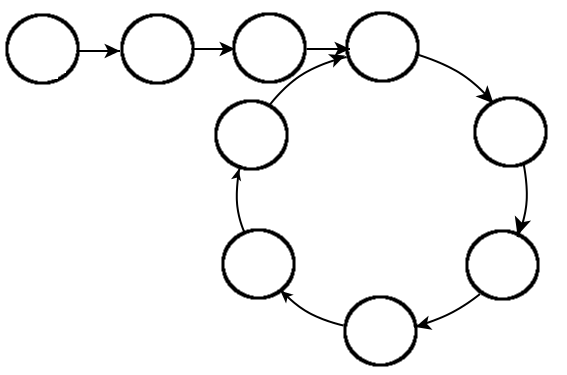
Floyd's cycle-finding algorithm

El algoritmo de Robert Floyd, es una serie de pasos que tienen como objetivo principal buscar y encontrar ciclos en una lista de datos. Para este proyecto, se utilizará una adaptación del algoritmo que trabaja con arreglos y no memoria dinámica, esto con el fin de resolver un problema en específico.

¿Qué es la detección de ciclos?

En ciencias de la computación, es el problema algorítmico de encontrar algún ciclo en una secuencia de valores iterados en función de algo. Es decir:

Y gráficamente un ciclo dentro de una estructura de datos podría verse algo así:



Esta área de estudio tiene aplicaciones dentro de la electrónica en la solución de circuitos eléctricos, exploración en entornos de robótica móvil y en la computación como control de información en redes informáticas, entre otras.

Planteamiento del problema: Encontrar el número duplicado.

Dado un arreglo de números conteniendo n + 1 enteros, donde cada dígito está entre **1 y n**, inclusivo. Está demostrado por el principio del palomar que al menos un número repetido existe. Se debe asumir que sólo un elemento aparece más de una vez, considerar que puede ser más de una. Encuentra el numero duplicado.

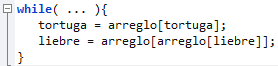
Restricciones.

* Sólo se puede usar O(1) espacio extra.
* El arreglo no puede ser modificado (solo lectura).
* La complejidad en tiempo debe ser menor a O(n2).

Problema rescatado de: <https://www.youtube.com/watch?v=pKO9UjSeLew>

El algoritmo.

La manera de trabajar del algoritmo hace referencia al muy bien conocido cuento de la tortuga y la liebre, esto es, se definen un par de punteros que recorren nuestro arreglo con velocidades distintas, ambos punteros se mueven refiriéndose al índice que se indica en el número que esta seleccionado en el arreglo. El puntero numero 1 (la tortuga) se mueve de esa manera, paso por paso y, por otro lado, el segundo puntero (la liebre) avanza de la misma forma, pero dando dos brincos en cada iteración. El movimiento se define en código de la siguiente manera:



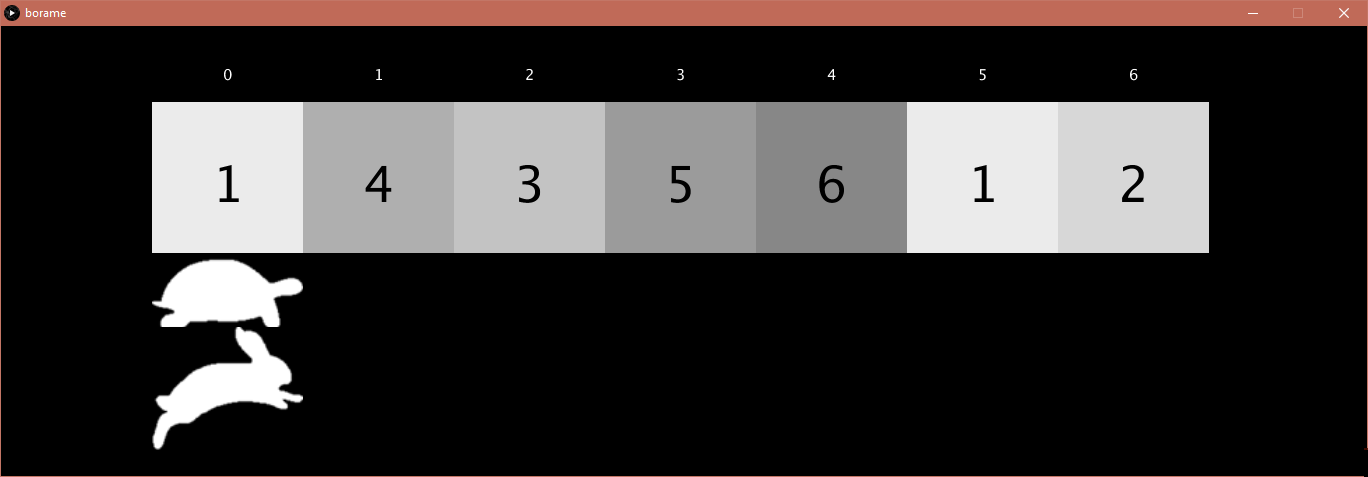
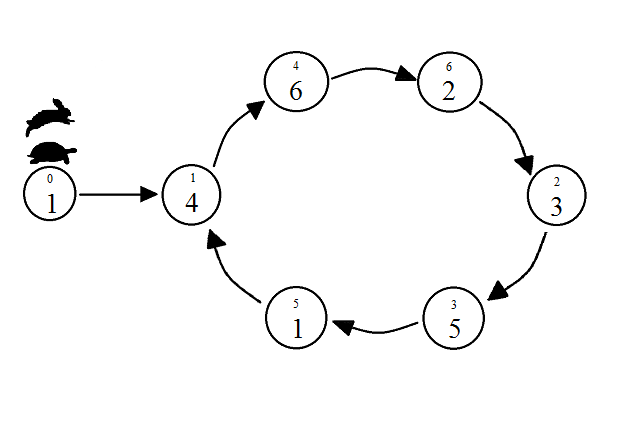
Las iteraciones se detienen una vez que ambos punteros se encuentran. Cuando eso ocurre, debe procederse descartando a la liebre y agregando otra tortuga en la posición inicial del arreglo. A partir de esas posiciones, ambos punteros lentos deben seguir avanzando de manera natural, y de nuevo, cuando coinciden se detiene todo avance y el valor que debe ser devuelto es el índice del lugar donde están ambas tortugas.

Solución paso a paso.

En el caso de que se nos dé un número **n = 6** y el siguiente arreglo: **{1, 4, 3, 5, 6, 1, 2}**; la ejecución del algoritmo procedería de la manera mostrada a continuación.

**NOTA:** Para cada paso se mostrarán dos formas distintas de visualización del arreglo de números y los punteros. La figura de la izquierda representa el arreglo como es trabajado por la máquina, y la otra figura muestra un acomodo más claro de entender para el ojo humano.

1. Se comienza ubicando los 2 punteros en el índice 0.

1. Después ambos punteros deben avanzar a su ritmo (la tortuga se dirige a la posición referida con su valor actual y, la liebre hace eso 2 veces) hasta que se encuentren. Tomando las siguientes posiciones.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TORTUGA | | LIEBRE | | VISUALIZACION | |
| Índice | **Valor** | **Índice** | **Valor** | **Máquina** | **Apoyo interpretativo** |
| 1 | 4 | 4 | 6 |  |  |
| 4 | 6 | 2 | 3 |  |  |
| 6 | 2 | 5 | 1 |  |  |
| 2 | 3 | 4 | 6 |  |  |
| 3 | 5 | 2 | 3 |  |  |
| 5 | **1** | **5** | **1** |  |  |

1. Llegados a este punto, es cuando se reemplaza a la liebre con la nueva tortuga en la posición inicial. Y deben avanzar ambos punteros lentos hasta encontrarse, nótese que el punto donde se encuentran es siempre el punto de articulación del ciclo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TORTUGA original | | TORTUGA nueva | | VISUALIZACION | |
| Índice | **Valor** | **Índice** | **Valor** | **Máquina** | **Apoyo interpretativo** |
| 5 | 1 | 0 | 1 |  |  |
| 1 | 4 | 1 | 4 |  |  |

1. De esta manera concluye el algoritmo, y se sabe que el numero repetido es 1 porque es el índice del punto de articulación de nuestro diagrama.

Conclusión.

Existen muchas maneras distintas de resolver este problema, pero se considera que el algoritmo de Floyd es el óptimo para la tarea, ya que sólo éste cumple con las restricciones dadas en el problema, se anexa una tabla de comparativa con las otras respuestas no viables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Características* | *Ordenar* | *Hash Table* | *COMPARAR* | *Floyd’s* |
| TIEMPO | O(nlogn) | O(1) | O(n2) | O(n) |
| ESPACIO | O(1) | O(n) | O(1) | O(1) |
| PROBLEMA | \*El arreglo no puede ser modificado (solo lectura). | \*Sólo se puede usar O(1) espacio extra. | \*Complejidad debe ser menor a O(n2). | --- |

Como último agregado, se dice que la complejidad del algoritmo es O(n), debido al hecho de que, si la estructura está compuesta por n elementos y m nodos representando la longitud del ciclo. Se sabe que ambos punteros estarán en la misma posición dentro del ciclo cuando la distancia entre la tortuga y la liebre es divisible entre m. Por lo tanto, se deduce que: