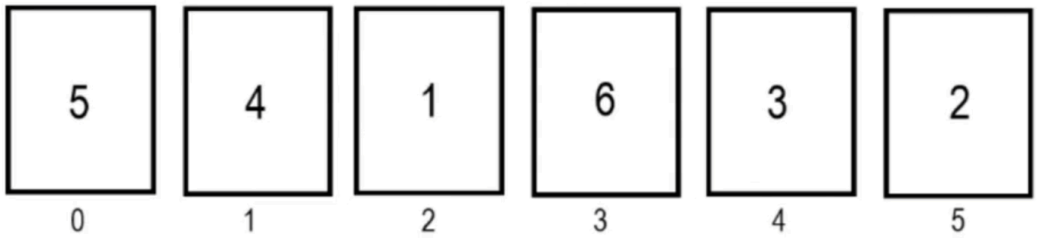
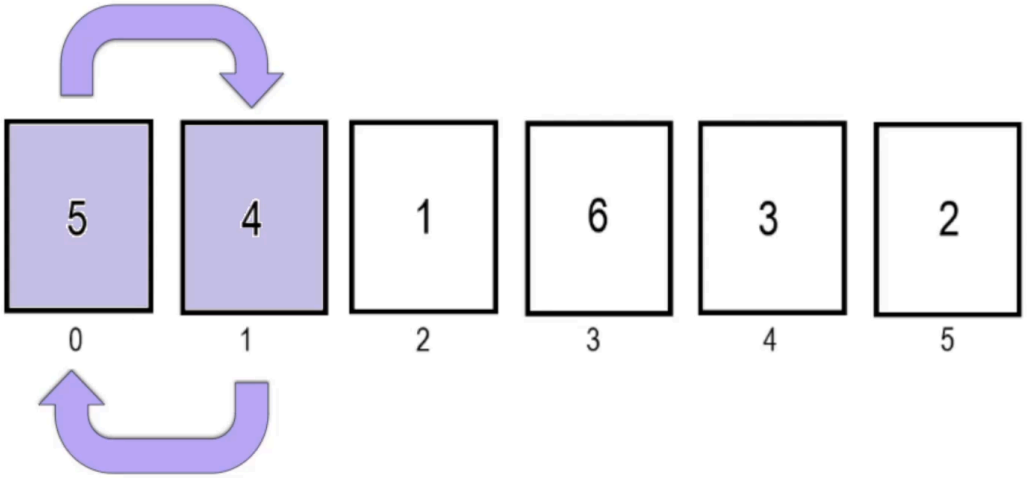
1. Métodos de ordenación
    1.1. Bubble sort - Parte I

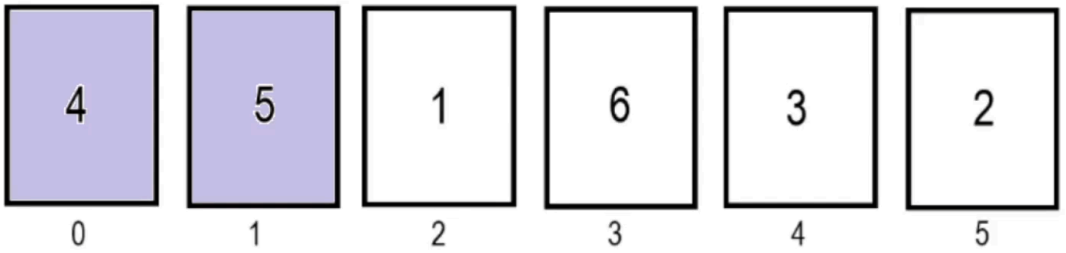
Ordenamiento burbuja. Un algoritmo es una serie de pasos predefinidos que siempre lleva al mismo resultado.
El algoritmo Bubble Sort es útil para ordenar elementos dentro de un arreglo de menor a mayor o de mayor a menor.



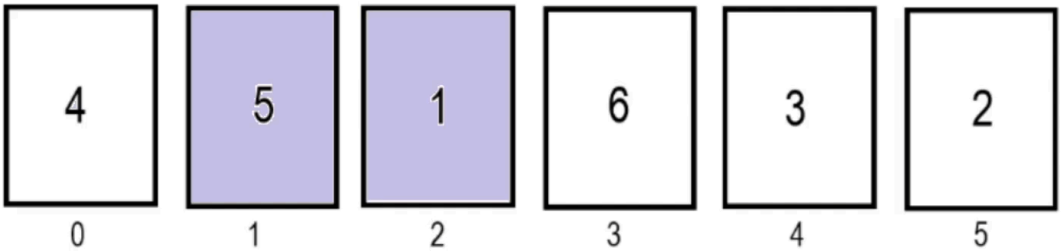
El algoritmo hace una comparación desde la posición 0, es decir, desde el inicio, y compara este elemento con el elemento que esté en su posición a la derecha.



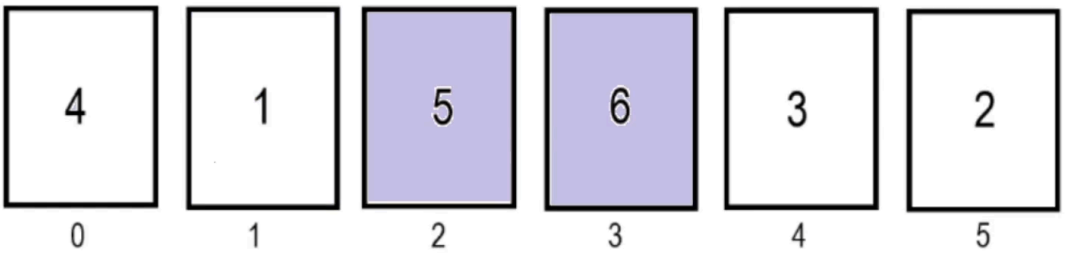
Luego de la comparación, el algoritmo se pregunta si están bien ubicados o si es necesario intercambiar los datos.



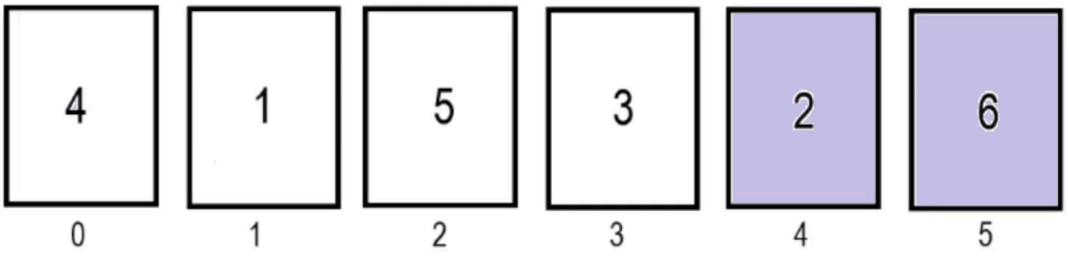
Seguidamente vuelve a continuar con la comparación, pero comenzando desde la posición 1 con el elemento que está en la posición de su derecha.



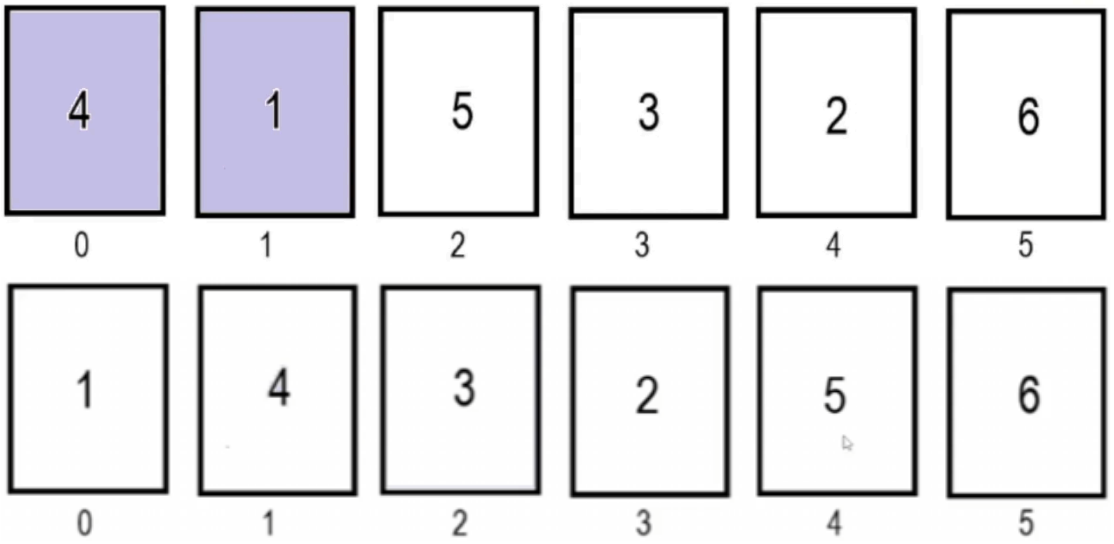
Se pregunta si es necesario nuevamente intercambiar los datos hasta completar el arreglo.



Si una comparación resulta en que no tiene que hacer un intercambio, solo continua a la siguiente posición.



Al finalizar un primer recorrido, el elemento más grande siempre quedará al final del arreglo, y el algoritmo vuelve a analizarlo para asegurarse que se complete la instrucción.



En cada iteración, el algoritmo siempre itera sobre todos los elementos, en cada una siempre posiciona un elemento en el lugar correspondiente, por lo que siempre se va a repetir tantas veces como elementos se encuentren en el arreglo.

(Ver Archivo)

1.2. Bubble sort - Parte II

(Ver Archivo)
Ordena strings según código ASCII.
En el caso de objetos, se debe determinar qué campo o propiedad se va a considerar para el ordenamiento, teniendo en cuenta que el recorrido de un arreglo de objetos puede generar un desbordamiento fuera de sus límites.

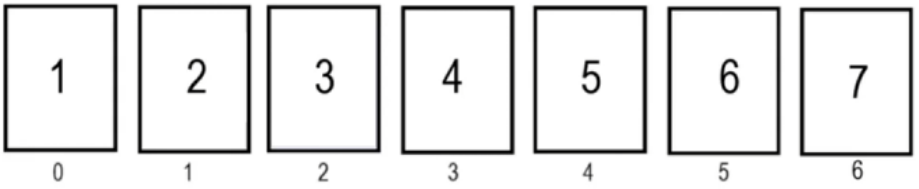
2. Algoritmos de búsqueda

2.1. Linear search

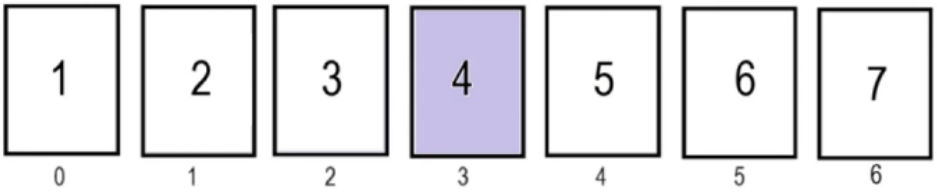
La búsqueda lineal es útil para encontrar valores específicos dentro de un arreglo, ya que accede al arreglo y lo analiza posición por posición para encontrar el dato buscado. En caso de encontrarlo, indica la posición en la que fue hallado, y en caso de no hacerlo devuelve un **-1**.
(Ver Archivo)

2.2. Binary search

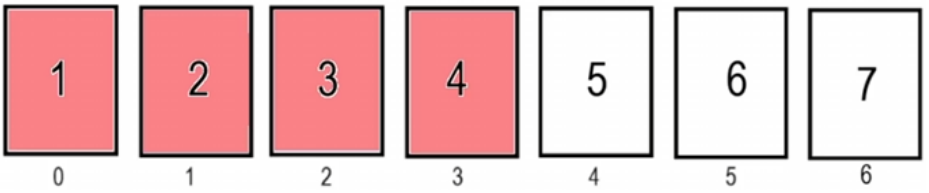
El algoritmo de búsqueda binaria es útil cuando es necesario encontrar un valor específico dentro de un arreglo en donde se retorna como respuesta la posición o el índice en el que se ubica y un **-1** si no lo encuentra, similar a la búsqueda lineal. La desventaja de la búsqueda binaria es que los elementos deben estar previamente ordenados de menor a mayor antes de aplicar la instrucción de búsqueda pero la ventaja radica en que ejecutará menos iteraciones que la búsqueda lineal para lograr el resultado.



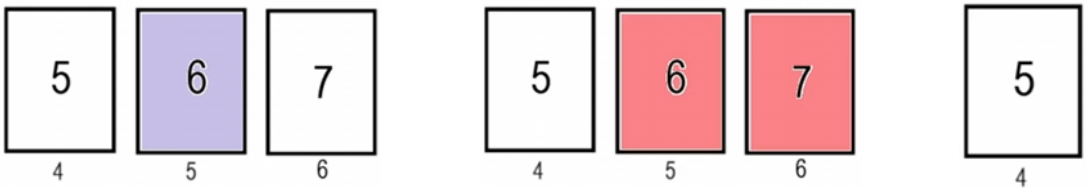
Una vez establecido el arreglo ordenados de menor a mayor y el alcance del mismo, es decir, la longitud, lo que hace el algoritmo es situarse en el medio.



Una vez posicionado en el medio, se pregunta si ese es el elemento que se está buscando mediante condicionales, si la respuesta es negativa, comienza a recorrerlo.



Toma el valor central como referencia para sacar conclusiones, si el valor buscado es un número menor al valor central, entonces no considera los elementos menores y empieza a iterar a partir del valor central, de allí la importancia de ordenarlos. Generando así, menos iteraciones, modificando el índice desde donde se comienza la búsqueda, comenzando desde los elementos restantes.



Luego, el algoritmo repite el proceso: se sitúa en el medio, consulta si el elemento buscado es menor o mayor, elimina los elementos restantes y se vuelve a situar para corroborar el elemento buscado. (Ver Archivo)

3. Integración de contenidos
    3.1. Desafio práctico

- BubbleSort:

Crea una función que reciba por parámetro un array de números y los ordene de menor a mayor mediante el ordenamiento burbuja.

- Indicar ordenamiento:

Crea una función que reciba por parámetro un array de números e indique si el array se encuentra ordenado de forma ascendente o no.

- Contar la cantidad de veces que aparece un número en un array:

Crea una función que reciba por parámetro un número y un array de números, y devuelva la cantidad de veces que aparece ese número en el array utilizando búsqueda lineal.

- Obtener votos:

Dado un array con los datos de una encuesta realizada sobre los temas musicales más populares —lo que se tiene en el array son objetos definidos con el nombre de un tema musical y cantidad de votos que obtuvo— crear el array con los datos de 5 temas y una cantidad de votos para cada uno. Los votos obtenidos deben estar desordenados. Una vez creado el array, se deberá escribir un algoritmo que, dado un tema, me informe cuántos votos obtuvo, considerando:

    - Resolverlo aplicando búsqueda lineal.
    - Resolverlo aplicando búsqueda binaria.

    3.2. Resolución desafío práctico

(Ver Archivo)

4. Implementación Avanzada de Conocimientos II
    4.1. Avanzando el proyecto integrador - Parte I

(Ver Archivo)

    4.2. Avanzando el proyecto integrador - Parte II

(Ver Archivo)

5. Checkpoint de conocimientos

- ¿Cuál es la característica principal de la implementación de Bubblesort en arrays de strings?

Compara los valores ASCII de los primeros caracteres.

- ¿Cuál es la condición necesaria para aplicar la búsqueda binaria en un array?

El array debe estar ordenado.

- ¿Cuál es la principal diferencia entre la búsqueda binaria y la búsqueda lineal?

La búsqueda binaria divide el conjunto de datos, mientras que la búsqueda lineal lo recorre secuencialmente.

- ¿Cuál es la condición faltante del for interno para que el algoritmo bubbleSort se complete?

```
function ordenarObjetos(array){
    for(let j = 0 ; j < array.length ; j++){
        for(let i = 0 ;              ; i++){
            if(array[i].numero > array[i+1].numero){
                let temp = array[i];
                array[i] = array[i+1];
                array[i+1] = temp;
            }
        }
    }
}
```

i < array[i].length.

- ¿Está completo el siguiente algoritmo de búsqueda binaria?

```
function busquedaBinaria(arr, objetivo) {
    let izquierda = 0;
    let derecha = arr.length - 1;

    while (izquierda <= derecha) {
        let medio = Math.round((izquierda + derecha) / 2);

        if (arr[medio] < objetivo) {
            izquierda = medio + 1;
        } else {
            derecha = medio - 1;
        }
    }
    return -1;
}
```

No, falta un condicional y un retorno.

- En la búsqueda binaria, ¿cuál es la importancia de establecer correctamente los límites (izquierda y derecha)?

```
function busquedaBinaria(arr, objetivo) {
    let izquierda = 0;
    let derecha = arr.length - 1;
```

Define el rango donde se realiza la búsqueda.

- ¿Cuántas iteraciones realiza la búsqueda lineal en el peor caso si el elemento no está presente en el array?

Igual al número de elementos en el array.

- ¿Está bien aplicado el algoritmo de ordenamiento bubbleSort en la siguiente función?

```
function ordenarObjetos(array){
    for(let j = 0 ; j < array.length ; j++){
        for(let i = 0 ; i < array[i].length ; i++){
            if(array[i].numero > array[i+1].numero){
                let temp = array[i].numero;
                array[i].numero = array[i+1].numero;
                array[i+1].numero = temp;
            }
        }
    }
}
```

No, no es correcto ya que al momento de realizar el intercambio no cambia de posición a los objetos, sino que intercambia los valores de sus propiedades.