ALGORITMOS DE ORDENAMIENTO

DEV.F:
DESARROLLAMOS(PERSONAS);

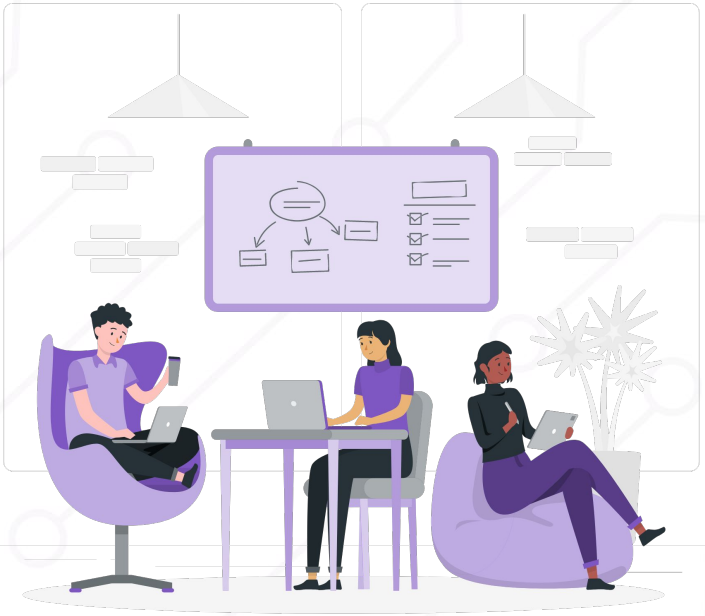
OBJETIVOS DE LA SESIÓN

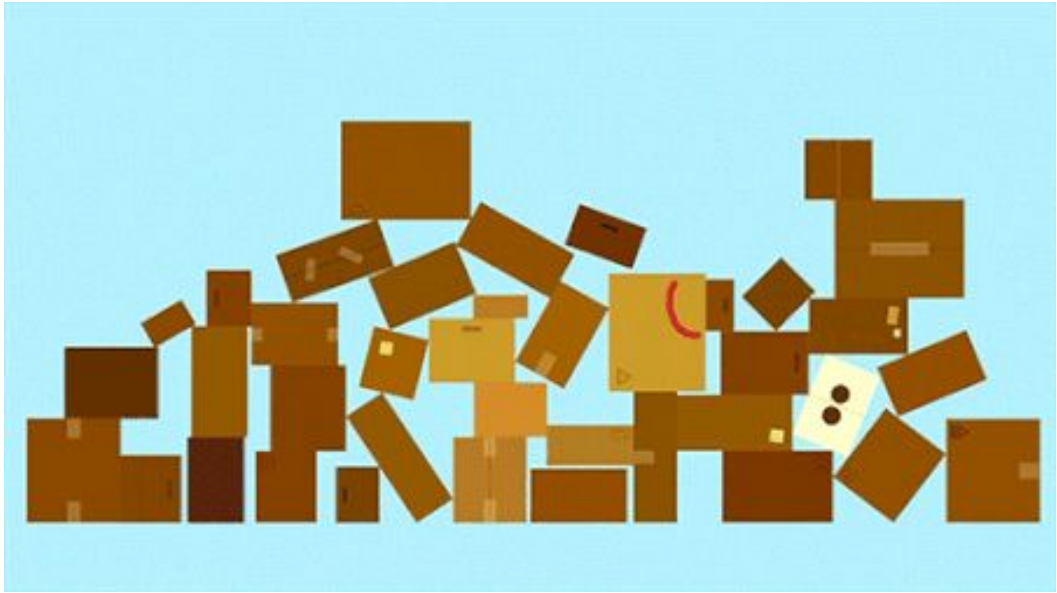
1. EL ALCANCE DE LA SESIÓN DE HOY SERÁ EL TEMA ALGORITMOS DE ORDENAMIENTO EN JAVASCRIPT

2. ENTENDER QUE SON LOS ALGORITMOS DE ORDENAMIENTO Y CÓMO FUNCIONAN.

3. VEREMOS SU FUNCIONAMIENTO EN CÓDIGO PARA ENTENDER MEJOR EL TEMA.

4. SEGUIREMOS CON LA DINÁMICA DE LOS EJERCICIOS PARA SEGUIR PRACTICANDO Y REFORZANDO EL TEMA VISTO DURANTE LA SESIÓN.





Algoritmos de ordenamiento

Un algoritmo de ordenamiento, es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, el resultado de salida es un reordenamiento de la entrada de datos.

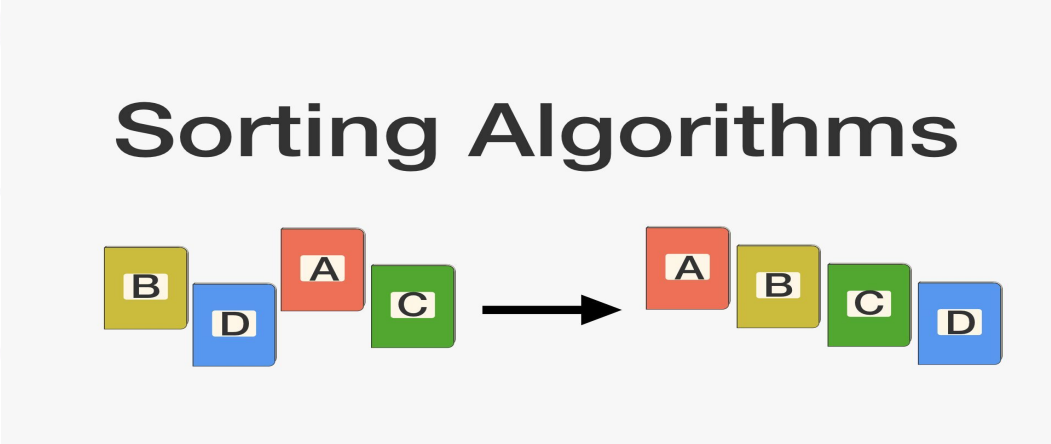
8 5 3 1 4 7 9

DEV.F

¿Qué podemos ordenar?

Cualquier estructura de datos con elementos que sean ordenables

- Podemos ordenar números, porque unos son mayores que otros.
- Podemos ordenar meses, porque unos vienen antes que otros
- Podemos ordenar palabras, por el orden en el alfabeto.



Algoritmos de ordenación

Algunos ejemplos:

- **Ordenamiento por inserción**
- **Ordenamiento de burbuja**
- **Ordenamiento por selección**

Receta algoritmos de ordenación

En la cocina del código primero:

1. Recorremos el array buscando el elemento mínimo.
2. Intercambiamos ese elemento con el que esté en la primera posición.
3. Buscamos el siguiente mínimo que no hayamos ordenado.
4. Lo cambiamos por el segundo.
5. Repetir hasta terminar.

Código para ordenar

¿Qué necesitamos conocer?

Cuál es la posición en la que va a ir el elemento ordenado: 0,1,2,3 ... ,N

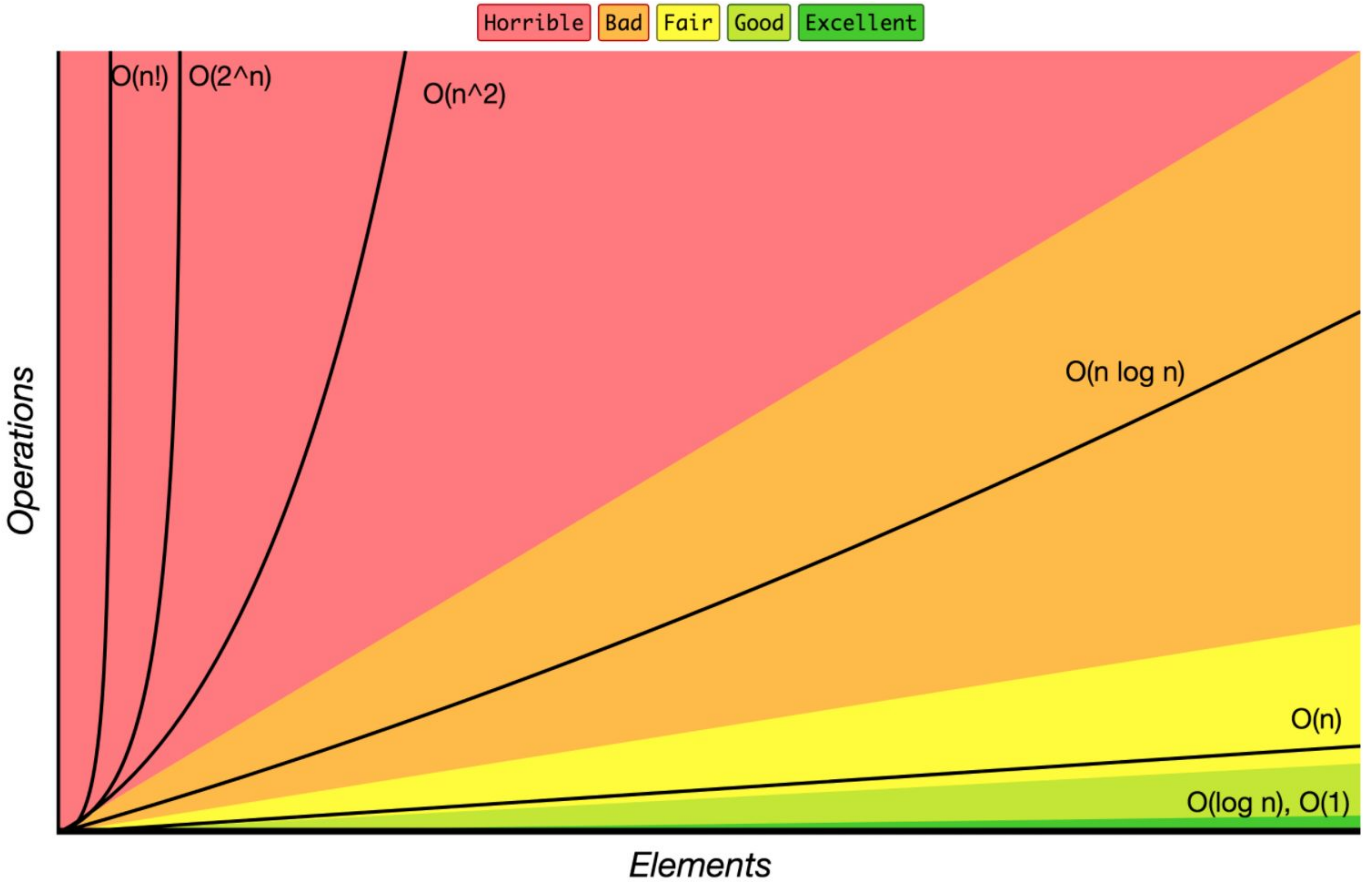
Cuál es el elemento a partir del cual vamos a buscar el mínimo.

El mínimo en cada repetición.

Algoritmos de Ordenamiento

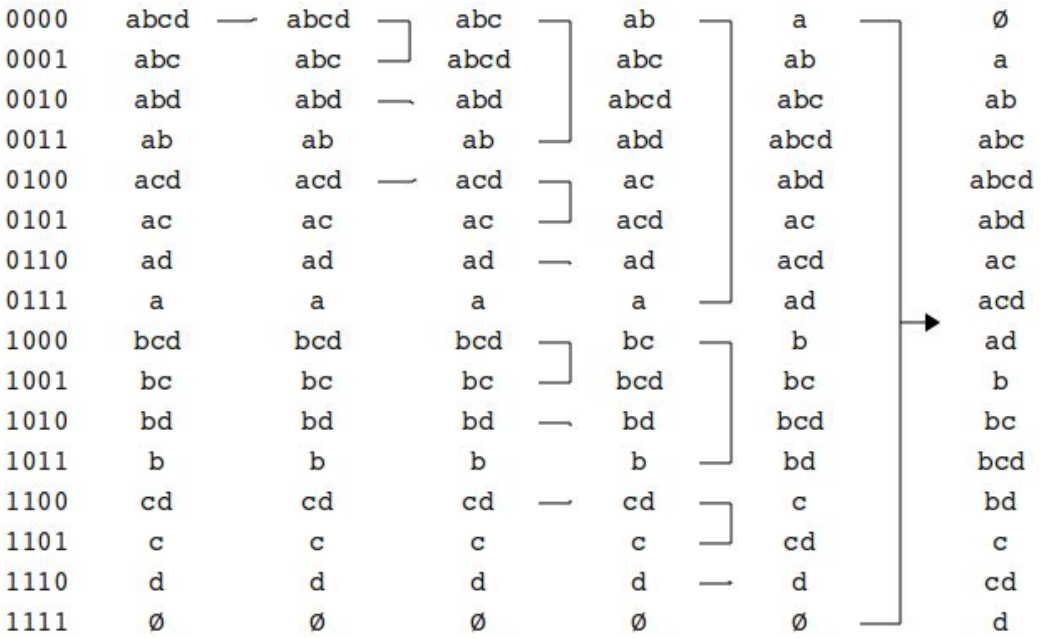
Algunos algoritmos de ordenamiento agrupados según estabilidad tomando en cuenta la complejidad computacional.

Estables				
Nombre traducido	Nombre original	Complejidad	Memoria	Método
Ordenamiento de burbuja	Bubblesort	$O(n^2)$	$O(1)$	Intercambio
Ordenamiento de burbuja bidireccional	Cocktail sort	$O(n^2)$	$O(1)$	Intercambio
Ordenamiento por inserción	Insertion sort	$O(n^2)$("(en el peor de los casos)")	$O(1)$	Inserción
Ordenamiento por casilleros	Bucket sort	$O(n)$	$O(n)$	No comparativo
Ordenamiento por cuentas	Counting sort	$O(n+k)$	$O(n+k)$	No comparativo
Ordenamiento por mezcla	Merge sort	$O(n \log n)$	$O(n)$	Mezcla
Ordenamiento con árbol binario	Binary tree sort	$O(n \log n)$	$O(n)$	Inserción



(peor caso, caso promedio y mejor caso)

Los órdenes usados con más frecuencia son los **órdenes numéricos** y **órdenes lexicográficos**.



Ordenamiento Burbuja (Bubble Sort)

Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiandolos de posición si están en el orden equivocado.

Características:

- Simple y fácil de entender.
- Ineficiente para listas grandes.
- Estable: mantiene el orden relativo de registros iguales.

Complejidad: $O(n^2)$

DEV.F::

Ordenamiento por inserción (Insertion Sort)

Concepto Básico

Imagina que estás jugando cartas. Cada vez que recibes una nueva carta, la insertas en el lugar correcto para mantener tu mano de cartas ordenada. Insertion Sort funciona de manera similar: toma un elemento de la lista y lo coloca en su posición correcta con respecto a los elementos ya ordenados que están a su izquierda.

Características:

- Eficiente para conjuntos de datos pequeños o casi ordenados.
- Simple de implementar.
- Estable: mantiene el orden relativo de registros iguales.
- Mayormente ineficiente para listas grandes.

Complejidad: $O(n^2)$

DEV.F

Ordenamiento por Mezcla (Merge Sort)

Concepto Básico

Es un algoritmo de ordenamiento eficiente basado en el enfoque de dividir para conquistar. Divide el conjunto de datos en mitades hasta que cada fragmento tenga un solo elemento o ninguno, y luego estos fragmentos se van combinando (o mezclando) en orden.

Características:

- Muy eficiente y con un tiempo de ejecución bastante predecible.
- Requiere memoria adicional para la fusión de las sublistas.
- Estable: mantiene el orden relativo de registros iguales.

Complejidad: O(n log n)

DEV.F

Uso de sort() en JavaScript

Concepto Básico

El método sort() de JavaScript ordena los elementos de un array in situ y devuelve el array.
Por defecto, convierte los elementos a cadenas y los ordena según la posición del valor Unicode de cada carácter. Para ordenar números o según otros criterios personalizados, es necesario proporcionar una función de comparación.

Características:

- Ordena el array in situ, es decir, modifica el array original.
- Flexible: Permite ordenar por criterios personalizados mediante una función de comparación.
- No es estable en todos los entornos (aunque en los motores modernos se ha hecho un esfuerzo por hacerlo estable).

Complejidad: O(n log n)

HORA DEL CÓDIGO