DEV.F::

# Clase 7

**Divide y venceras**

## Comenzamos en 10 min

6:40pm (hora CDMX)


Presentación creada por contribución de Diego Lechuga

# Temas de la clase (180 min)

## Divide y venceras!

Dividir
Conquistar
Combinar

## Algoritmos básicos de D&Q

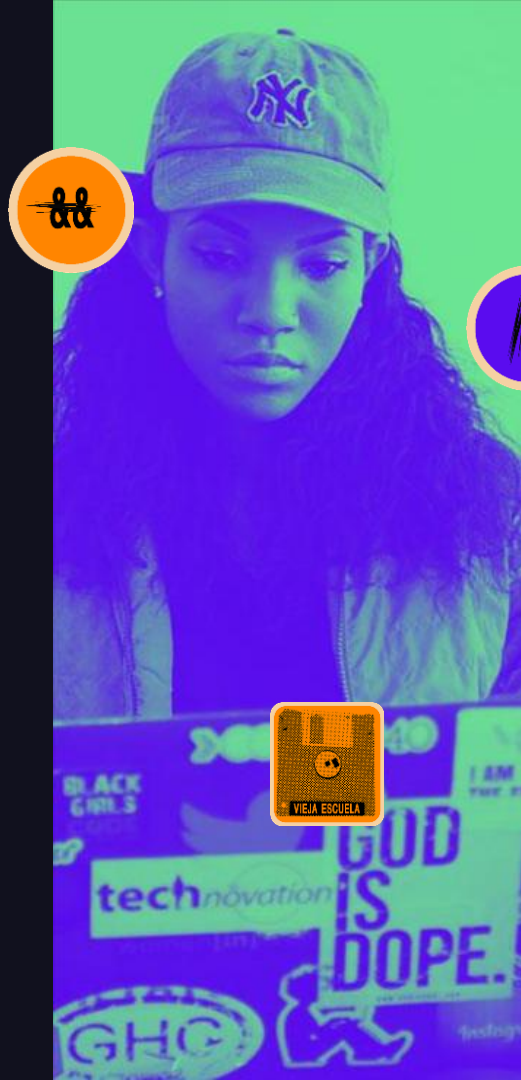
Número máximo y mínimo

Busqueda binaria

## Recomendaciones

Usos practicos de D&Q

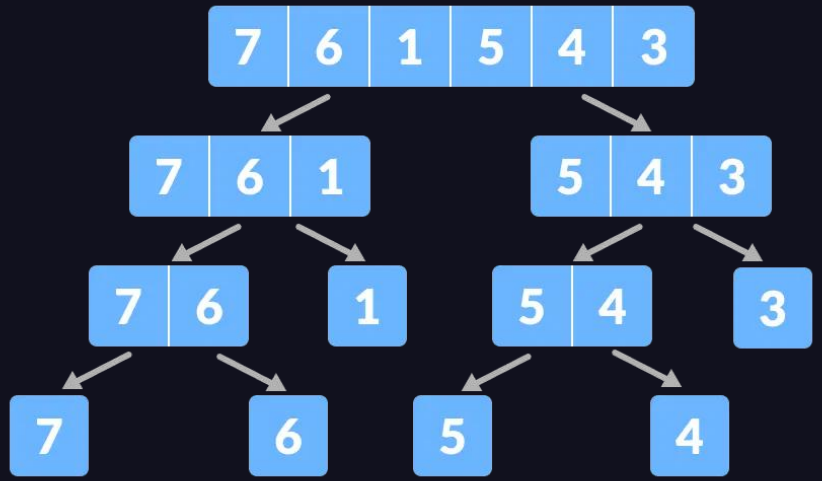
## Reto de la semana




DEV.F::

# Dividir

- **Descomposición en subproblemas:** Identifica **subproblemas** más simples que, combinados, representen el problema general.
- **División progresiva:** Continúa dividiendo los subproblemas en fracciones aún más pequeñas hasta que no sea posible **simplificar** más.
- **Uso de la recursividad**: La descomposición debe realizarse mediante **recursividad**, dividiendo el problema una y otra vez. Este proceso continúa hasta alcanzar subproblemas tan simples que puedan resolverse directamente, sin necesidad de más divisiones.



# Conquistar

**Una vez que el problema se ha dividido en subproblemas, el siguiente paso es aplicar una estrategia para encontrar la solución:**

- Si los subproblemas son lo suficientemente simples, puedes resolverlos directamente.
- Si no, utiliza recursividad para seguir dividiendo y resolviendo cada subproblema hasta alcanzar una solución directa.
- También puedes aplicar algoritmos intermedios según la naturaleza del problema para optimizar el proceso.

**El enfoque elegido dependerá del tipo de problema y de las herramientas o técnicas disponibles para resolverlo de manera eficiente.**

DEV.F::

# Combinar

- Una vez que hemos resuelto los subproblemas, el siguiente paso es combinar sus soluciones para obtener la solución completa del problema original.
- Esta fase de combinación es clave, ya que el algoritmo debe unir las respuestas de los subproblemas más simples de manera coherente para construir la solución del problema complejo. La manera de combinar las soluciones dependerá del tipo de problema y de la estructura de la solución buscada.
- Por ejemplo, en algoritmos como merge sort, la combinación consiste en fusionar listas ordenadas, mientras que en otros casos puede requerir sumar, concatenar o aplicar reglas específicas de agregación.

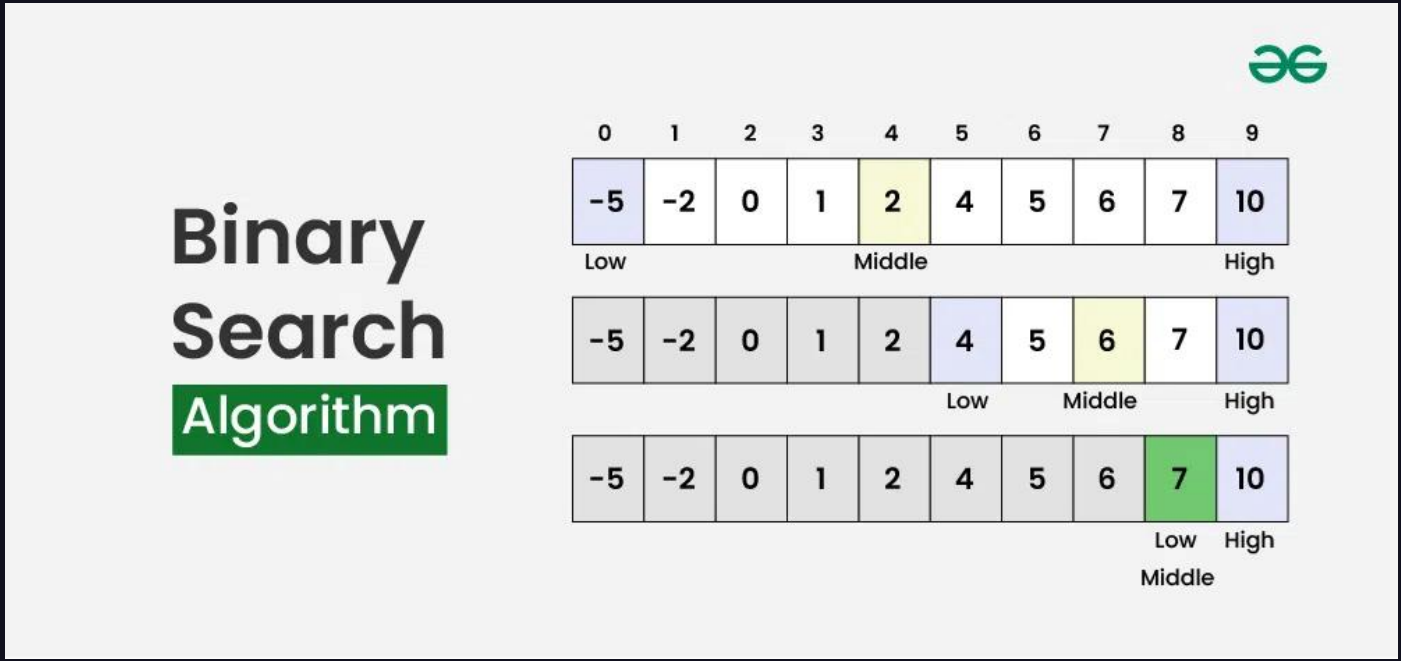
DEV.F::

# Binary Search

```
40  const binarySearch = (Array, target, low, high) => {
41      if (low > high) {
42          return -1;
43      }
44
45      let mid = Math.floor((low + high) / 2);
46
47      if (Array[mid] === target) {
48          return mid;
49      } else if (Array[mid] > target) {
50          return binarySearch(Array, target, low, mid - 1);
51      } else {
52          return binarySearch(Array, target, mid + 1, high);
53      }
54  }
```

# Recomendación:

https://www.geeksforgeeks.org/binary-search/



# Min and Max

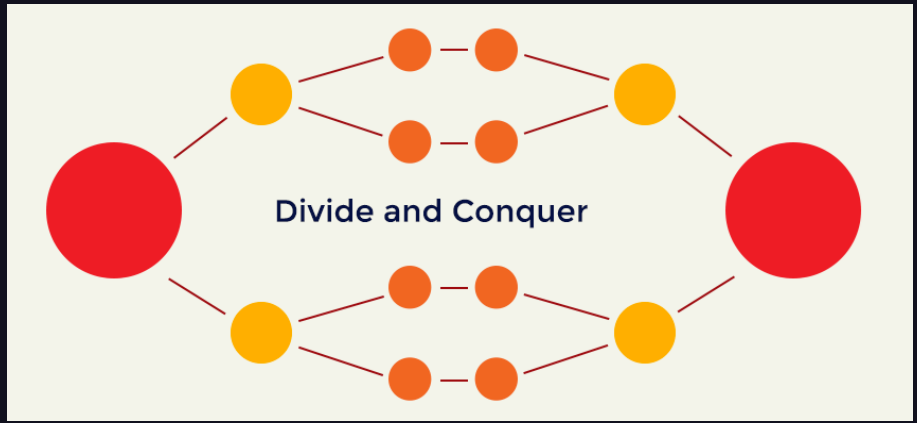
```
14  const MinMax = (Array, startIndex, lastIndex) => {
15      if(startIndex == lastIndex) {
16          return [Array[startIndex], Array[startIndex]];
17      }
18
19      if((lastIndex - startIndex) == 1) {
20          if(Array[startIndex] > Array[lastIndex]) {
21              return [Array[lastIndex], Array[startIndex]];
22          } else {
23              return [Array[startIndex], Array[lastIndex]];
24          }
25      }
26
27      else {
28          let mid = Math.floor((startIndex + lastIndex) / 2);
29          let [leftMin, leftMax] = MinMax(Array, startIndex, mid);
30          let [rightMin, rightMax] = MinMax(Array, mid + 1, lastIndex);
31
32          return [Math.min(leftMin, rightMin), Math.max(leftMax, rightMax)];
33      }
34  }
35
```

DEV.F::

# Recomendaciones para D&Q

**Divide y vencerás solo uno de los muchos métodos para resolver problemas, algunos de los usos más comunes son:**

- Algoritmos de ordenamiento (Merge Sort y Quick Sort)
- Algoritmos de búsqueda (Binary Search)
- Multiplicación de Matrices (Strassen Algorithm
- Transformadas de Fourier.



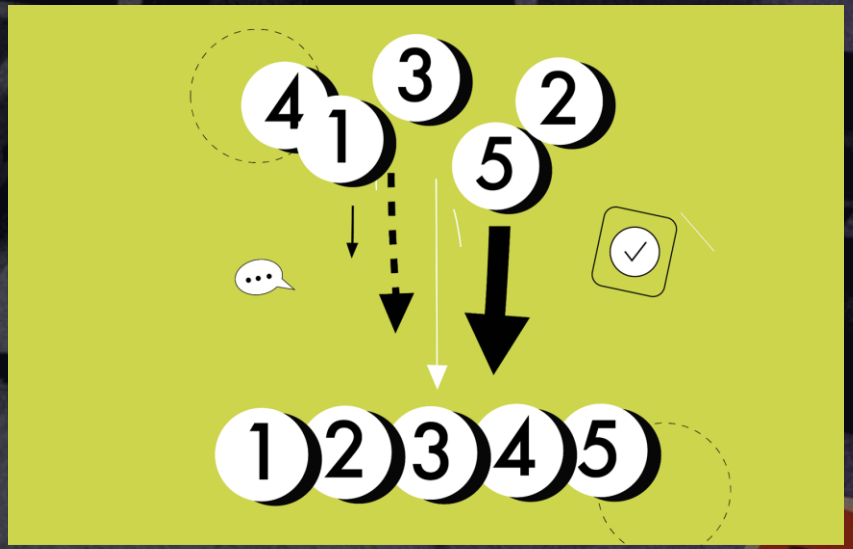
# BigO

| NOTACIÓN BIG O | ALGORITMO DE EJEMPLO |
| --- | --- |
| O(log n) | Búsqueda binaria |
| O(n) | Búsqueda simple |
| O(n * log n) | Ordenación rápida (Quicksort) |
| O(n2) | Ordenación por selección |
| O(n!) | Vendedor viajero |

La notación Big O es una manera de describir la rapidez o complejidad de un algoritmo dado. Si tu proyecto actual requiere un algoritmo predefinido, es importante entender qué tan rápido o lento es comparado con otras opciones.

# Reto de la semana: Merge y Quick Sort

- Revisa y aprende los algoritmos de:
  - Merge Sort
  - Quick Sort



# No olviden

## Las lecturas de edu.devf.la