ESCUELA POLITÉCNICA NACIONAL

ALGORITMOS FUNDAMENTALES

MÉTODO DE ORDENAMIENTO QUICK SORT

VICENTE DELGADO
ERICK BOLAÑOS
MAURICIO RODRIGUEZ
DANNY VACA

PARALELO: GR1

INTRODUCCIÓN

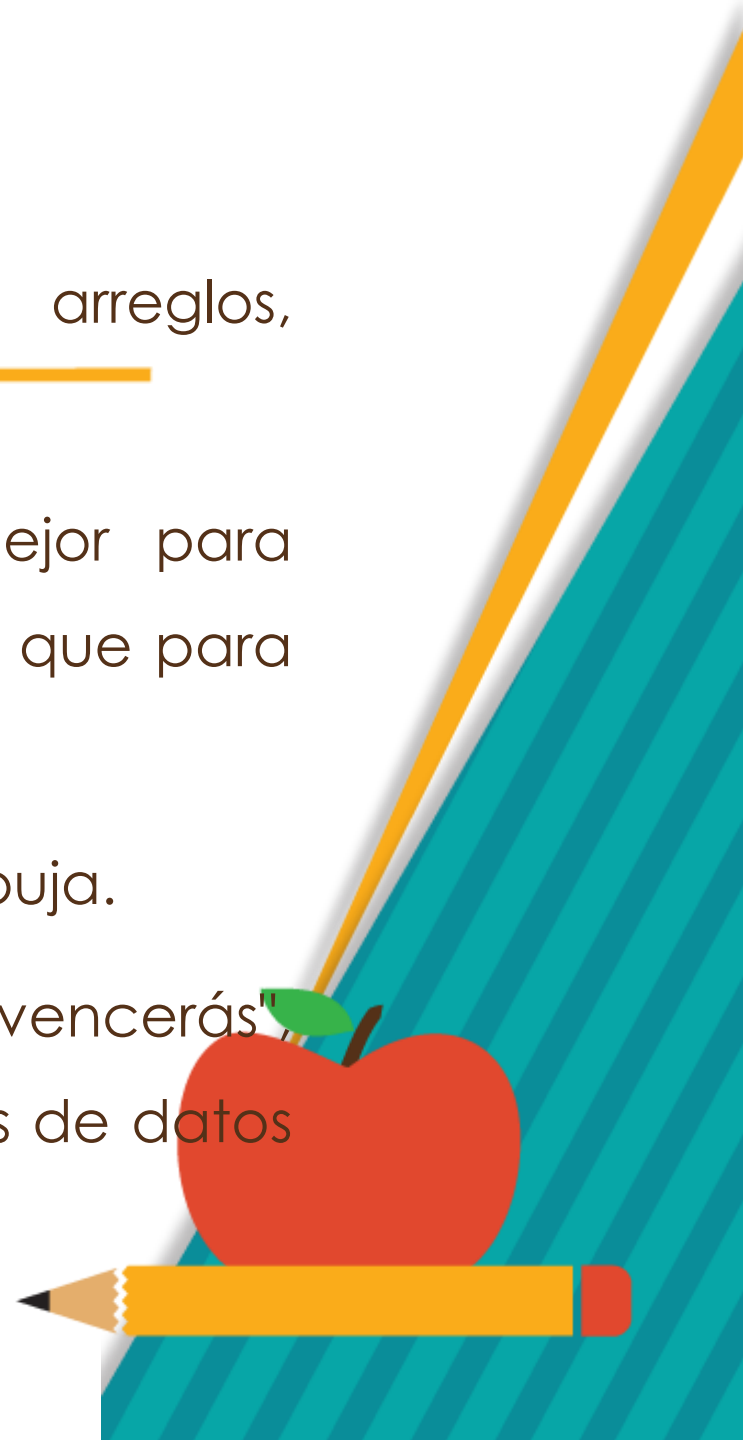
QuickSort es un algoritmo de ordenamiento de Dividir y Vencerás, creado por el científico británico en computación Charles Antony Richard Hoare. Selecciona un elemento como pivote y divide la matriz dada alrededor del pivote seleccionado.

Método de ordenamento QuickSort:

Concepto:

- El ordenamiento rápido es un algoritmo, que permite, en promedio, ordenar n elementos en un tiempo proporcional a n log n.
- Esta es la técnica de ordenamiento más rápida conocida.
- El algoritmo original es recursivo, pero se utilizan versiones iterativas para mejorar su rendimiento (los algoritmos recursivos son en general más lentos que los iterativos, y consumen más recursos).

- Método de ordenamiento ampliamente usado en arreglos, vectores y matrices.
- Tiene aparentemente la propiedad de trabajar mejor para elementos de entrada desordenados completamente, que para elementos semiordenados.
- Es un método que trabajo de forma contrario al de burbuja.
- Este tipo de algoritmos se basa en la técnica "divide y vencerás", o sea es más rápido y fácil ordenar dos arreglos o listas de datos pequeños , que un arreglo o lista grande.



Ventajas:

- Requiere de pocos recursos en comparación a otros métodos de ordenamiento.
- En la mayoría de los casos se requiere aproximadamente (N log N) operaciones.
- Ciclo interno extremadamente corto.
- No requiere espacio adicional durante la ejecucion

Desventajas:

- Se complica la implementación si la recursión no es posible.
- En el peor caso se requiere N2.
- Se debe prestar mucha atención para evitar errores que conlleven a bucles infinitos.
- No es muy bueno para ordenar conjuntos pequeños.

Aplicaciones:

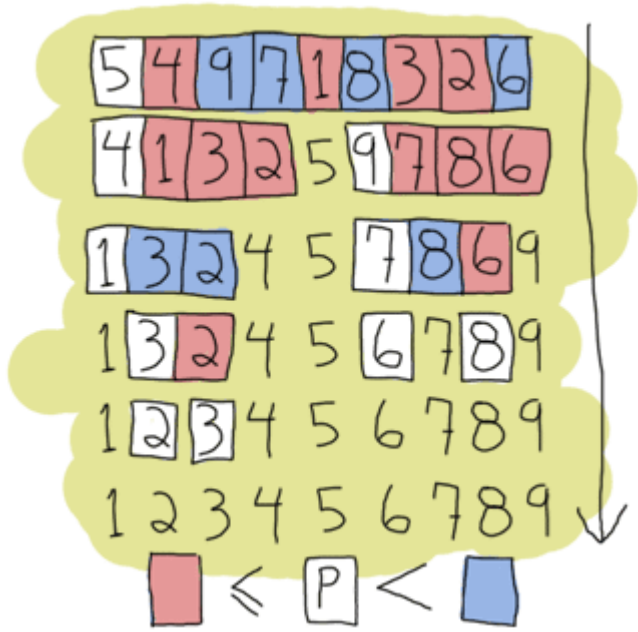
Es utilizado en algunas empresas para ordenar productos, agrupar datos estadísticos.

Ingresar ciertos datos (en desorden) tengan que sacar la mediana (Estadística)

Para ordenar una lista de números/nombres.

Utilización antes de implementar una búsqueda binaria.

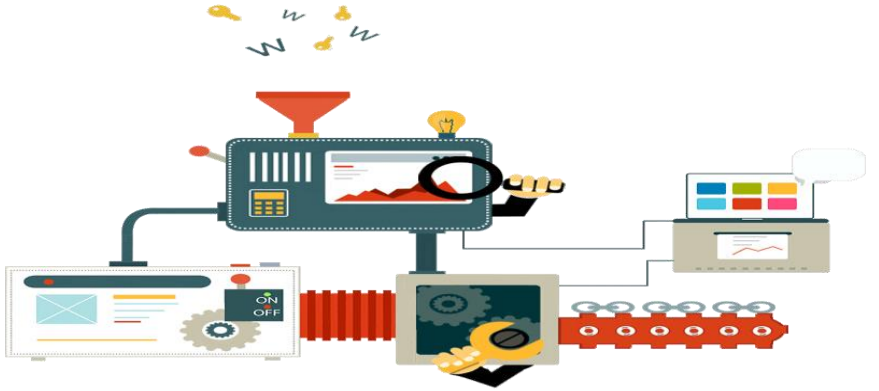
Utilizado como el método de ordenamiento en tarjetas gráficas.



¿CÓMO FUNCIONA?

- **Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.**
- **Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.**
- **La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.**
- **Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.**

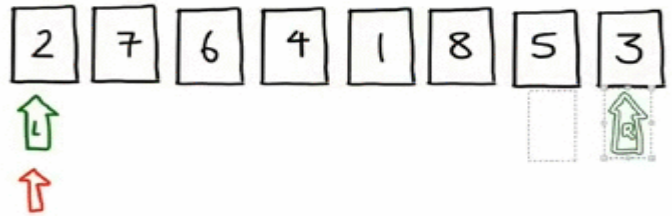
Eficiencia:



Depende de la posición en la que termine el pivote elegido.

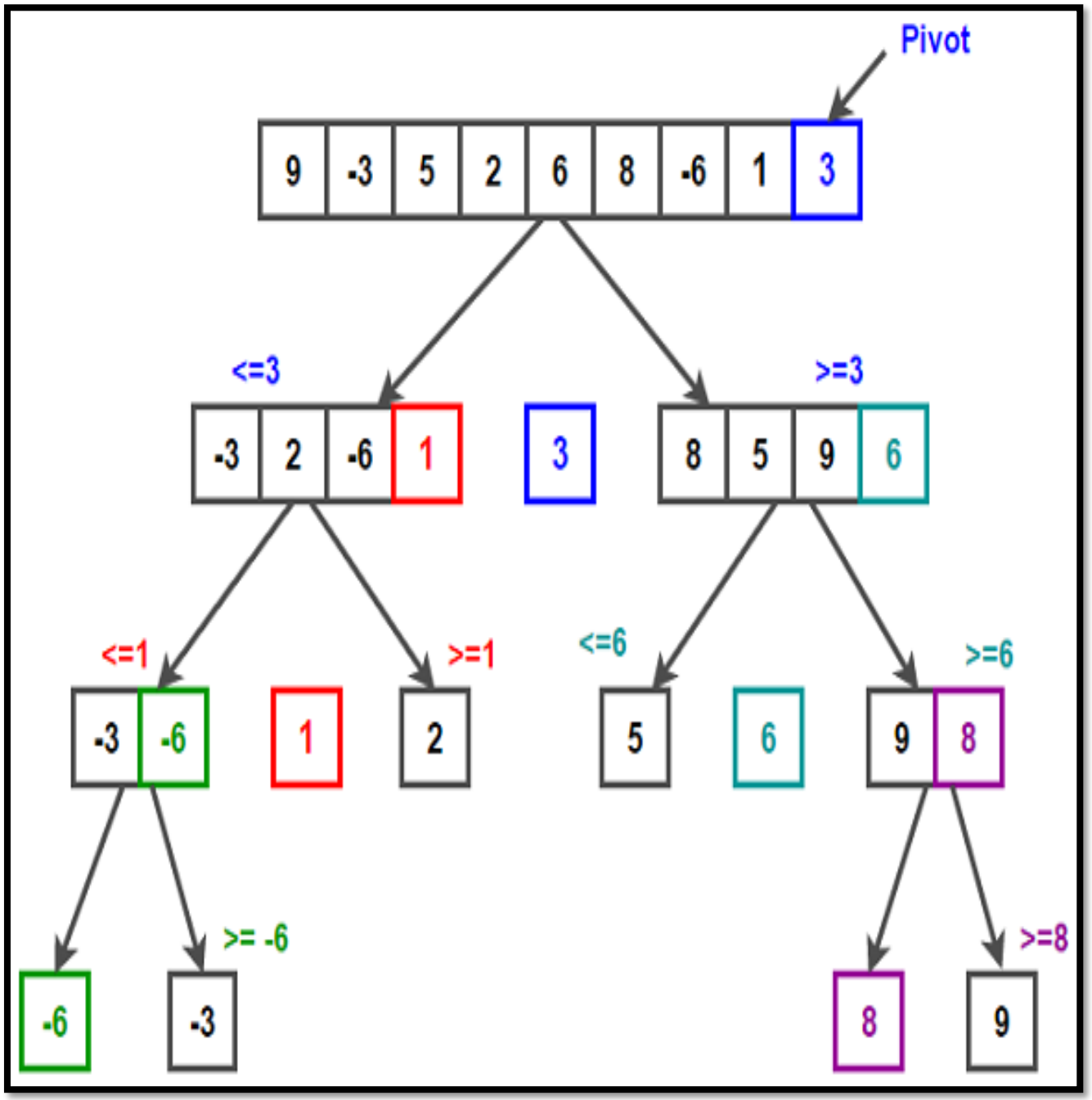
- **En el mejor caso, el pivote termina en el centro de la lista, dividiéndola en dos sublistas de igual tamaño. En este caso, el orden de complejidad del algoritmo es O(n·log n).**
- **En el peor caso, el pivote termina en un extremo de la lista. El orden de complejidad del algoritmo es entonces de $O(n^2)$. El peor caso dependerá de la implementación del algoritmo, aunque habitualmente ocurre en listas que se encuentran ordenadas, o casi ordenadas. Pero principalmente depende del pivote, si por ejemplo el algoritmo implementado toma como pivote siempre el primer elemento del array, y el array que le pasamos está ordenado, siempre va a generar a su izquierda un array vacío, lo que es ineficiente.**
- **En el caso promedio, el orden es O(n·log n).**

Elección Pivote:

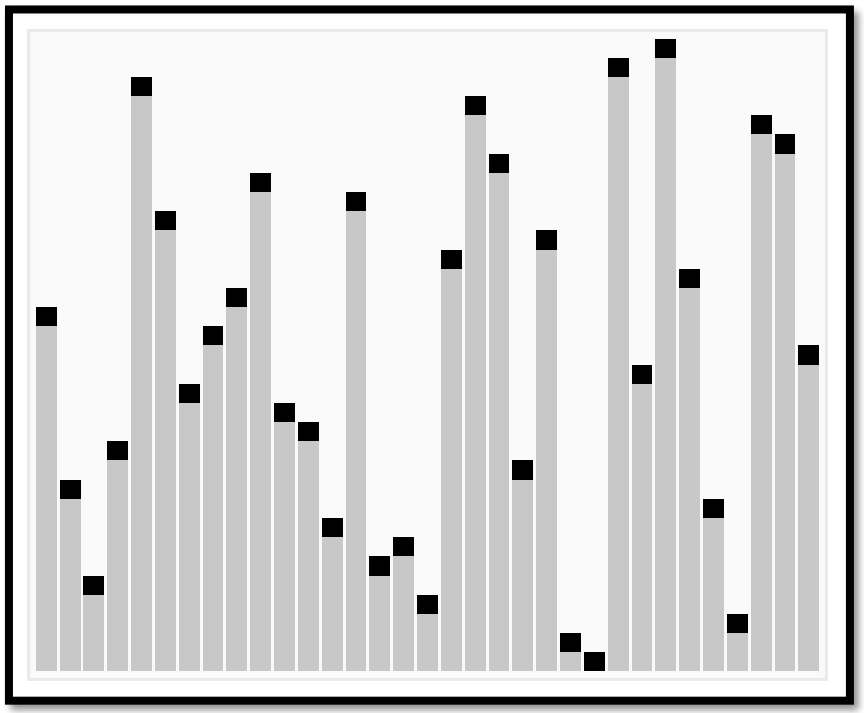


Dependiendo de la partición en que se elija, el algoritmo será más o menos eficiente.

- Tomar un elemento cualquiera como pivote tiene la ventaja de no requerir ningún cálculo adicional, lo cual lo hace bastante rápido. Sin embargo, esta elección «a ciegas» siempre provoca que el algoritmo tenga un orden de $O(n^2)$ para ciertas permutaciones de los elementos en la lista.
- Otra opción puede ser recorrer la lista para saber de antemano qué elemento ocupará la posición central de la lista, para elegirlo como pivote. Esto puede hacerse en $O(n)$ y asegura que hasta en el peor de los casos, el algoritmo sea $O(n \cdot \log n)$. No obstante, el cálculo adicional rebaja bastante la eficiencia del algoritmo en el caso promedio.
- La opción a medio camino es tomar tres elementos de la lista - por ejemplo, el primero, el segundo, y el último - y compararlos, eligiendo el valor del medio como pivote.



6 5 3 1 8 7 2 4




Gracias