**Trabajo de investigación**

**Métodos de Búsqueda**

**1. Búsqueda Lineal (Linear Search)**

**Descripción**: Busca un elemento recorriendo secuencialmente todos los elementos de una lista hasta encontrar el objetivo o llegar al final de la lista.

**Ventajas**:

* Fácil de implementar.
* No requiere que la lista esté ordenada.

**Desventajas**:

* Ineficiente para listas grandes (O(n) en el peor caso).

**2. Búsqueda Binaria (Binary Search)**

**Descripción**: Busca un elemento en una lista ordenada dividiendo repetidamente el intervalo de búsqueda a la mitad.

**Ventajas**:

* Mucho más rápida que la búsqueda lineal para listas grandes (O (log n)).
* Eficiente para listas ordenadas.

**Desventajas**:

* Requiere que la lista esté ordenada previamente.

**3. Búsqueda de Máximos y Mínimos**

**Descripción**: Encuentra el máximo o mínimo elemento en una lista recorriendo todos los elementos y manteniendo el valor máximo o mínimo encontrado hasta el momento.

**Ventajas**:

* Simple de implementar.
* No requiere que la lista esté ordenada.

**Desventajas**:

* Ineficiente para listas grandes (O(n) en el peor caso).

**4. Búsqueda Hash**

**Descripción**: Utiliza una tabla hash para encontrar un elemento en O (1) en el mejor caso. Los elementos son almacenados usando una función hash que mapea claves a índices.

**Ventajas**:

* Muy rápida para inserciones, búsquedas y eliminaciones (O (1) en el mejor caso).
* Ideal para grandes volúmenes de datos.

**Desventajas**:

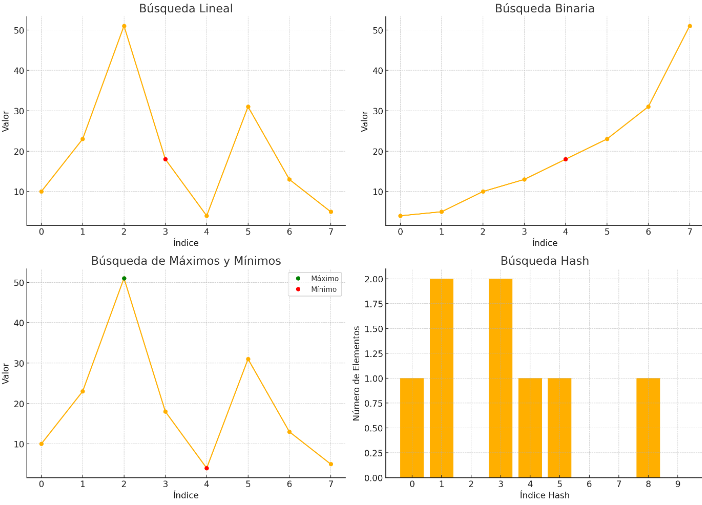
* Puede haber colisiones que disminuyen la eficiencia.
* Requiere espacio adicional para la tabla hash.

**5. Gráficos**

 Búsqueda **Lineal (Linear Search)**: Se recorre la lista secuencialmente hasta encontrar el elemento objetivo (marcado en rojo).

 Búsqueda **Binaria (Binary Search)**: La lista ordenada se divide repetidamente por la mitad hasta encontrar el elemento objetivo (marcado en rojo).

 Búsqueda **de Máximos y Mínimos**: Se recorre la lista para encontrar el valor máximo (marcado en verde) y el valor mínimo (marcado en rojo).

 Búsqueda **Hash**: Los elementos se almacenan en una tabla hash. La barra muestra el número de elementos en cada índice hash. ****

**Métodos de Ordenamiento**

**1. Burbuja (Bubble Sort)**

**Descripción**: Recorre repetidamente la lista, comparando y, si es necesario, intercambiando elementos adyacentes. Esto se repite hasta que la lista esté ordenada.

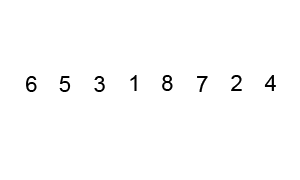
**Ventajas**:

* Fácil de entender e implementar.
* Bueno para listas pequeñas o casi ordenadas.

**Desventajas**:

* Ineficiente para listas grandes (O(n^2)).

**Gráfico**:



**2. Método de Inserción (Insertion Sort)**

**Descripción**: Construye la lista ordenada uno por uno, insertando cada nuevo elemento en su posición correcta.

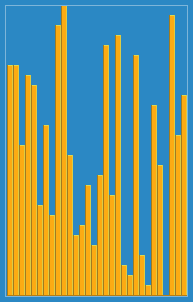
**Ventajas**:

* Eficiente para listas pequeñas o casi ordenadas.
* Simple de implementar.

**Desventajas**:

* Ineficiente para listas grandes (O(n^2)).

**Gráfico**:



**Métodos Logarítmicos de Ordenamiento**

**1. Ordenación por Mezcla (Merge Sort)**

**Descripción**: Divide la lista en sub listas más pequeñas hasta llegar a listas de un solo elemento y luego las mezcla ordenadamente para obtener la lista ordenada final.

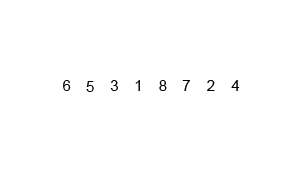
**Ventajas**:

* Eficiente para listas grandes (O (n log n)).
* Estable y garantiza el peor caso O (n log n).

**Desventajas**:

* Requiere espacio adicional (O(n)).

**Gráfico**:



**2. Ordenación Rápida (Quick Sort)**

**Descripción**: Selecciona un 'pivote' y particiona la lista en dos sub listas: una con elementos menores al pivote y otra con elementos mayores. Luego ordena recursivamente las sub listas.

**Ventajas**:

* Muy eficiente en la práctica para listas grandes (O(n log n) en promedio).
* No requiere espacio adicional significativo.

**Desventajas**:

* Peor caso O(n^2) (puede evitarse con buenas elecciones de pivote).
* No es estable.

**Gráfico**:

