Operación	Lista enlazada	Lista enlazada ordenada	Árbol binario de búsqueda	Árbol AVL	Heap	Trie
Pertenencia	$O(n)$	$O(n)$	$O(n)$	$O(\log n)$	$O(n)$	$O(m)$
Inserción	$O(1)$	$O(n)$	$O(n)$	$O(\log n)$	$O(\log n)$	$O(m)$
Borrado	$O(n)$	$O(n)$	$O(n)$	$O(\log n)$	$O(n)$	$O(m)$
Búsqueda del mínimo	$O(n)$	$O(1)$	$O(n)$	$O(\log n)$	$O(1)$	$O(n)$
Borrado del mínimo	$O(n)$	$O(1)$	$O(n)$	$O(\log n)$	$O(\log n)$	$O(n)$

Explicaciones Adicionales

1. **Lista enlazada**:
   - **Pertenencia**: Debe recorrer toda la lista para encontrar un elemento.
   - **Inserción**: Puede insertar un nuevo nodo al inicio en $O(1)$.
   - **Borrado**: Necesita buscar el nodo antes de borrarlo, lo cual toma $O(n)$.
   - **Búsqueda del mínimo**: Debe recorrer toda la lista para encontrar el mínimo.
   - **Borrado del mínimo**: Debe encontrar el mínimo primero, lo cual toma $O(n)$.
2. **Lista enlazada ordenada**:
   - **Pertenencia**: Debe recorrer la lista, similar a la lista enlazada no ordenada.
   - **Inserción**: Debe encontrar la posición correcta para mantener el orden, lo cual toma $O(n)$.
   - **Borrado**: Similar a la lista no ordenada, necesita buscar el nodo.
   - **Búsqueda del mínimo**: El primer nodo siempre es el mínimo, por lo que es $O(1)$.
   - **Borrado del mínimo**: Borrar el primer nodo es $O(1)$.
3. **Árbol binario de búsqueda (BST)**:
   - **Pertenencia**: En el peor caso, el árbol puede degenerar en una lista enlazada, tomando $O(n)$.
   - **Inserción**: Similar a la búsqueda, puede ser $O(n)$ en el peor caso.
   - **Borrado**: Igual que la búsqueda e inserción, en el peor caso es $O(n)$.
   - **Búsqueda del mínimo**: Puede requerir recorrer todo el árbol en el peor caso, $O(n)$.
   - **Borrado del mínimo**: Debe encontrar el mínimo primero, lo cual es $O(n)$.
4. **Árbol AVL**:

- **Pertenencia**: Los AVL están balanceados, por lo que siempre es $O(\log n)$.
- **Inserción**: Mantiene el balance, lo cual toma $O(\log n)$.
- **Borrado**: Similar a la inserción, manteniendo balance toma $O(\log n)$.
- **Búsqueda del mínimo**: Siempre es $O(\log n)$.
- **Borrado del mínimo**: Igual que la búsqueda del mínimo, luego un re-balanceo que es $O(\log n)$.

5. **Heap**:
   - **Pertenencia**: No optimizado para búsqueda de elementos arbitrarios, toma $O(n)$.
   - **Inserción**: Insertar un elemento y reajustar toma $O(\log n)$.
   - **Borrado**: Debe encontrar el elemento, lo cual es $O(n)$ si no es el mínimo.
   - **Búsqueda del mínimo**: El mínimo siempre está en la raíz, $O(1)$.
   - **Borrado del mínimo**: Quitar la raíz y reajustar toma $O(\log n)$.
6. **Trie**:
   - **Pertenencia**: Basado en la longitud de la clave, es $O(m)$, donde m es la longitud de la clave.
   - **Inserción**: También $O(m)$ basado en la longitud de la clave.
   - **Borrado**: $O(m)$, igual que la inserción y pertenencia.
   - **Búsqueda del mínimo**: No está optimizado para encontrar el mínimo en términos de valores naturales, puede requerir $O(n)$.
   - **Borrado del mínimo**: Similar a la búsqueda del mínimo, $O(n)$.