SEMANA 13 (junio 24, 26 ,28)

ÁRBOLES TRIE (Ejercicios)

Introducción al Problema

Problema: Imagina que estamos construyendo una aplicación de autocompletado para un buscador. Queremos que el buscador sugiera palabras a medida que el usuario escribe. Necesitamos una estructura de datos eficiente para almacenar y buscar rápidamente palabras con el mismo prefijo.

Paso 1: Presentar el Problema

Contextualización:

- Plantea una situación cotidiana: "Estás usando un motor de búsqueda y, mientras escribes 'app', el buscador te sugiere palabras como 'apple', 'application' y 'apartment'. ¿Cómo crees que el sistema sabe qué palabras sugerir tan rápido?"
- Muestra ejemplos en tiempo real de autocompletado en buscadores.

Pregunta:

- "¿Qué tipo de estructura de datos podríamos usar para resolver este problema?"

Paso 2: Introducción de los Tries

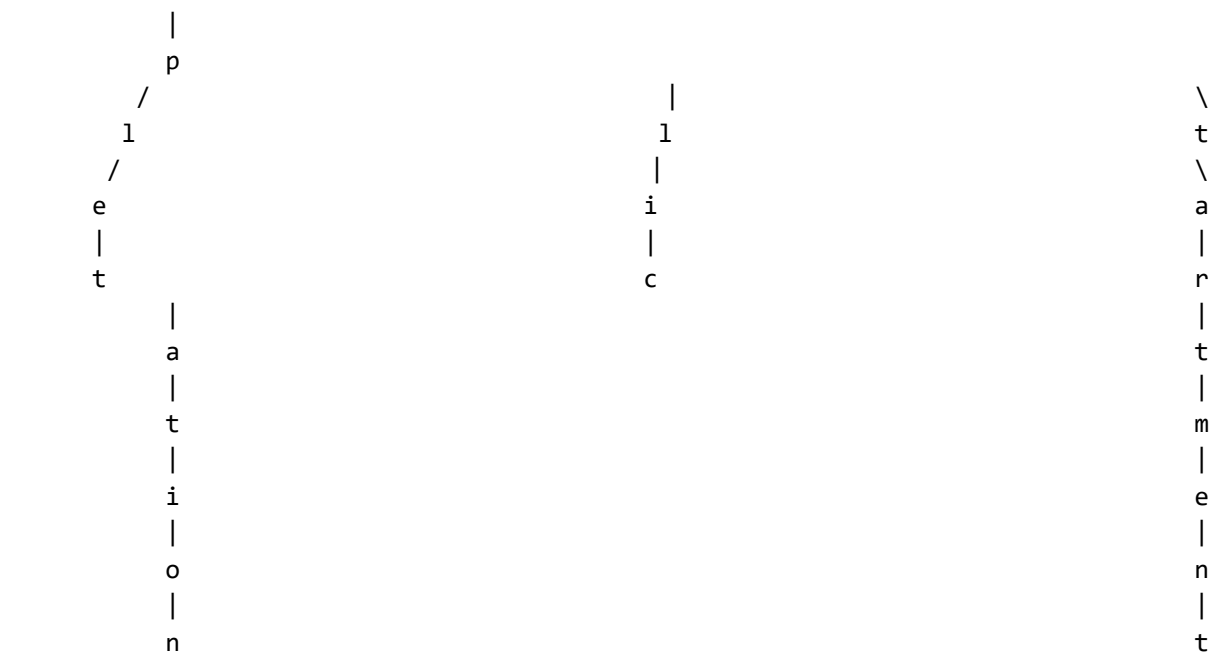
Solución Propuesta:

- Presenta los árboles Trie como una posible solución. Explica que un Trie es una estructura de datos especializada para almacenar cadenas de caracteres, donde cada nodo representa un carácter y los caminos desde la raíz a un nodo hoja representan palabras completas.

Visualización:

- Dibuja un Trie en la pizarra para las palabras 'apple', 'app', 'application' y 'apartment'.

```
(root)
   |
   a
   |
   p
```



Paso 3: Implementación Práctica

Actividad:

- Proporciona el código de la implementación básica del Trie en JavaScript (como se mostró anteriormente).
- Implementen este código y realicen pruebas insertando y buscando palabras.

Paso 4: Resolución del Problema

Aplicación Práctica:

- Formula una lista de palabras para insertar en el Trie.
- Implemente un método que devuelva todas las palabras que comienzan con un prefijo dado. Esto ayudará a emular el autocompletado del buscador.

```
class Trie {
    // Implementación existente...

    // Método para listar palabras con un prefijo dado
    wordsWithPrefix(prefix) {
        let node = this.root;
        for (let char of prefix) {
            if (!node.children[char]) {
                return []; // No hay palabras con este prefijo
            }
            node = node.children[char];
        }
```

```
        return this._collectAllWords(node, prefix);
    }

    _collectAllWords(node, prefix) {
        let words = [];
        if (node.isEndOfWord) {
            words.push(prefix);
        }
        for (let char in node.children) {
            words.push(...this._collectAllWords(node.children[char], prefix +
char));
        }
        return words;
    }
}

// Crear una instancia del Trie y probar el nuevo método
let trie = new Trie();
trie.insert("apple");
trie.insert("app");
trie.insert("application");
trie.insert("apartment");

console.log(trie.wordsWithPrefix("app")); // ["apple", "app", "application"]
```