EJEMPLO DE BK-TREE CON FUNCIONES DISCRETAS

Este código implementa una interfaz web para agregar y buscar productos utilizando un árbol BK (Burkhard-Keller Tree) en JavaScript. La interfaz permite a los usuarios agregar productos con detalles como nombre, stock, veces reordenado y categoría. Además, proporciona una funcionalidad de búsqueda que permite encontrar productos similares basándose en una métrica de distancia discreta.

Los resultados de la búsqueda se muestran dinámicamente en la página, junto con botones que reflejan los criterios de búsqueda utilizados.

ENCABEZADO DEL DOCUMENTO:

- **Metadatos y Estilos**

```
1 <head>
2    <meta  charset= "UTF-8" >
3    <meta  name= "viewport"  content= "width=device-width, initial-scale=1.0" >
4    <title> Ejemplo de BK-Tree </title>
5    <style>
6       /* Estilos CSS */
7    </style>
8 </head>
```

SECCIÓN PARA AGREGAR PRODUCTOS:

- **Campos de Entrada y Botón**

```
1          <div  class= "add-section" >
2             <h2> Agregar producto </h2>
3             <label  for= "name" > Nombre: </label>
4             <input  type= "text"  id= "name" >
5             <br>
6             <label  for= "stock" > Stock: </label>
7             <input  type= "number"  id= "stock" >
8             <br>
9             <label  for= "reordered" > Vez reordenado: </label>
10            <input  type= "number"  id= "reordered" >
11            <br>
12            <label  for= "category" > Categoría: </label>
13            <input  type= "text"  id= "category" >
14            <br>
15            <button  onclick= "addProduct()" > Agregar producto </button>
dieciséis </div>
```

- Sección para agregar productos con un borde inferior y un margen inferior.
- Campos de entrada (<input>) para nombre, stock, veces reordenado y categoría del producto.
- Botón que llama a la función addProduct al ser clicado.

SECCIÓN PARA BUSCAR PRODUCTOS:

```
<div class="search-section">
  <h2>Buscar producto</h2>
  <label for="searchName">Nombre:</label>
  <input type="text" id="searchName">
  <br>
  <label for="searchStock">Stock:</label>
  <input type="number" id="searchStock">
  <br>
  <label for="searchReordered">Vez reordenado:</label>
  <input type="number" id="searchReordered">
  <br>
  <label for="searchCategory">Categoría:</label>
  <input type="text" id="searchCategory">
  <br>
  <label for="threshold">Umbral de distancia:</label>
  <input type="number" id="threshold" value="3">
  <br>
  <button onclick="searchProduct()">Buscar producto</button>
</div>
```

- : Sección para buscar productos con un borde inferior y un margen inferior.
- Campos de entrada (<input>) para nombre, stock, veces reordenado, categoría y umbral de distancia del producto a buscar.
- <button onclick="searchProduct()">Buscar Producto</button>: Botón que llama a la función searchProduct al ser clicado.

SECCIÓN PARA MOSTRAR RESULTADOS:

```
<div class="results-section">
  <h2>Resultados</h2>
  <ul id="results"></ul>
  <div class="result-buttons" id="searchedItems"></div>
</div>
```

 : Sección para mostrar resultados con un borde inferior y un margen inferior.

 <ul id="results">: Lista sin orden que contendrá los resultados de búsqueda.

 : Contenedor para mostrar los botones de los criterios de búsqueda.

ESTRUCTURA DEL SCRIPT JAVASCRIPT

Definición de la Clase BK-Tree:

```
clase BKTree {
1     constructor(métrica) {
2       este .metric = metric;
3        este .root =  null ;
4     }
5
6     añadir artículo) {
7       si ( este . raíz ===  null ) {
8         este . raíz = { elemento : elemento, hijos : {} };
9       } else {
10        let node =  this .root;
11        while ( true ) {
12          const distance =  this .metric(item, node.item);
13          if (node.children[distancia] ===  indefinido ) {
14            nodo.hijos[distancia] = { elemento : elemento, hijos : {} };
15             break ;
dieciséis       } demás {
17            nodo = nodo.hijos[distancia];
18          }
19        }
20      }
21    }
22
23    buscar(elemento, umbral) {
24      const resultados = [];
25       const nodo_búsqueda = (nodo) => {
26        const distancia =  this.metric (item, nodo.item);
27        if (distancia <= umbral) {
28          resultados.push(nodo.item);
29        }
30        para ( const clave en nodo.children) {
31          si ( parseInt (clave) >= distancia - umbral y  parseInt (clave) <= distancia +
32    umbral) {
33            searchNode(nodo.children[clave]);
34          }
35        }
36      };
37      si ( este .root !==  null ) {
38        searchNode( este .root);
39      }
40      devolver resultados;
41    }
42  }
```

- Inicializa el árbol con una función de métrica y una raíz nula.
- Agrega un elemento al árbol. Si el árbol está vacío, el elemento se convierte en la raíz. De lo contrario, se agrega a la posición adecuada según la distancia métrica.
- Busca elementos en el árbol que están dentro del umbral de distancia especificado. Utiliza una función recursiva searchNode para explorar el árbol.

CONFIGURACIÓN DE PRODUCTOS Y BK-TREE:

```
const productos = [];
 const productoBKTree =  new BKTree((a, b) => {
    let distancia =  0 ;
    if (a.nombre !== b.nombre) distancia ++ ;
    if (a.stock !== b.stock) distancia ++ ;
    if (a.reordenado !== b.reordenado) distancia ++ ;
    if (a.categoria !== b.categoria) distancia ++ ;
    return distancia;
});
```

1
2
3
4
5
6
7
8
9

- Inicializa el árbol con una función de métrica y una raíz nula.
- Agrega un elemento al árbol. Si el árbol está vacío, el elemento se convierte en la raíz. De lo contrario, se agrega a la posición adecuada según la distancia métrica.
- Busca elementos en el árbol que están dentro del umbral de distancia especificado. Utiliza una función recursiva searchNode para explorar el árbol.

FUNCIÓN PARA AGREGAR PRODUCTOS:

```
1  función addProduct() {
2      const nombre =  documento . getElementById( 'nombre' ). valor;
3      const stock =  parseInt ( documento . getElementById( 'stock' ). valor);
4      const reordenado =  parseInt ( documento . getElementById( 'reordenado' ). valor);
5      const categoría =  documento . getElementById( 'categoría' ). valor;
```

```
6
7     const producto = { nombre, stock, reordenado, categoría };
8     productos.push(producto);
9     productoBKTree.add(producto);
10
11    alert( ` Producto ${name} agregado exitosamente ! ` );
12 }
```

Obtiene los valores de los campos de entrada, crea un objeto de producto, lo agrega al array de productos y al árbol BK, y muestra una alerta confirmando la adición.

FUNCIÓN PARA BUSCAR PRODUCTOS:

```
          función searchProduct() {
1             const searchName =  documento . getElementById( 'searchName' ). value;
2             const searchStock =  parseInt ( documento . getElementById( 'searchStock ' ).
3         value
4             ); const searchReordered =  parseInt ( documento . getElementById(
5         'searchReordered' ). value);
6             const searchCategory =  documento . getElementById( 'searchCategory' ). value;
7             const umbral =  parseInt ( documento . getElementById( 'umbral' ). value);
8
9             constante buscarItem = {
10                nombre : nombreBuscar,
11                stock : buscarStock,
12                reordenado : buscarReordenado,
13                Categoría : BuscarCategoría
14            };
15
dieciséis     const resultados = productBKTree.search(searchItem, umbral);
17
18            const resultsContainer =  document .getElementById( 'resultados' );
19            resultadosContainer.innerHTML =  '' ;
20
21            si (resultados.longitud ===  0 ) {
22                resultsContainer.innerHTML =  '<li>No se encontraron productos que coincidan
23        con los criterios.</li>' ;
24            } demás {
25                resultsContainer.innerHTML = results.map(product =>
26                    `< li > ${product.name} - Stock : ${product.stock}, Reordenado :
27        ${product.reordered}, Categoría : $ {product.category} </li> `
28                ).join( '' ) ;
29            }
30
31            const searchedItems =  documento . getElementById( 'searchedItems' );
32            searchedItems.innerHTML =
33                `< botón > ${nombreDeBúsqueda} < /botón>
34                < botón > ${StockDeBúsqueda} < /botón>
35                < botón > ${ReordenadoDeBúsqueda} < /botón>
                  < botón > ${CategoríaDeBúsqueda} < /botón>
```

< botón > ${umbral} < /botón>`;
}

Obtiene los valores de los campos de entrada para la búsqueda, crea un objeto de búsqueda, realiza la búsqueda en el árbol BK, y actualiza la lista de resultados y los botones con los criterios de búsqueda.