Informe – entrega preliminar

Alumno: **Carranza Braian**

Comisión: **03**

Explicación de los puntos entregados:

1. **AgregarDato (ArbolGeneral<DatoDistancia> arbol, DatoDistancia dato):** insertar un nuevo dato en el árbol BK siguiendo las reglas del BK-tree.

   Código:

```
2 referencias
public void AgregarDato(ArbolGeneral<DatoDistancia> arbol, DatoDistancia dato)
{
    //calculamos la distancia entre el dato a agregar y el nodo actual
    int distancia = CalcularDistancia(arbol.getDatoRaiz().texto, dato.texto);

    //buscamos si ya existe un hijo con esa distancia
    ArbolGeneral<DatoDistancia> hijoExistente = null;

    foreach (var hijo in arbol.getHijos()) {
        if (CalcularDistancia(arbol.getDatoRaiz().texto, hijo.getDatoRaiz().texto) == distancia) {
            hijoExistente = hijo;
            break;
        }
    }

    //si no existe, lo agregamos como hijo
    if (hijoExistente == null) {
        arbol.agregarHijo(new ArbolGeneral<DatoDistancia>(dato));
    }

    //si existe, agregamos el dato al hijo existente
    else
    {
        AgregarDato(hijoExistente, dato);
    }
}
```

   1. Calculamos la distancia del nuevo dato con el nodo actual.
   2. Buscamos si hay un hijo ya en ese “nivel de distancia”, es decir, revisa si ya hay un hijo a esa misma distancia:
      - Si no hay, crea un nuevo hijo con el dato.
      - Si hay, baja a ese hijo y repite el proceso.

3. **Buscar (ArbolGeneral<DatoDistancia> arbol, string elementoABuscar, int umbral, List<DatoDistancia> collected):** buscar en el árbol todos los nodos con “texto” sea parecido al elemento que buscamos, según el nivel de tolerancia (umbral).

```
2 referencias
public void Buscar(ArbolGeneral<DatoDistancia> arbol, string elementoABuscar, int umbral, List<DatoDistancia> collected)
{
    //si el nodo existe
    if (arbol == null)
    {
        return;
    }

    int distancia = CalcularDistancia(arbol.getDatoRaiz().texto, elementoABuscar);

    //si la distancia esta dentro del umbral, agregamos el nodo a la lista
    if(distancia <= umbral)
    {
        collected.Add(arbol.getDatoRaiz());
    }

    //recorremos los hijos dentro del rango
    foreach(var hijo in arbol.getHijos())
    {
        int distanciaHijo = CalcularDistancia(arbol.getDatoRaiz().texto, hijo.getDatoRaiz().texto);
        if(distanciaHijo >= distancia - umbral && distanciaHijo <= distancia + umbral)
        {
            Buscar(hijo, elementoABuscar, umbral, collected);
        }
    }
```

1. Verifica si el nodo existe.
2. Calcula la distancia entre el nodo y el elemento buscado.
3. Si la distancia está dentro del umbral, agrega el nodo a los resultados.
4. Recorre recursivamente solo los hijos que podrían cumplir el criterio de distancia, usando el rango {distancia – umbral, distancia + umbral}.

Esto permite buscar rápido porque solo se revisan nodos que podrían estar dentro del umbral, esto evita recorrer todo el árbol.

¿Cómo planeo hacer los siguientes tres métodos?

- Consulta1: realizaría un recorrido recursivo desde la raíz del árbol, verificando en cada nodo si no tiene hijos (es hoja). En ese caso, se agrega su dato a un resultado acumulado. Donde al final se obtiene un texto que contiene todas las hojas del árbol.
- Consulta2: realizaría un recorrido recursivo desde la raíz, manteniendo el camino acumulado desde la raíz hasta cada nodo. Cada vez que se llega a una hoja, se agrega el camino completo al resultado final. Donde todo esto permitiría mostrar todos los caminos posibles hasta las hojas del árbol.
- Consulta3: utilizaría un recorrido por niveles, donde cada nodo se agrega al resultado junto con el nivel en que se encuentra. Donde al finalizar, se obtiene un texto que muestra los nodos organizados según el nivel.