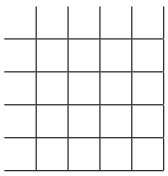
Analisis de Algoritmos
Tarea 2

Nombres de los alumnos del grupo

June 1, 2025

1 Casos de Prueba

prueba distintos para una función `EditDistanceDeleteInsert`(S, T) la cual calcula la distancia de edición más corta de una cadena de texto S a una cadena de texto T con los operadores de edición $\{$`Delete`, `Insert`$\}$. Para cada caso de prueba, justifique la correctitud del output. 1.1 Cadenas seleccionadas: `(...)`, `(...)`, `(...)` y `(...)`. La matriz de distancia es simétrica, porque $d(S, T) = d(T, S)$, pero se consideran todos los valores como casos de pruebas del código:



A parte de las distancias entre cadenas iguales, las otras distancias se justifican de la manera siguiente:

- $d(,) = d(,) =$ por (...);
- $d(,) = d(,) =$ por ;
- $d(,) = d(,) =$ por ;
- $d(,) = d(,) =$ por ;
- $d(,) = d(,) =$ por ; y
- $d(,) = d(,) =$ por .

2 Fórmula Recursiva

más corta de una cadena de texto S a una cadena de texto T con los operadores de edición $\{$`Delete`, `Insert`$\}$. 1.2 **Caso base**: **Caso recursivo**: La función recursiva sería:

3 Implementación

paradigmas siguientes: `editDistanceDP`; `editDistanceDPOptimized`; previamente definidos.

3.1 Programa recursivo

```
int editDistanceRecursive(const string &A, const string &B, int i, int j) {
(...)
}
if (editDistance("(...)","(...)") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
```

3.2 Programa recursivo con Memoización

```
int editDistanceMemo(string &A, string &B, int i, int j, vector<vector<int>> &memo)
{
}
if (editDistance("(...)","(...)") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
```

3.3 Programación dinámica

```
int editDistanceDP(const string &A, const string &B) {
}
if (editDistance("(...)","(...)") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
```

```
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
```

3.4 Programación dinámica optimizada

```
int editDistanceDPOptimized(const string &A, const string &B) {
}
if (editDistance("(...)","(...)") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
if (editDistance("","") != ) return -1;
```

4 Complejidad

de tiempo computacional como en términos de espacio.

4.1 Programa recursivo

4.2 Programa recursivo con Memoización

4.3 Programación dinámica

4.4 Programación dinámica optimizada

5 Protocolo Experimental

implementaciones distintas calculando la distancia de edición con los operadores de edición {`Delete`, `Insert`}.

6 Experimentación

textos. Textos escogidos: Provienen del texto (`https://www.gutenberg.org/cache/epub/(...)`)

- `text1`: Las dos primeras palabras de (...): `XXXX YYYY` ((...) bytes).
- `text2`: La tercera y cuarta palabra de (...): `VVVV WWWW` ((...) bytes).

Figure 1: Resultados Edit Distance

Figure 2: Resultados Edit Distance ajustados

- `text3`: Párrafo 2 del Capítulo X ((...) kB).
- `text4`: Capítulo 2 ((...) kB).

La Figura 1 muestra el tiempo de ejecución y el peak de memoria de la ejecución de los 4 algoritmos para las combinaciones de los 4 textos. La Figura 2 muestra el mismo resultado, pero excluyendo los valores extremos para una mejor observación. Observaciones:

1.
2.
3. (...)

7 Conclusión

complejidad.