URL VIDEOS

Explicación enunciado del problema: https://youtu.be/MsoUykfRZoE
Explicación código: https://youtu.be/dWWcYVOTlhw
Análisis de complejidad: https://youtu.be/N_R7tEwcBhg

ENUNCIADO DEL PROBLEMA

Escriba un método para encontrar la cadena prefijo común más larga entre un arreglo de strings.

Si no hay un prefijo común, retorna una cadena vacía “”

Restricciones:

- 1 <= strs.length <= 200
- 0 <= strs[i].length <= 200
- strs[i] Consiste en solo letras en inglés en minúsculas si no está vacía.

URL DE GITLAB

Ejercicios Proyecto ADA / Ejercicio 10 - Longest Common Prefix · GitLab

MÉTODO CÁNDIDO

```java
public String metodoCandido(String[] palabras) {
    String primera = palabras[0];
    int minLen = primera.length();
    int cantidadPalabras = palabras.length;

    for(int i = 0; i<cantidadPalabras; i++) {
        minLen = Math.min(minLen, palabras[i].length());
    }
    for(int i = minLen; i ⩾ 1; i--) {
        String prefijo = primera.substring(0, i);
        boolean valido = true;
        for(int j = 0; j < cantidadPalabras; j++) {
            for (int k = 0; k<i; k++) {
                if (palabras[j].charAt(k) ≠ prefijo.charAt(k)) {
                    valido = false;
                    break;
                }
            }
            if(!valido) break;
        }
        if (valido) {
            return prefijo;
        }
    }
    return "";
}
```

Eficacia	El método es eficaz, puesto a que tiene una solución para todas las entradas válidas ingresadas.
	Instancia 1: PALABRAS: flower, flow, flight RESULTADO MÉTODO CÁNDIDO: fl
	Instancia 2: PALABRAS: dog, racecar, car RESULTADO MÉTODO CÁNDIDO:
	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO MÉTODO CÁNDIDO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Eficiencia	El método no es el más eficiente ya que para cada prefijo estamos buscando cada palabra y en cada palabra estamos haciendo la comprobación letra por letra, por lo que en casos extremos tomaría mucho tiempo.
	Instancia 1: PALABRAS: flower, flow, flight RESULTADO MÉTODO CÁNDIDO: fl **Tiempo del proceso: 58 microsegundos**
	Instancia 2: PALABRAS: dog, racecar, car RESULTADO MÉTODO CÁNDIDO: **Tiempo del proceso: 105 microsegundos**
	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj",

	"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO MÉTODO CÁNDIDO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa **Tiempo del proceso: 110 microsegundos**
Correctitud	El algoritmo es correcto, ya que da un resultado esperado para todas las entradas posibles
	Instancia 1: PALABRAS: flower, flow, flight RESPUESTA ESPERADA: fl RESULTADO MÉTODO CÁNDIDO: fl
	Instancia 2: PALABRAS: dog, racecar, car RESPUESTA ESPERADA: RESULTADO MÉTODO CÁNDIDO:
	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf",

	"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO ESPERADO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa RESULTADO MÉTODO CÁNDIDO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Completitud	El método es completo, ya que la lógica usada para buscar el prefijo común más grande es correcta, aunque es lento asegura siempre un resultado correcto, si es que este existe
	Instancia 1: PALABRAS: flower, flow, flight RESPUESTA ESPERADA: fl RESULTADO MÉTODO CÁNDIDO: fl
	Instancia 2: PALABRAS: dog, racecar, car RESPUESTA ESPERADA: RESULTADO MÉTODO CÁNDIDO:
	Instancia 3:

	PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO ESPERADO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa RESULTADO MÉTODO CÁNDIDO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

MÉTODO 2

```java
public String metodoOptimo(String[] palabras) {
    if (palabras==null || palabras.length == 0) {
        return "";
    }

    return metodoOptimo(palabras, 0, palabras.length-1);
}

public String metodoOptimo (String[] palabras, int izq, int der) {
    if (izq == der) return palabras[izq];
    int mitad = (der+izq) /2;
    String stringIzq = metodoOptimo(palabras, izq, mitad);
    String stringDer = metodoOptimo(palabras, mitad+1, der);

    return buscarPrefijo(stringIzq, stringDer);
}

public String buscarPrefijo(String strIzq, String strDer) {
    int limite = Math.min(strIzq.length(), strDer.length());
    for(int i = 0; i<limite; i++) {
        if (strIzq.charAt(i) != strDer.charAt(i)) {
            return strIzq.substring(0, i);
        }
    }
    return strIzq.substring(0, limite);
}
```

Eficacia	El método es eficaz, puesto a que tiene una solución para todas las entradas válidas ingresadas
	Instancia 1: PALABRAS: flower, flow, flight RESULTADO MÉTODO ÓPTIMO: fl
	Instancia 2: PALABRAS: dog, racecar, car RESULTADO MÉTODO ÓPTIMO:
	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO MÉTODO ÓPTIMO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

Eficiencia	El método es eficiente ya que solamente dividimos el arreglo de palabras hasta solo tener 2, luego vemos el prefijo entre estas y vamos subiendo la respuesta, no tenemos que recorrer cada palabra para cada prefijo posible.
	Instancia 1: PALABRAS: flower, flow, flight RESULTADO MÉTODO ÓPTIMO: fl **Tiempo del proceso: 31 microsegundo**
	Instancia 2: PALABRAS: dog, racecar, car RESULTADO MÉTODO ÓPTIMO: **Tiempo del proceso: 67 microsegundos**
	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh",

	"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO MÉTODO ÓPTIMO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa **Tiempo del proceso: 75 microsegundos**
Correctitud	El algoritmo es correcto, ya que da un resultado esperado para todas las entradas posibles
	Instancia 1: PALABRAS: flower, flow, flight RESPUESTA ESPERADA: fl RESULTADO MÉTODO ÓPTIMO: fl
	Instancia 2: PALABRAS: dog, racecar, car RESULTADO ESPERADO: RESULTADO MÉTODO ÓPTIMO:
	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad",

	"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO ESPERADO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa RESULTADO MÉTODO ÓPTIMO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Completitud	El método es completo, ya que la lógica usada para buscar el prefijo común más grande es correcta, es rápido y asegura siempre un resultado correcto, si es que este existe
	Instancia 1: PALABRAS: flower, flow, flight RESPUESTA ESPERADA: fl RESULTADO MÉTODO ÓPTIMO: fl
	Instancia 2: PALABRAS: dog, racecar, car RESULTADO ESPERADO: RESULTADO MÉTODO ÓPTIMO:

	Instancia 3: PALABRAS: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaab", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaad", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaae", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaf", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabg", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabh", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabi", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabj", "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabk RESULTADO ESPERADO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa RESULTADO MÉTODO ÓPTIMO: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa