

URL VIDEOS

Explicación enunciado del problema: <https://youtu.be/MsoUykfRZoE>

Explicación código: <https://youtu.be/dWWcYVOTIhw>

Análisis de complejidad: https://youtu.be/N_R7tEwcBhq

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:

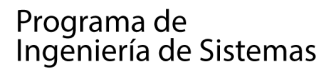
- $1 \leq \text{strs.length} \leq 200$
- $0 \leq \text{strs}[i].\text{length} \leq 200$
- `strs[i]` Consiste en solo letras en inglés en minúsculas si no está vacía.

URL DE GITLAB

[Ejercicios Proyecto ADA / Ejercicio 10 - Longest Common Prefix · GitLab](#)

MÉTODO CÁNDIDO

```
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

Instancia 1:

RESULTADO MÉTODO CÁNDIDO: fl

Instancia 2:

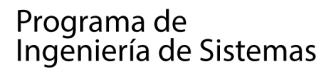
RESULTADO MÉTODO CÁNDIDO:

Instancia 3:

PALABRAS: "aab",
 "aac",
 "aad",
 "aaae",
 "aaf",
 "aaabg",
 "aaabh",
 "aaabi",
 "aaabj",
 "aaabk"

RESULTADO MÉTODO CÁNDIDO: aaa

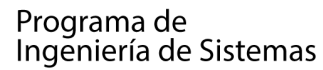
Eficiencia	El método no es el más eficiente ya que para cada prefijo estamos buscando cada palabra y en cada palabra estamos haciendo la comprobación letra por letra, por lo que en casos extremos tomaría mucho tiempo.
	<p>Instancia 1:</p> <p>PALABRAS: flower, flow, flight</p> <p>RESULTADO MÉTODO CÁNDIDO: fl Tiempo del proceso: 58 microsegundos</p>
	<p>Instancia 2:</p> <p>PALABRAS: dog, racecar, car</p> <p>RESULTADO MÉTODO CÁNDIDO: Tiempo del proceso: 105 microsegundos</p>
	<p>Instancia 3:</p> <p>PALABRAS: "aab", "aac", "aad", "aae", "aaf", "aaabg", "aaabh", "aaabi", "aaabj",</p>

[illegible]

	<p>"aaabg", "aaabh", "aaabi", "aaabj", "aaabk"</p> <p>RESULTADO ESPERADO: aa RESULTADO MÉTODO CÁNDIDO: aa</p>
Completitud	<p>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</p>
	<p>Instancia 1:</p> <p>PALABRAS: flower, flow, flight</p> <p>RESPUESTA ESPERADA: fl RESULTADO MÉTODO CÁNDIDO: fl</p>
	<p>Instancia 2:</p> <p>PALABRAS: dog, racecar, car</p> <p>RESPUESTA ESPERADA: RESULTADO MÉTODO CÁNDIDO:</p>
	<p>Instancia 3:</p>

	<p>PALABRAS: "aab", "aaac", "aaad", "aaae", "aaaf", "aabg", "aabh", "aabi", "aabj", "aabk"</p> <p>RESULTADO ESPERADO: aa RESULTADO MÉTODO CÁNDIDO: aa</p>
--	---

```
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

Instancia 1:

RESULTADO MÉTODO ÓPTIMO: fl

Instancia 2:

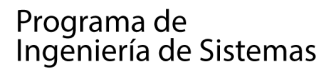
RESULTADO MÉTODO ÓPTIMO:

Instancia 3:

PALABRAS: "aaab",
 "aac",
 "aad",
 "aae",
 "aaf",
 "aabg",
 "aabgh",
 "aabhi",
 "aabji",
 "aabkj",
 "aabkl"

RESULTADO MÉTODO ÓPTIMO: aaa

[illegible]

[illegible]

	<pre> "aaae", "aaaf", "aabg", "aabh", "aabi", "aabj", "aabk </pre> <p>RESULTADO ESPERADO: aa</p> <p>RESULTADO MÉTODO ÓPTIMO: aa</p>
Completitud	<p>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</p>
	<p>Instancia 1:</p> <p>PALABRAS: flower, flow, flight</p> <p>RESPUESTA ESPERADA: fl</p> <p>RESULTADO MÉTODO ÓPTIMO: fl</p>
	<p>Instancia 2:</p> <p>PALABRAS: dog, racecar, car</p> <p>RESULTADO ESPERADO:</p> <p>RESULTADO MÉTODO ÓPTIMO:</p>

Instancia 3:

PALABRAS: "aab",
"aaac",
"aaad",
"aaae",
"aaaf",
"aabg",
"aabh",
"aabi",
"aabj",
"aabk"

RESULTADO ESPERADO: aa

RESULTADO MÉTODO ÓPTIMO: aa