

**JOSE GABRIEL GARCIA PARDO**

**ALFREDO TORREGROSA LLORET**

**BWA**



# **INEX RECUA**

**JAVIER DE LA TORRE COSTA**

**AITANA PASCUAL BELDA**



# Índice

1. Introducción
2. Materiales y métodos
3. Resultados
4. Conclusiones
5. Bibliografía



# Índice



Introducción

2.

Materiales y métodos

3.

Resultados

4.

Conclusiones

5.

Bibliografía



# Introducción

## ¿Qué es un algoritmo recursivo?



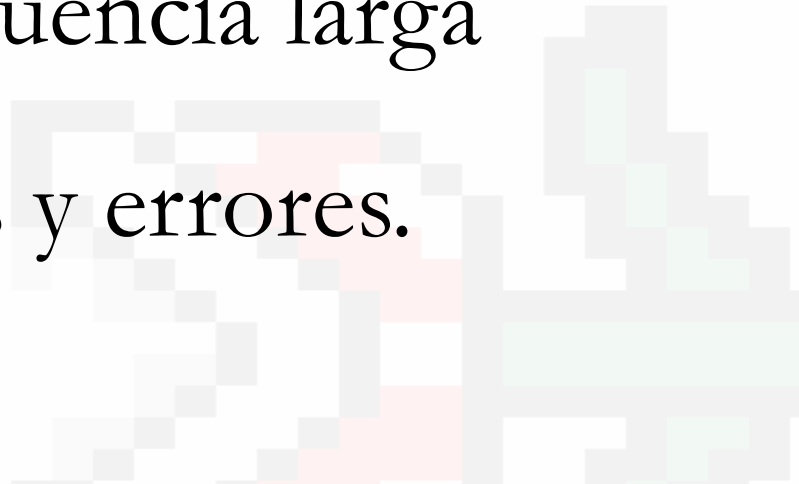
La recursividad es una técnica de programación que se utiliza para realizar una llamada a una función desde ella misma.

# Introducción

## OBJETIVO

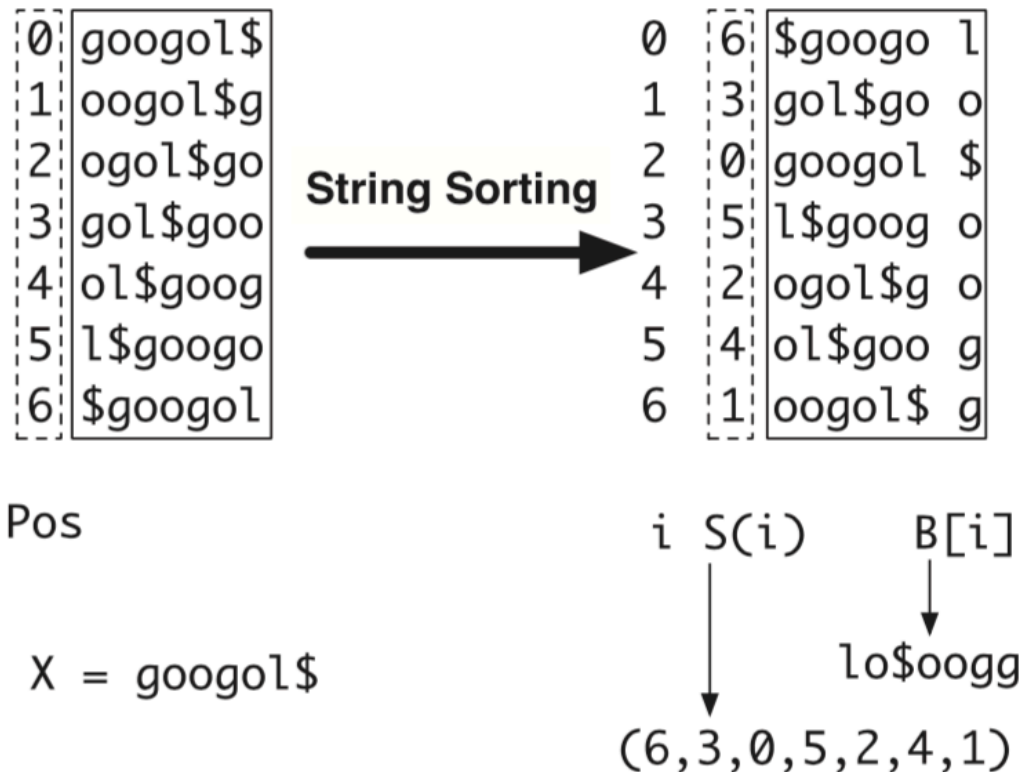
---

Implementación del algoritmo BWA para alinear secuencias cortas dentro de una secuencia larga de referencia, permitiendo huecos y errores.

A decorative graphic in the bottom right corner consisting of a grid of squares in various shades of gray, red, and green, arranged in a pattern that suggests a DNA sequence or a data matrix.

# Introducción

## El algoritmo BWA



## Construcción del Suffix Array

Si el string  $W$  es un substring de  $X$ , la posición de cada ocurrencia de  $W$  en  $X$  va a darse en un determinado intervalo del suffix array.

$$\underline{R}(W) = \min\{k : W \text{ is the prefix of } X_{S(k)}\}$$

$$\overline{R}(W) = \max\{k : W \text{ is the prefix of } X_{S(k)}\}$$

# Introducción

## El algoritmo BWA

---

Exact matching: Backward search

Sea:

- $C(a)$  el número de símbolos en  $X[0,n-2]$  que son lexicográficamente menores que  $a$ .
- $O(a,i)$  el número de ocurrencias de  $a$  en  $B[0,i]$

Si  $W$  es un substring de  $X$ :

$$\underline{R}(aW) = C(a) + O(a, \underline{R}(W) - 1) + 1$$

$$\overline{R}(aW) = C(a) + O(a, \overline{R}(W))$$

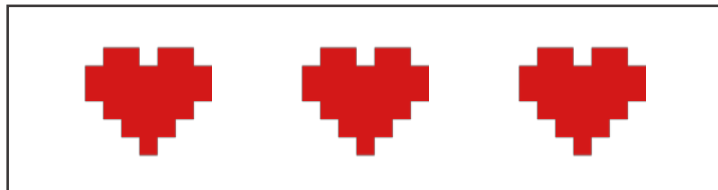
# Introducción

## El valor $Z$ y el vector $D$ :

Sobre este par de vectores recae la eficiencia del algoritmo BWA, supongamos por ejemplo el siguiente caso:

El **valor  $Z$**  representaría el número de errores restantes que nos podemos permitir.

$$Z=3$$



Las acciones de **inserción**, **sustitución** o **delección** supondrán disminuir en uno el valor de  $Z$ .

*Cuanto más alto, más permisivos a fallos estaremos siendo*



# Introducción

## El valor $Z$ y el vector $D$ :

El **vector  $D$**  se calcula antes de entrar en el algoritmo InexRecur.

Es uno de los puntos más importantes del mismo, y representa el número de fallos que debemos disponer antes de entrar a realizar operaciones en cada posición:

```
CALCULATED(W)
   $k \leftarrow 1$ 
   $l \leftarrow |X| - 1$ 
   $z \leftarrow 0$ 
  for  $i=0$  to  $|W| - 1$  do
     $k \leftarrow C(W[i]) + O'(W[i], k - 1) + 1$ 
     $l \leftarrow C(W[i]) + O'(W[i], l)$ 
    if  $k > l$  then
       $k \leftarrow 1$ 
       $l \leftarrow |X| - 1$ 
       $z \leftarrow z + 1$ 
   $D(i) \leftarrow z$ 
```

# Introducción

## El valor $Z$ y el vector $D$ :

Supongamos...

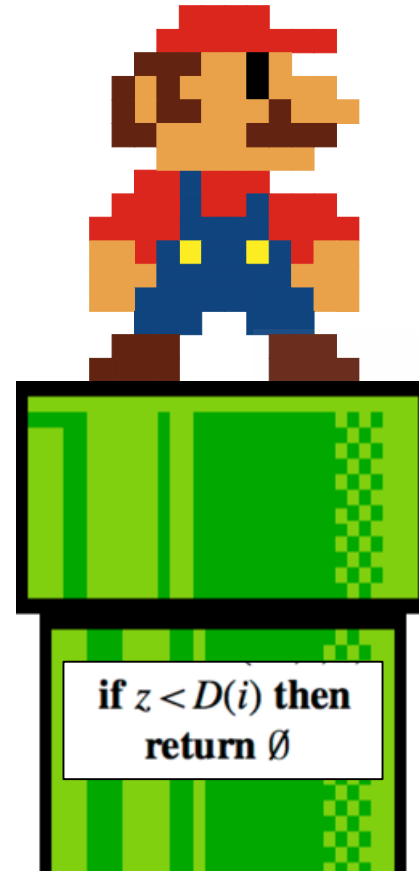


$Z = 2$

$D = [0, 1, 1]$



$Z = 2$   
 $D(i) = 1$   
 $2 > 1$   
**Entramos!**



```
INEXRECUR( $W, i, z, k, l$ )  
  if  $z < D(i)$  then  
    return  $\emptyset$ 
```

```
if  $z < D(i)$  then  
  return  $\emptyset$ 
```

# Introducción

## El valor $Z$ y el vector $D$ :

Supongamos ahora...

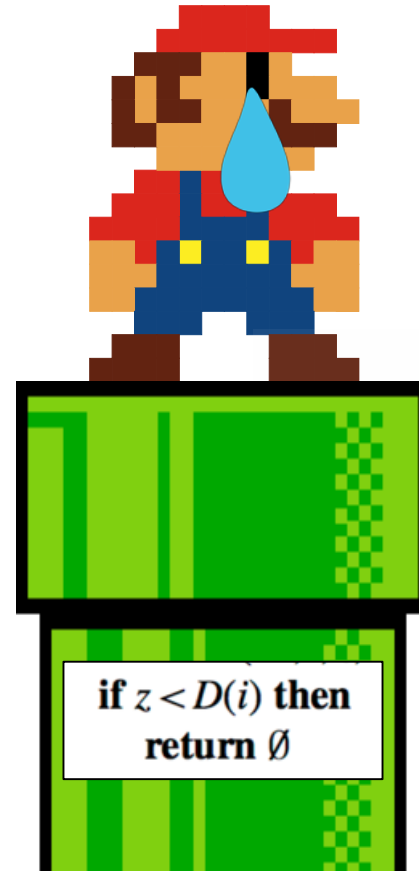


$Z = 2$

$D = [0, 1, 3]$



$Z = 2$   
 $D(i) = 3$   
 $2 < 3$   
**NO entramos!**



```
INEXRECUR(W, i, z, k, l)
  if  $z < D(i)$  then
    return  $\emptyset$ 
```

```
if  $z < D(i)$  then
  return  $\emptyset$ 
```

# Índice

1. Introducción



2. Materiales y métodos

3. Resultados

4. Conclusiones

5. Bibliografía



# Materiales y métodos

**CALCULATED( $W$ )**

$k \leftarrow 1$

$l \leftarrow |X| - 1$

$z \leftarrow 0$

**for**  $i=0$  **to**  $|W|-1$  **do**

$k \leftarrow C(W[i]) + O'(W[i], k-1) + 1$

$l \leftarrow C(W[i]) + O'(W[i], l)$

**if**  $k > l$  **then**

$k \leftarrow 1$

$l \leftarrow |X| - 1$

$z \leftarrow z + 1$

$D(i) \leftarrow z$

**INEXRECUR( $W, i, z, k, l$ )**

**if**  $z < D(i)$  **then**

**return**  $\emptyset$

**if**  $i < 0$  **then**

**return**  $\{[k, l]\}$

$I \leftarrow \emptyset$

$I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, l)$

**for each**  $b \in \{A, C, G, T\}$  **do**

$k \leftarrow C(b) + O(b, k-1) + 1$

$l \leftarrow C(b) + O(b, l)$

**if**  $k \leq l$  **then**

$I \leftarrow I \cup \text{INEXRECUR}(W, i, z-1, k, l)$

**if**  $b = W[i]$  **then**

$I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z, k, l)$

**else**

$I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, l)$

**return**  $I$



# Materiales y métodos

## Funciones implementadas

---

- **Script SuffixArray:** creación del SA a partir de la secuencia referencia X
- **Función C(a):** número de símbolos en  $X[0,n-2]$  que son lexicográficamente menores que a.
- **Función O(a,i):** número de ocurrencias de a en  $B[0,i]$
- **Función INEXRECUR:** posición del SA en el que se ha encontrada el alineamiento.
  - Archivo .txt con la traza

# Materiales y métodos

## Problema a resolver

---

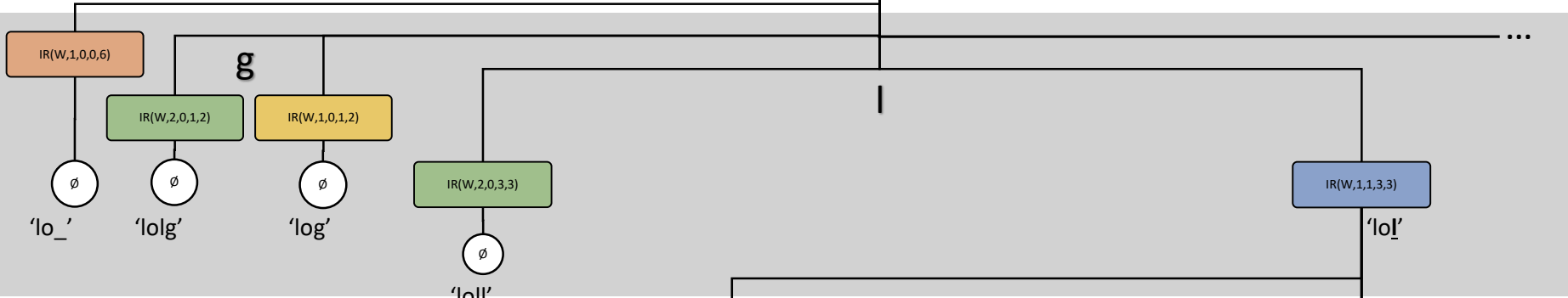


Dado  $X = \text{“googol\$”}$ , buscar los posibles alineamientos de  $W = \text{“lol”}$  con  $X$ , siendo:

$$Z = 1$$
$$D = (0,1,1)$$

INEXRECUR(W,2,1,0,6)

'lol'

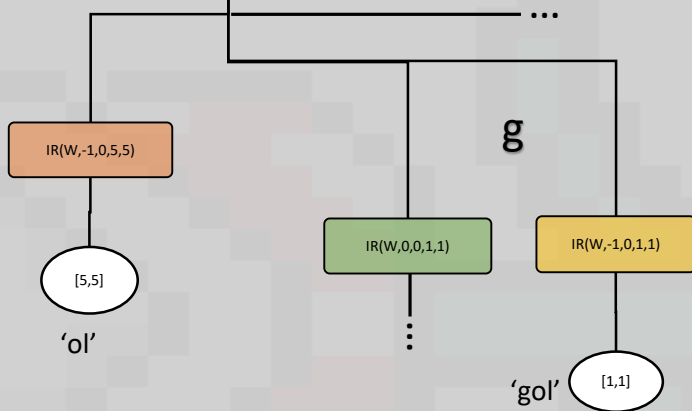
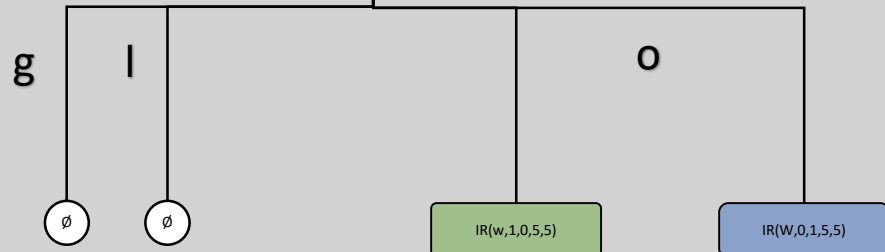
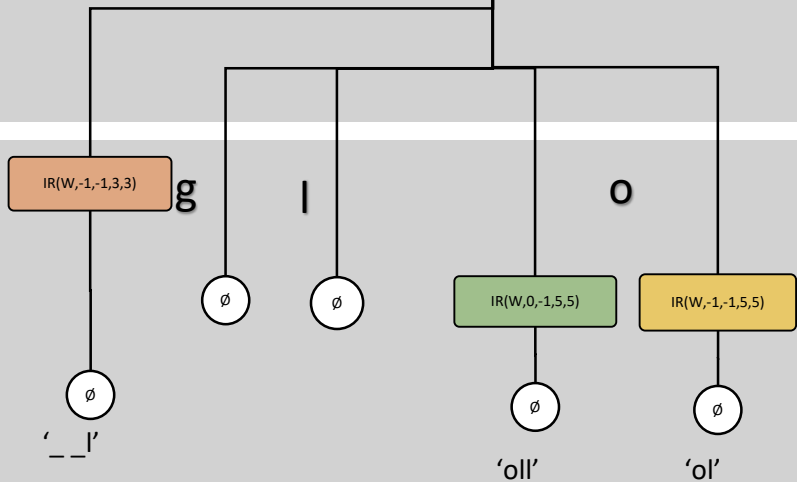


```

INEXRECUR(W,i,z,k,l)
  if z < D(i) then
    return ∅
  if i < 0 then
    return {[k,l]}
  I ← ∅
  I ← I ∪ INEXRECUR(W,i-1,z-1,k,l)  DELETE
  for each b ∈ {A,C,G,T} do
    k ← C(b) + O(b,k-1) + 1
    l ← C(b) + O(b,l)
    if k ≤ l then
      I ← I ∪ INEXRECUR(W,i,z-1,k,l)  INSERT
      if b = W[i] then
        I ← I ∪ INEXRECUR(W,i-1,z,k,l)  MATCH
      else
        I ← I ∪ INEXRECUR(W,i-1,z-1,k,l)  REPLACE
  return I
  
```

IR(W,0,0,3,3)

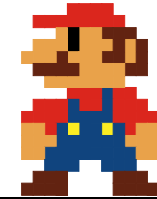
'l\_l'







$z < D(i)?$



INEXRECUR(W,2,1,0,6)

IR(W,1,0,0)



$z < D(i)?$

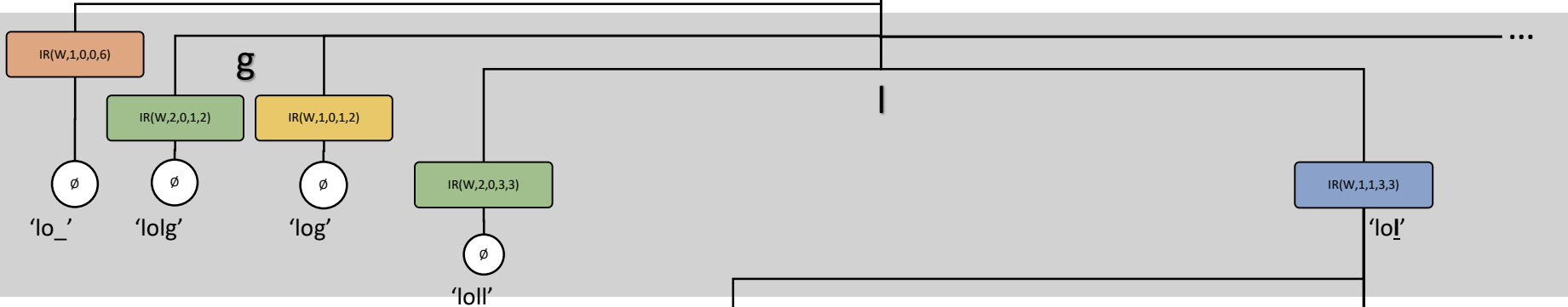
$\emptyset$

```
INEXRECUR(W, i, z, k, l)
  if  $z < D(i)$  then
    return  $\emptyset$ 
  if  $i < 0$  then
    return  $\{[k, l]\}$ 
   $I \leftarrow \emptyset$ 
   $I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, l)$  DELETE
  for each  $b \in \{A, C, G, T\}$  do
     $k \leftarrow C(b) + O(b, k-1) + 1$ 
     $l \leftarrow C(b) + O(b, l)$ 
    if  $k \leq l$  then
       $I \leftarrow I \cup \text{INEXRECUR}(W, i, z-1, k, l)$  INSERT
      if  $b = W[i]$  then
         $I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z, k, l)$  MATCH
      else
         $I \leftarrow I \cup \text{INEXRECUR}(W, i-1, z-1, k, l)$  REPLACE
  return  $I$ 
```



INEXRECUR(W,2,1,0,6)

'lol'



INEXRECUR(W,i,z,k,l)

if  $z < D(i)$  then

return  $\emptyset$

if  $i < 0$  then

return  $\{\{k,l\}\}$

$I \leftarrow \emptyset$

$I \leftarrow I \cup \text{INEXRECUR}(W,i-1,z-1,k,l)$  DELETE

for each  $b \in \{A, C, G, T\}$  do

$k \leftarrow C(b) + O(b, k-1) + 1$

$l \leftarrow C(b) + O(b, l)$

if  $k \leq l$  then

$I \leftarrow I \cup \text{INEXRECUR}(W,i,z-1,k,l)$  INSERT

if  $b = W[i]$  then

$I \leftarrow I \cup \text{INEXRECUR}(W,i-1,z,k,l)$  MATCH

else

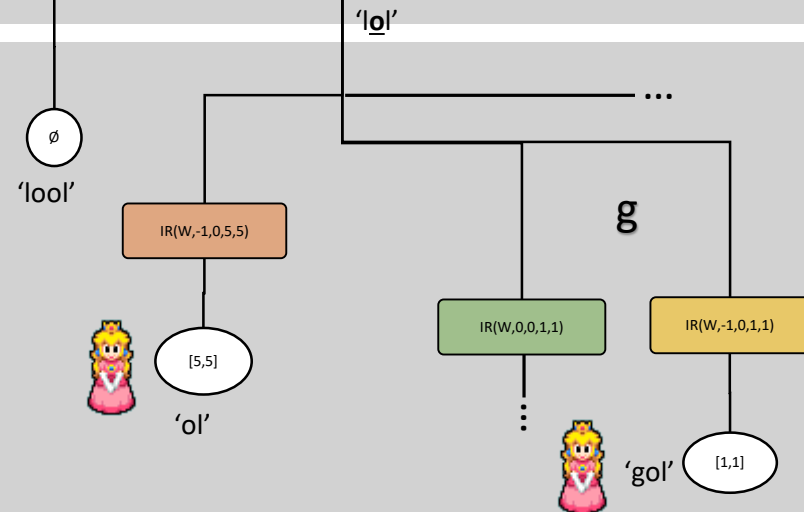
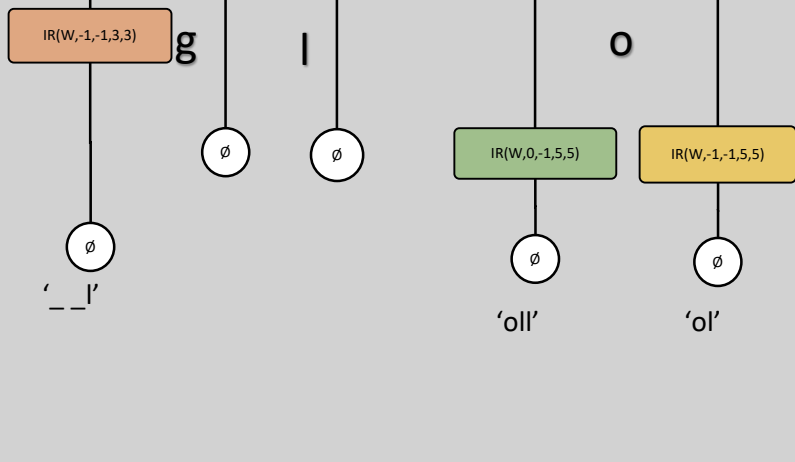
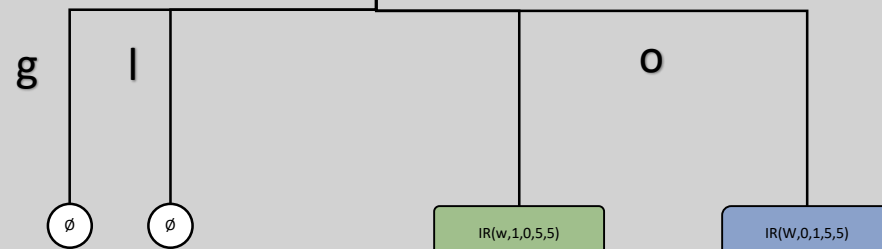
$I \leftarrow I \cup \text{INEXRECUR}(W,i-1,z-1,k,l)$  REPLACE

return  $I$


g

l

o



# Índice

1. Introducción
2. Materiales y métodos
-  3. Resultados
4. Conclusiones
5. Bibliografía



# Resultados

## Ejemplo de traza obtenida:

1	-----									
2		INEXRECUR - by XAVI GABRI AITANA ALFREDO								
3	-----									
4										
5	-	D	[ 1 ]		1	0	0	6		
6	-	I	[ g ]		2	0	1	2		
7	-	S	[ g -> 1 ]		1	0	1	2		
8	-	I	[ 1 ]		2	0	3	3		
9	-	M	[ 1 ]		1	1	3	3		
10	-	-	D	[ o ]		0	0	3	3	
11	-	-	-	D	[ 1 ]		-1	-1	3	3
12	-	-	-	I	[ o ]		0	-1	5	5
13	-	-	-	S	[ o -> 1 ]		-1	-1	5	5
14	-	-	I	[ o ]		1	0	5	5	
15	-	-	M	[ o ]		0	1	5	5	
16	-	-	-	D	[ 1 ]		-1	0	5	5
17	-----> [ c(5, 5) ]									
18	-	-	-	I	[ g ]		0	0	1	1
19	-	-	-	-	D	[ 1 ]		-1	-1	1
20	-	-	-	-	I	[ o ]		0	-1	4
21	-	-	-	-	S	[ o -> 1 ]		-1	-1	4
22	-	-	-	S	[ g -> 1 ]		-1	0	1	1
23	-----> [ c(1, 1) ]									
24	-	I	[ o ]		2	0	4	6		
25	-	S	[ o -> 1 ]		1	0	4	6		

*Equivale al camino recorrido hasta llegar a la solución*




# Resultados

## Otras búsquedas:

1	-----									
2		INEXRECUR - by XAVI GABRI AITANA ALFREDO								
3	-----									
4										
5	-	D	[ A ]	1	0	0	6			
6	-	I	[ A ]	2	0	1	3			
7	-	M	[ A ]	1	1	1	3			
8	-	-	D [ N ]		0	0	1	3		
9	-	-	- D [ A ]			-1	-1	1	3	
10	-	-	- I [ B ]			0	-1	4	4	
11	-	-	- S [ B -> A ]			-1	-1	4	4	
12	-	-	- I [ N ]			0	-1	5	6	
13	-	-	- S [ N -> A ]			-1	-1	5	6	
14	-	-	- I [ B ]			1	0	4	4	
15	-	-	- S [ B -> N ]			0	0	4	4	
16	-	-	- D [ A ]				-1	-1	4	4
17	-	-	- I [ N ]			1	0	5	6	
18	-	-	- M [ N ]			0	1	5	6	
19	-	-	- D [ A ]				-1	0	5	6
20	-----> [ c(5, 6) ]									
21	-	-	- I [ A ]			0	0	2	3	
22	-	-	- - D [ A ]				-1	-1	2	3
23	-	-	- - I [ B ]				0	-1	4	4
24	-	-	- - S [ B -> A ]				-1	-1	4	4
25	-	-	- - I [ N ]				0	-1	6	6
26	-	-	- - S [ N -> A ]				-1	-1	6	6
27	-	-	- M [ A ]				-1	1	2	3
28	-----> [ c(2, 3) ]									
29	-	I	[ B ]	2	0	4	4			
30	-	S	[ B -> A ]	1	0	4	4			
31	-	I	[ N ]	2	0	5	6			
32	-	S	[ N -> A ]	1	0	5	6			

**X** = “BANANA\$”

**W** = “ANA”

**Z** = 1 

**D** = (0,1,1)

0	\$BANANA
1	A\$BANAN
2	ANA\$BAN
3	ANANA\$B
4	BANANA\$
5	NA\$BANA
6	NANA\$BA


**SA**

# Resultados

## Otras búsquedas:

**X** = “TTGAAGAAGCAGGCTGCCATGTTGCAAGCTGCCTCATGGAGGGGATCAGCTGCGAGGAGCTAAGAGCCCC\$”


$$W = \text{“GATA”}$$

**Z** = 1 

$$\mathbf{D} = (0, 1, 1, 1)$$

1	-----									
2	INEXRECUR - by XAVI GABRI AITANA ALFREDO									
3	-----									
4										
5	-	D	[ A ]	2	0	0	70			
6	-	I	[ A ]	3	0	1	17			
7	-	M	[ A ]	2	1	1	17			
8	-	-	D	[ T ]	1	0	1	17		
9	-	-	I	[ A ]	2	0	1	4		
10	-	-	S	[ A -> T ]	1	0	1	4		
11	-	-	I	[ C ]	2	0	19	23		
12	-	-	S	[ C -> T ]	1	0	19	23		
13	-	-	I	[ G ]	2	0	35	41		
14	-	-	S	[ G -> T ]	1	0	35	41		
15	-	-	I	[ T ]	2	0	59	59		
16	-	-	M	[ T ]	1	1	59	59		
17	-	-	-	D	[ A ]	0	0	59	59	
18	-	-	-	-	D	[ G ]	-1	-1	59	59
19	-	-	-	-	I	[ C ]	0	-1	30	30
20	-	-	-	-	S	[ C -> G ]	-1	-1	30	30
21	-	-	-	-	I	[ C ]	1	0	30	30
22	-	-	-	-	S	[ C -> A ]	0	0	30	30
23	-	-	-	-	D	[ G ]	-1	-1	30	30
24	-	-	-	-	I	[ G ]	0	-1	48	48
25	-	-	-	-	M	[ G ]	-1	0	48	48
26	-----> [ c(48, 48) ]									
27	-	I	[ C ]	3	0	18	34			
28	-	S	[ C -> A ]	2	0	18	34			
29	-	I	[ G ]	3	0	35	58			
30	-	S	[ G -> A ]	2	0	35	58			
31	-	I	[ T ]	3	0	59	70			
32	-	S	[ T -> A ]	2	0	59	70			

# Índice

1. Introducción
2. Materiales y métodos
3. Resultados
-  4. Conclusiones
5. Bibliografía



# Conclusiones

- Se ha conseguido implementar de manera exitosa el algoritmo recursivo InexRecur, mediante el cual conseguimos alinear una secuencia corta dentro de otra de mayor longitud, con permisividad a huecos y errores.
- Se podría implementar una aproximación a una *matriz de puntuación* según el valor en que disminuye  $z$  en cada una de las llamadas a la función.
- Como líneas futuras, se podría trabajar en la obtención de la función  $D$ , que en este caso no ha sido necesario calcular dado que disponíamos de la misma como dato de entrada.



# Índice

1. Introducción
2. Materiales y métodos
3. Resultados
4. Conclusiones



Bibliografía



# Bibliografía

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14), pp.1754-1760.



**JOSE GABRIEL GARCIA PARDO**

**ALFREDO TORREGROSA LLORET**

**BWA**



# **INEX RECUA**

**JAVIER DE LA TORRE COSTA**

**AITANA PASCUAL BELDA**

