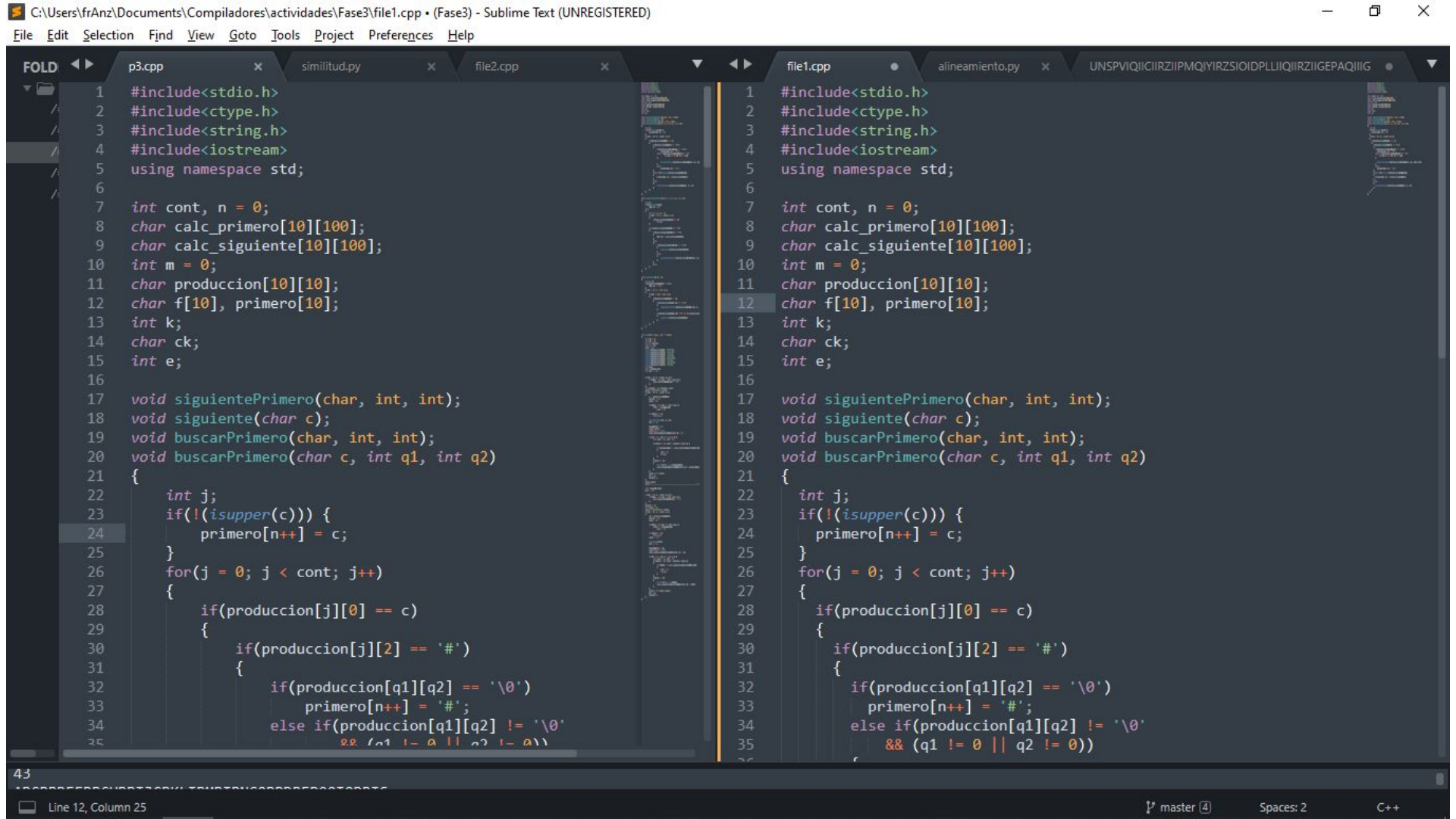
# Detección de plagio: Similitud de códigos c++

Johann F. Huaypuna Huanca


C:\Users\frAnz\Documents\Compiladores\actividades\Fase3\file1.cpp • (Fase3) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLD
p3.cpp
similitud.py
file2.cpp
file1.cpp
alineamiento.py
UNSPVIQIICIIRZIIPMQIYIRZSIOIDPLLIIQIIRZIIGEPAQIIIG
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<iostream>
using namespace std;
int cont, n = 0;
char calc_primero[10][100];
char calc_siguiente[10][100];
int m = 0;
char produccion[10][10];
char f[10], primero[10];
int k;
char ck;
int e;
void siguientePrimero(char, int, int);
void siguiente(char c);
void buscarPrimero(char, int, int);
void buscarPrimero(char c, int q1, int q2)
{
int j;
if(!(isupper(c))) {
primero[n++] = c;
}
for(j = 0; j < cont; j++)
{
if(produccion[j][0] == c)
{
if(produccion[j][2] == '#')
{
if(produccion[q1][q2] == '\0')
primero[n++] = '#';
else if(produccion[q1][q2] != '\0'
&& (q1 != 0 || q2 != 0))
43
Line 12, Column 25
master 4
Spaces: 2
C++

C:\Users\frAnz\Documents\Compiladores\actividades\Fase3\file1.cpp • (Fase3) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLD

p3.cpp similitud.py file2.cpp

```cpp
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<iostream>
using namespace std;

int contador, z = 0;
char prims[10][100];
char next[10][100];
int l = 0;
char prod[10][10];
char f[10], prim[10];
int k;
char ck;
int e;

void siguienteprim(char, int, int);
void siguiente(char letra);
void buscarprim(char, int, int);
void buscarprim(char letra, int s1, int s2)
{
    int Fj;
    if(!(isupper(c))) {
        prim[n++] = c;
    }
    for(Fj = 0; Fj < contador; Fj++)
    {
        if(prod[j][0] == c)
        {
            if(prod[j][2] == '#')
            {
                if(prod[q1][q2] == '\0')
                    prim[n++] = '#';
                else if(prod[q1][q2] != '\0'
                        && (q1 != 0 || q2 != 0))
                {
```

file1.cpp alineamiento.py UNSPVIQIICIIRZIIPMQIYIRZSIOIDPLLIIQIIRZIIGEPAQIIIG

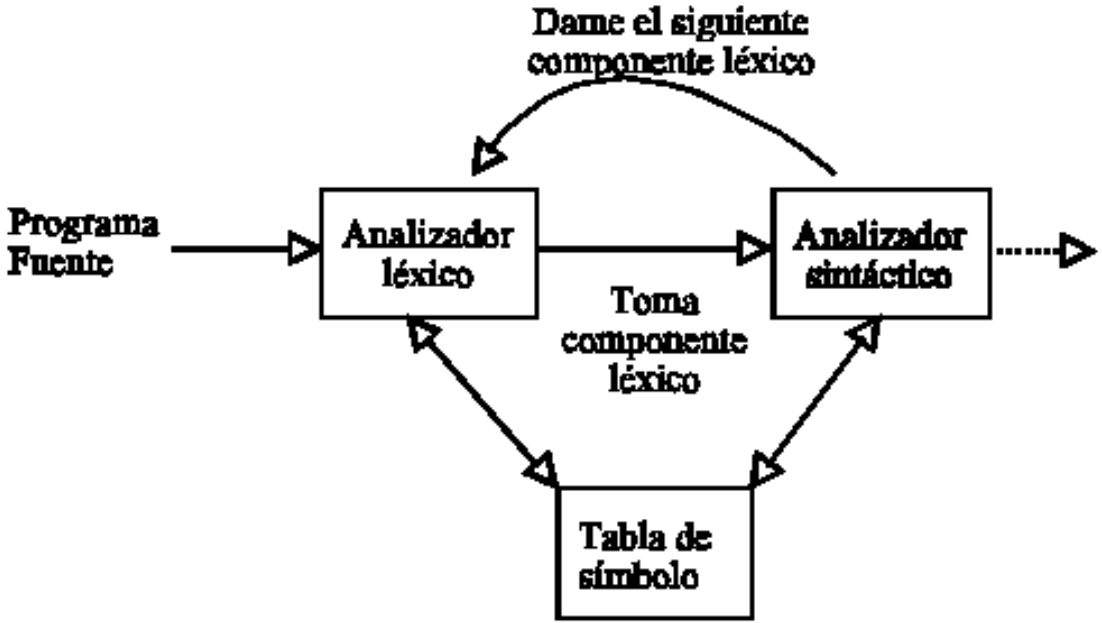
```cpp
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<iostream>
using namespace std;

int cont, n = 0;
char calc_primero[10][100];
char calc_siguiente[10][100];
int m = 0;
char produccion[10][10];
char f[10], primero[10];
int k;
char ck;
int e;

void siguientePrimero(char, int, int);
void siguiente(char c);
void buscarPrimero(char, int, int);
void buscarPrimero(char c, int q1, int q2)
{
  int j;
  if(!(isupper(c))) {
    primero[n++] = c;
  }
  for(j = 0; j < cont; j++)
  {
    if(produccion[j][0] == c)
    {
      if(produccion[j][2] == '#')
      {
        if(produccion[q1][q2] == '\0')
          primero[n++] = '#';
        else if(produccion[q1][q2] != '\0'
            && (q1 != 0 || q2 != 0))
        {
```

Line 13, Column 7 master 4 Spaces: 2 C++

# Análisis léxico



# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |
| 1 | ASIGNACIÓN | `=` | 10 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |
| 1 | ASIGNACIÓN | `=` | 10 |
| 1 | VARIABLE | `SubTotal` | 12 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |
| 1 | ASIGNACIÓN | `=` | 10 |
| 1 | VARIABLE | `SubTotal` | 12 |
| 1 | OP_PRODUCTO | `*` | 21 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |
| 1 | ASIGNACIÓN | `=` | 10 |
| 1 | VARIABLE | `SubTotal` | 12 |
| 1 | OP_PRODUCTO | `*` | 21 |
| 1 | VARIABLE | `Precio` | 23 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |
| 1 | ASIGNACIÓN | `=` | 10 |
| 1 | VARIABLE | `SubTotal` | 12 |
| 1 | OP_PRODUCTO | `*` | 21 |
| 1 | VARIABLE | `Precio` | 23 |
| 1 | OP_DIVIDE | `/` | 31 |

# Análisis léxico

```
int Impuesto = SubTotal * Precio / 100
```

| LINEA | TIPO | VALOR | POSICIÓN |
|---|---|---|---|
| 1 | ENTERO | `int` | 1 |
| 1 | VARIABLE | `Impuesto` | 5 |
| 1 | ASIGNACIÓN | `=` | 10 |
| 1 | VARIABLE | `SubTotal` | 12 |
| 1 | OP_PRODUCTO | `*` | 21 |
| 1 | VARIABLE | `Precio` | 23 |
| 1 | OP_DIVIDE | `/` | 31 |
| 1 | NUMERO | `100` | 33 |

## Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

**SECUENCIAS A COMPARAR**

Secuencia $j$: `AAG`
Secuencia $i$: `AGC`

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

**SECUENCIAS A COMPARAR**

Secuencia $j$: AAG

Secuencia $i$: AGC

**PARÁMETROS**

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

$\gamma = -5$

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

**SECUENCIAS A COMPARAR**

Secuencia $j$: AAG

Secuencia $i$: AGC

**PARÁMETROS**

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

$\gamma = -5$

**DEFINICIÓN DEL PUNTAJE**

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) \\ S(i-1,j) + \gamma \\ S(i,j-1) + \gamma \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

**SECUENCIAS A COMPARAR**

Secuencia $j$: AAG
Secuencia $i$: AGC


**PARÁMETROS**

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

$\gamma = -5$

**DEFINICIÓN DEL PUNTAJE**

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) & \begin{matrix}\text{A}\\\text{A}\end{matrix} \\ S(i-1,j) + \gamma & \begin{matrix}\text{A–}\\\text{A}\end{matrix} \\ S(i,j-1) + \gamma & \begin{matrix}\text{A}\\\text{A–}\end{matrix} \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

## SECUENCIAS A COMPARAR

Secuencia $j$: AAG
Secuencia $i$: AGC


## PARÁMETROS

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

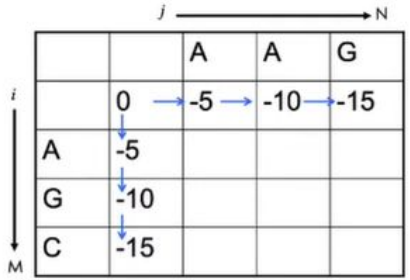
$\gamma = -5$

## DEFINICIÓN DEL PUNTAJE

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) & \begin{matrix} A \\ A \end{matrix} \\ S(i-1,j) + \gamma & \begin{matrix} A- \\ A \end{matrix} \\ S(i,j-1) + \gamma & \begin{matrix} A \\ A- \end{matrix} \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

## MATRIZ DE PROGRAMACIÓN DINÁMICA




Inicialización

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

## SECUENCIAS A COMPARAR

Secuencia $j$: AAG
Secuencia $i$: AGC


## PARÁMETROS

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

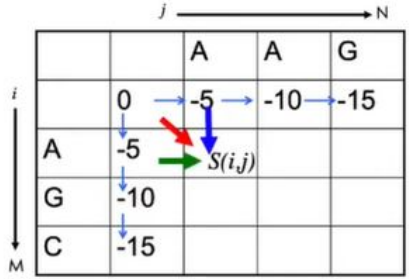
$\gamma = -5$

## DEFINICIÓN DEL PUNTAJE

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) & \text{A / A} \\ S(i-1,j) + \gamma & \text{A- / A} \\ S(i,j-1) + \gamma & \text{A / A-} \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

## MATRIZ DE PROGRAMACIÓN DINÁMICA




Inicialización

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

**SECUENCIAS A COMPARAR**

Secuencia $j$: AAG
Secuencia $i$: AGC


**PARÁMETROS**

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

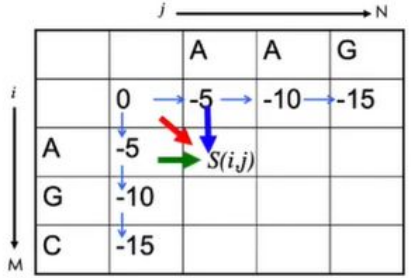
$\gamma = -5$

**DEFINICIÓN DEL PUNTAJE**

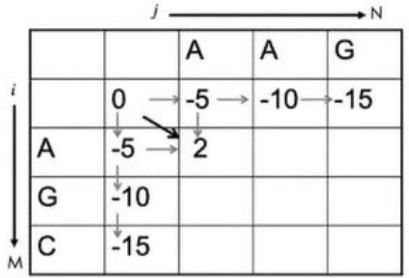
$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) & \begin{matrix}\text{A}\\\text{A}\end{matrix} \\ S(i-1,j) + \gamma & \begin{matrix}\text{A–}\\\text{A}\end{matrix} \\ S(i,j-1) + \gamma & \begin{matrix}\text{A}\\\text{A–}\end{matrix} \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

**MATRIZ DE PROGRAMACIÓN DINÁMICA**




Inicialización




$$S(i-1,j-1) + s(x_i, y_j) = 0 + 2 = 2$$
$$S(i-1,j) + \gamma = -5 + (-5) = -10$$
$$S(i,j-1) + \gamma = -5 + (-5) = -10$$

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

## SECUENCIAS A COMPARAR

Secuencia $j$: AAG
Secuencia $i$: AGC


## PARÁMETROS

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

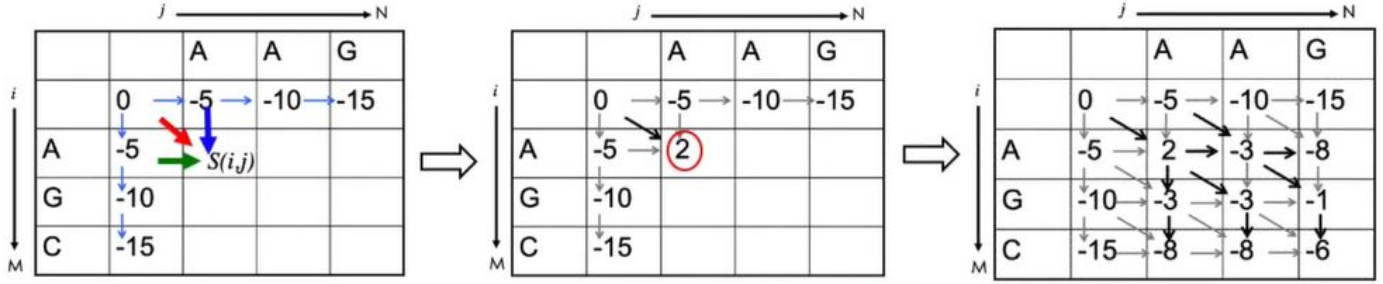
$\gamma = -5$

## DEFINICIÓN DEL PUNTAJE

$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) & \text{A / A} \\ S(i-1,j) + \gamma & \text{A– / A} \\ S(i,j-1) + \gamma & \text{A / A–} \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

## MATRIZ DE PROGRAMACIÓN DINÁMICA




Inicialización

$S(i-1,j-1) + s(x_i, y_j) = 0 + 2 = 2$

$S(i-1,j) + \gamma = -5 + (-5) = -10$

$S(i,j-1) + \gamma = -5 + (-5) = -10$

# Métodos de programación dinámica (PD) para la comparación de secuencias

*Alineamiento Global. Algoritmo Needleman-Wunsch*

## SECUENCIAS A COMPARAR

Secuencia $j$: AAG
Secuencia $i$: AGC


## PARÁMETROS

Matriz de sustitución $S(i,j)$

| | A | C | G | T |
|---|---|---|---|---|
| A | **2** | -7 | -5 | -7 |
| C | -7 | **2** | -7 | -5 |
| *G* | -5 | -7 | **2** | -7 |
| T | -7 | -5 | -7 | **2** |

*Penalización por "gap"*

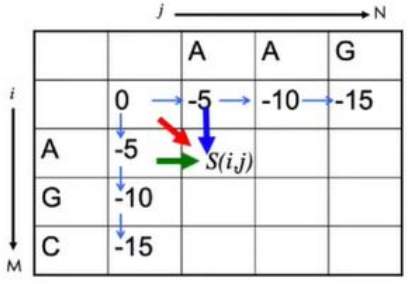
$\gamma = -5$

## DEFINICIÓN DEL PUNTAJE

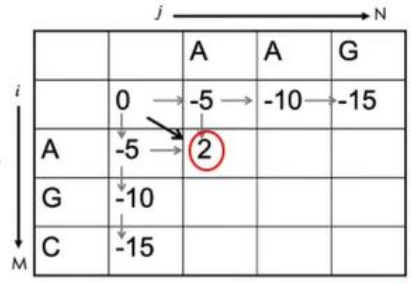
$$S(i,j) = \max \begin{cases} S(i-1,j-1) + s(x_i, y_j) & \text{A / A} \\ S(i-1,j) + \gamma & \text{A– / A} \\ S(i,j-1) + \gamma & \text{A / A–} \end{cases}$$

Escojer el máximo valor de puntaje $S(i,j)$

## MATRIZ DE PROGRAMACIÓN DINÁMICA




Inicialización

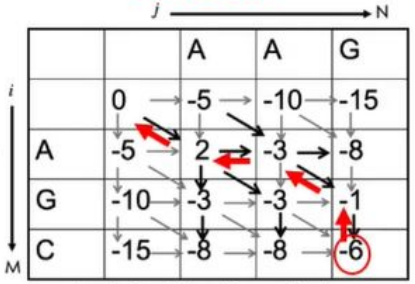



$S(i-1,j-1) + s(x_i, y_j) = 0 + 2 = 2$

$S(i-1,j) + \gamma = -5 + (-5) = -10$

$S(i,j-1) + \gamma = -5 + (-5) = -10$

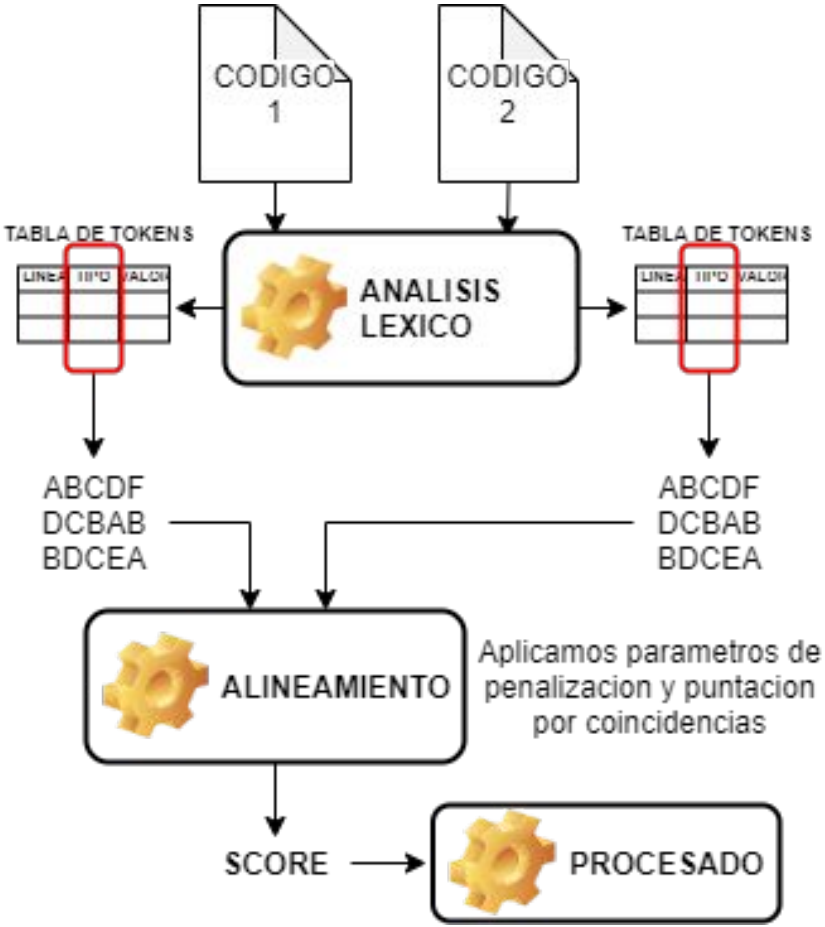
*"Traceback"*




Puntaje del alineamiento óptimo $S(N,M)$

| | |
|---|---|
| Alineamiento | A A G – |
| Resultante | A – G C |

# Propuesta para abordar el problema



GRACIAS!