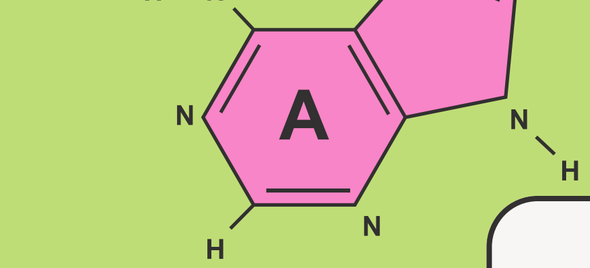
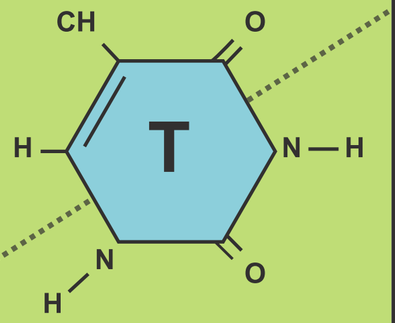
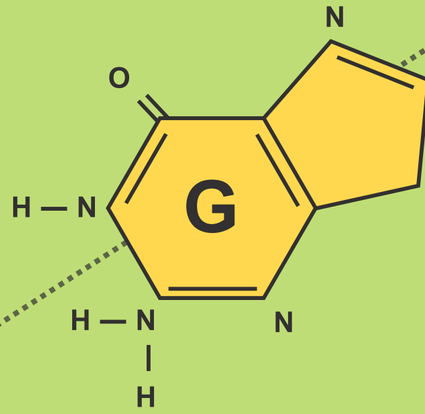




APROXIMACIONES CONCURRENTES A SMITH-WATERMAN

Hugo Salas Calderón
Programación Avanzada en Bioinformática

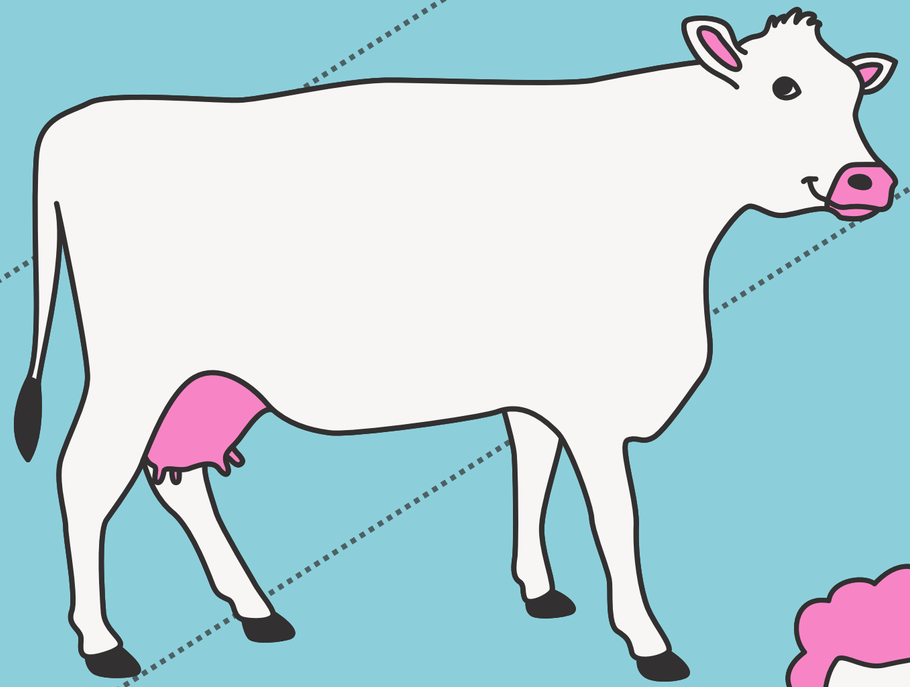


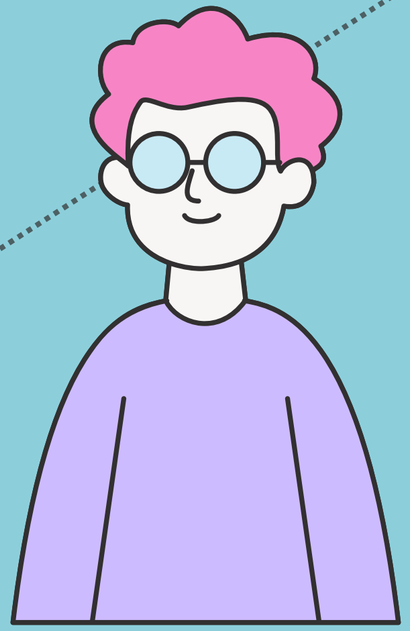
SELECCIÓN NATURAL

¿Por qué no?

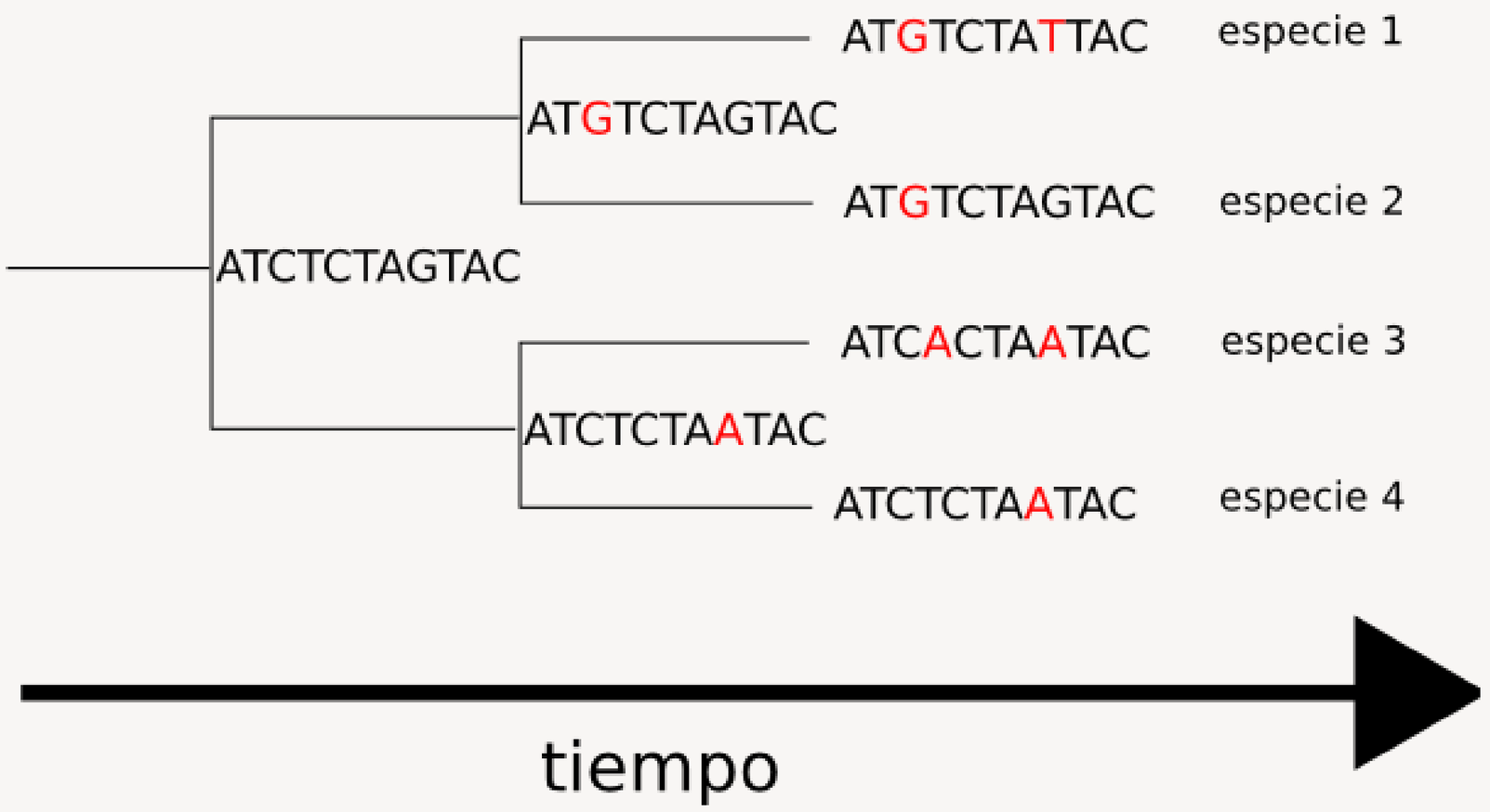
Pelo Rosa + Afro + Gafas = ↓ Supervivencia

Vaca sin manchas → Selección Artificial





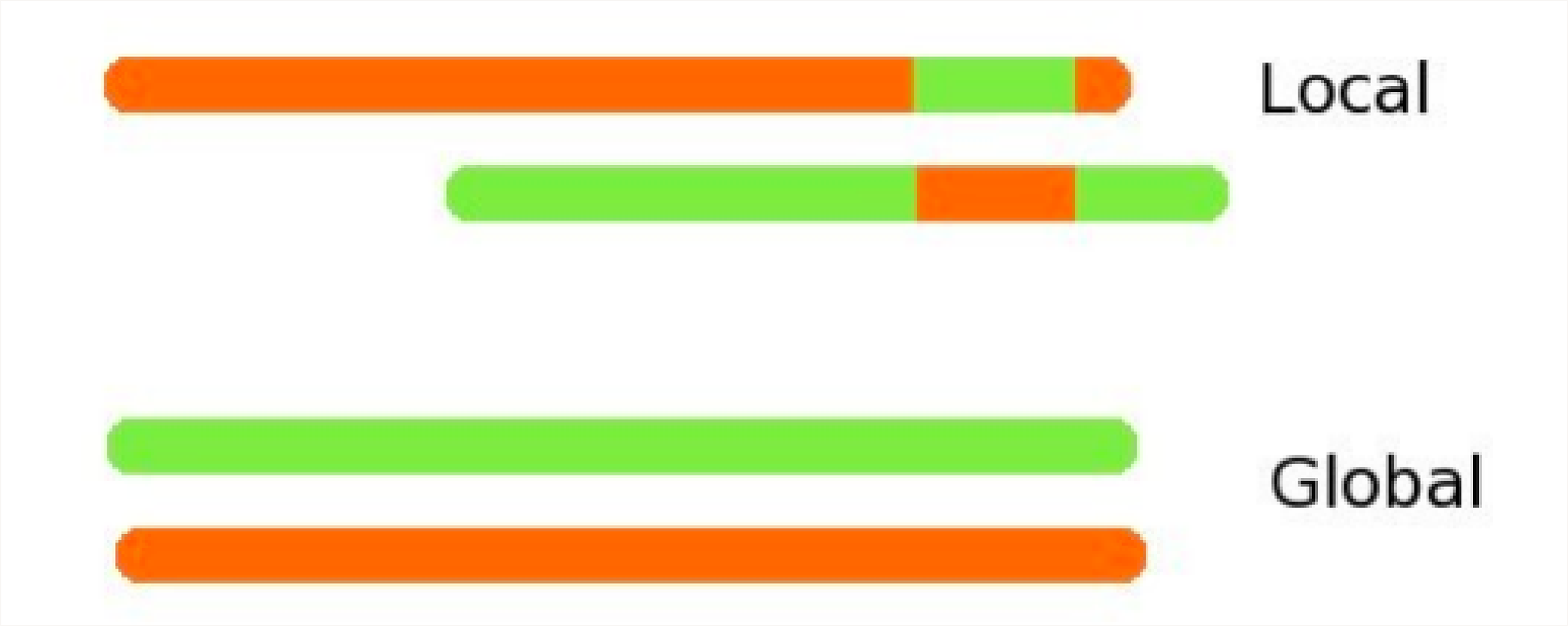
MUTACIÓN GENÉTICA



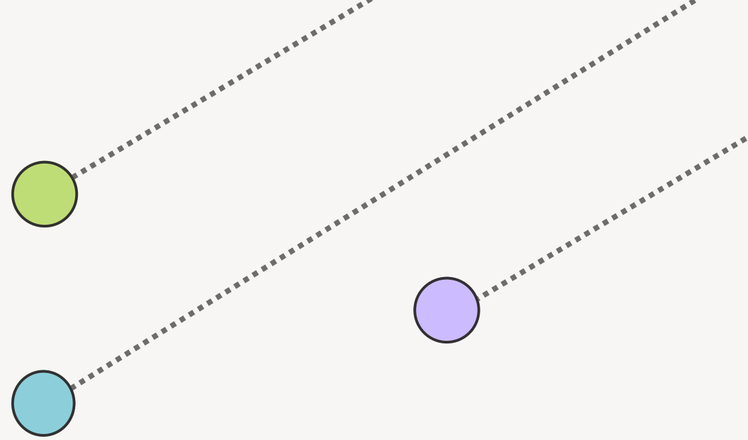
ALINEAMIENTO LOCAL DE SECUENCIAS

Temple F. Smith

Michael S. Waterman


Local
Global

COMPONENTES

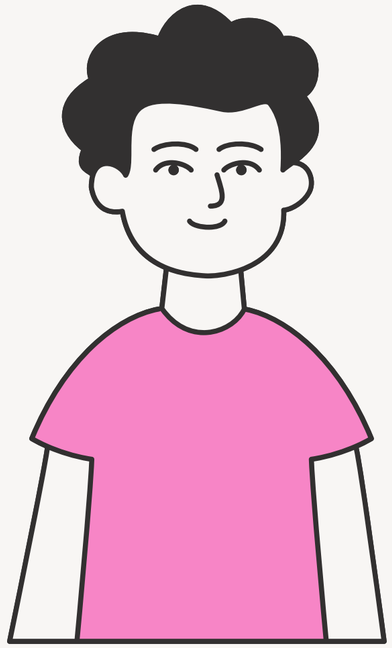


Secuencias

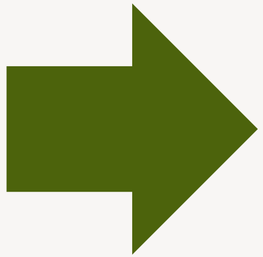
Matriz de Similitud

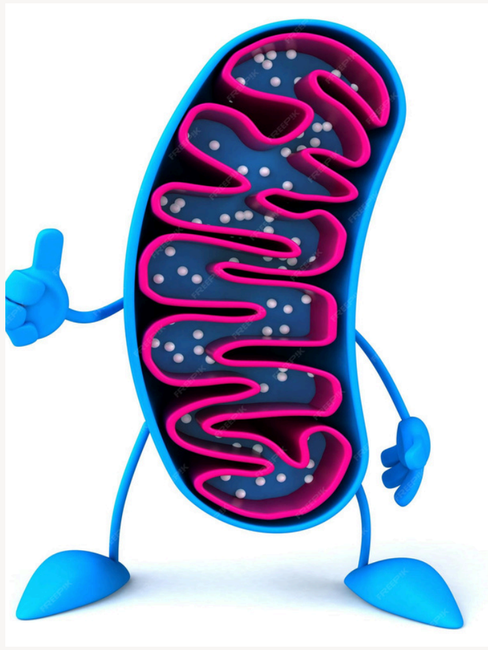
Función de Puntuación

SECUENCIAS



VS





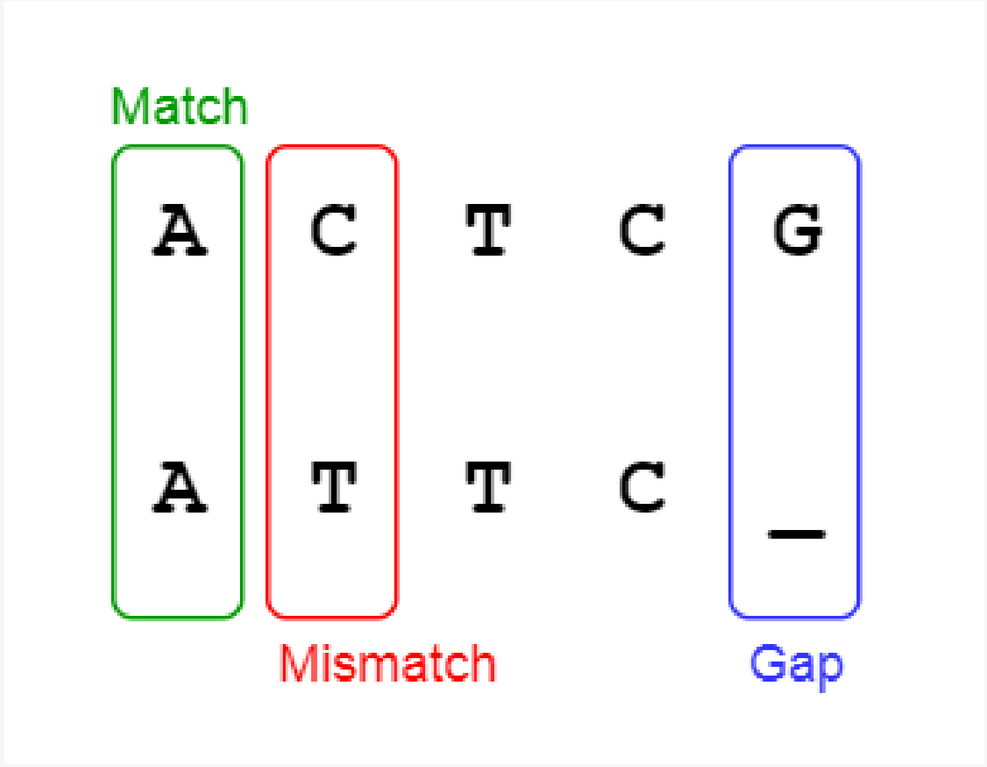
Jorge

Jorge el Curioso

Genoma Mitocondrial

~ 16kB

MATRIZ DE SIMILITUD



Match = +3

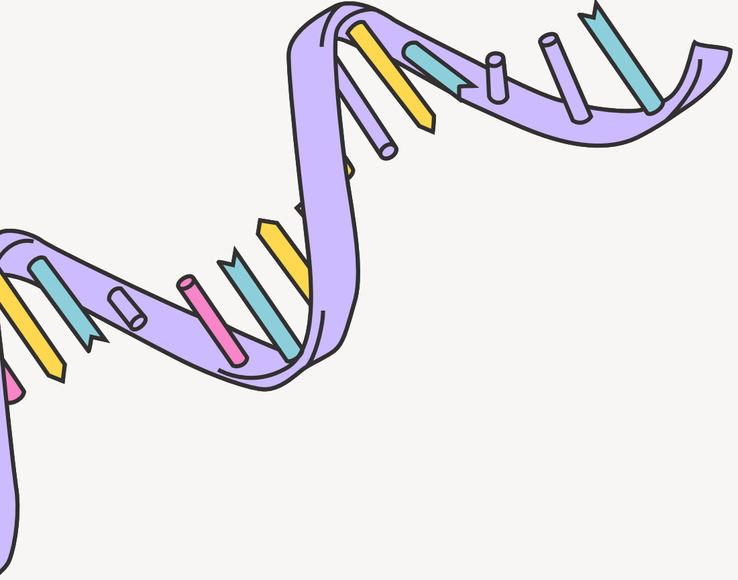
Mismatch = –1

Gap = –2

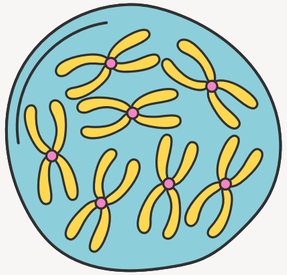
Score = +3 – 1 + 3 + 3 –2 = 6

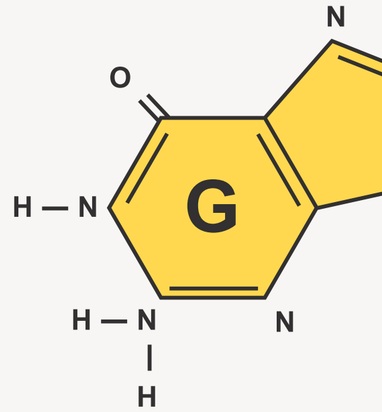
FUNCIÓN DE PUNTUACIÓN

$$M[i,j] = \text{máx} \begin{cases} 0 & \\ M[i-1,j-1] + s\big(A[i],B[j]\big), & (match) \\ M[i-1,j] + g, & (gap\ en\ B) \\ M[i,j-1] + g, & (gap\ en\ A) \end{cases} \quad \text{para } 1 \leq i \leq m,\ 1 \leq j \leq n$$



TRACEBACK





ALINEACIÓN

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	↖ 5	1	0	0	0	0	0	5	1	0	0
G	0	5	↖ 2	0	0	0	0	0	5	2	0	0
A	0	1	10	↖ 7	← 3	0	0	5	1	2	0	5
T	0	0	6	7	↖ 12	↖ ← 8	4	1	2	6	7	3
C	0	0	2	3	8	9	↖ 13	9	5	2	3	4
G	0	5	1	0	4	5	↑ 9	10	14	10	6	2
A	0	1	10	6	2	1	5	↖ 14	10	11	7	11

↖	↖	↖	↖	←	↖	↑	↖
5	2	7	12	8	13	9	14
G	G	A	T	–	C	G	A
G	A	A	T	T	C	–	A

Score = +3 -1 +3 +3 -2 +3 -2 +3 = 10

IMPLEMENTACIÓN SECUENCIAL

ENTORNO: C

1. Leer las secuencias FASTA desde stdin
2. Reservar espacio en memoria para las matrices
3. Iterar sobre la longitud de las secuencias
4. Calcular valor de similitud, luego puntuación para diag, up, left → max(0, diag, up, left)
5. Realizar traceback

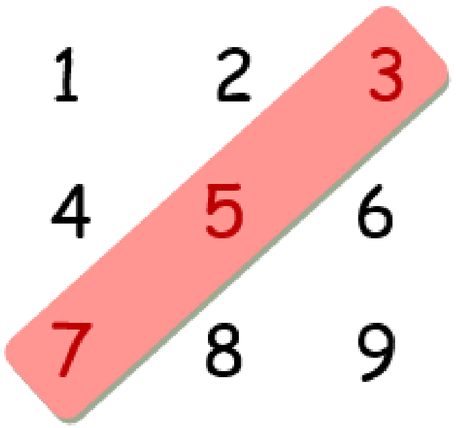
```
for (int i = 1; i <= m; i++) {
    for (int j = 1; j <= n; j++) {
        int s = 0;
        int ind1 = caracter(secuencia1[i - 1]);
        int ind2 = caracter(secuencia2[j - 1]);
        if (ind1 >= 0 && ind2 >= 0)
            s = similitud[ind1][ind2];

        int diag = H[(i - 1) * (n + 1) + (j - 1)] + s;
        int up = H[(i - 1) * (n + 1) + j] + GAP;
        int left = H[i * (n + 1) + (j - 1)] + GAP;
        int val = max4(0, diag, up, left);
        H[i * (n + 1) + j] = val;
```

IMPLEMENTACIÓN CONCURRENTE

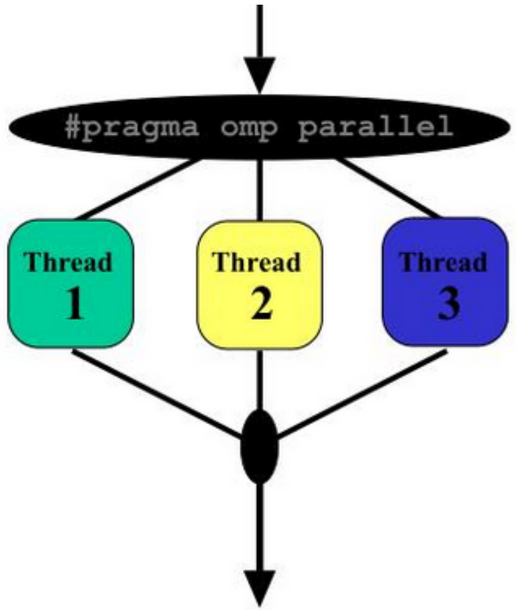
Procesamiento Diagonal Wavefront

1

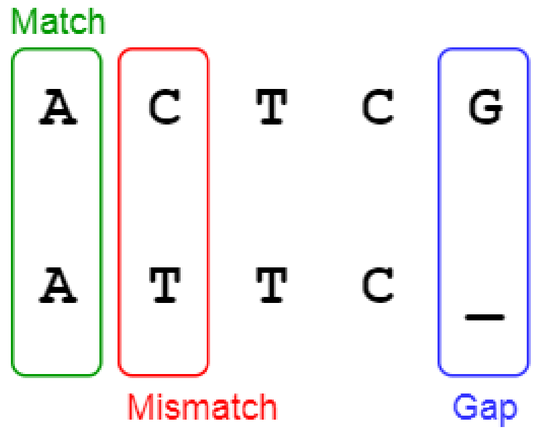


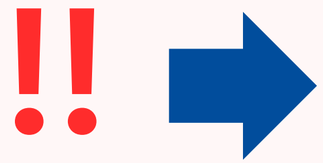
ANTIDIAGONAL [i+j]

2



3





```
for (int diag = 2; diag <= m + n; diag++) {
#pragma omp parallel for reduction(max:max_score)
for (int i = 1; i <= m; i++) {
    int j = diag - i;
    if (j >= 1 && j <= n) {
        int s = 0;
        int ind1 = caracter(seq1[i - 1]);
        int ind2 = caracter(seq2[j - 1]);
        if (ind1 >= 0 && ind2 >= 0) {
            s = similitud[ind1][ind2];
        }

        int diag = H[(i - 1) * (n + 1) + (j - 1)] + s;
        int up = H[(i - 1) * (n + 1) + j] + GAP;
        int left = H[i * (n + 1) + (j - 1)] + GAP;
        int val = max4(0, diag, up, left);
        H[i * (n + 1) + j] = val;
```

CUDA

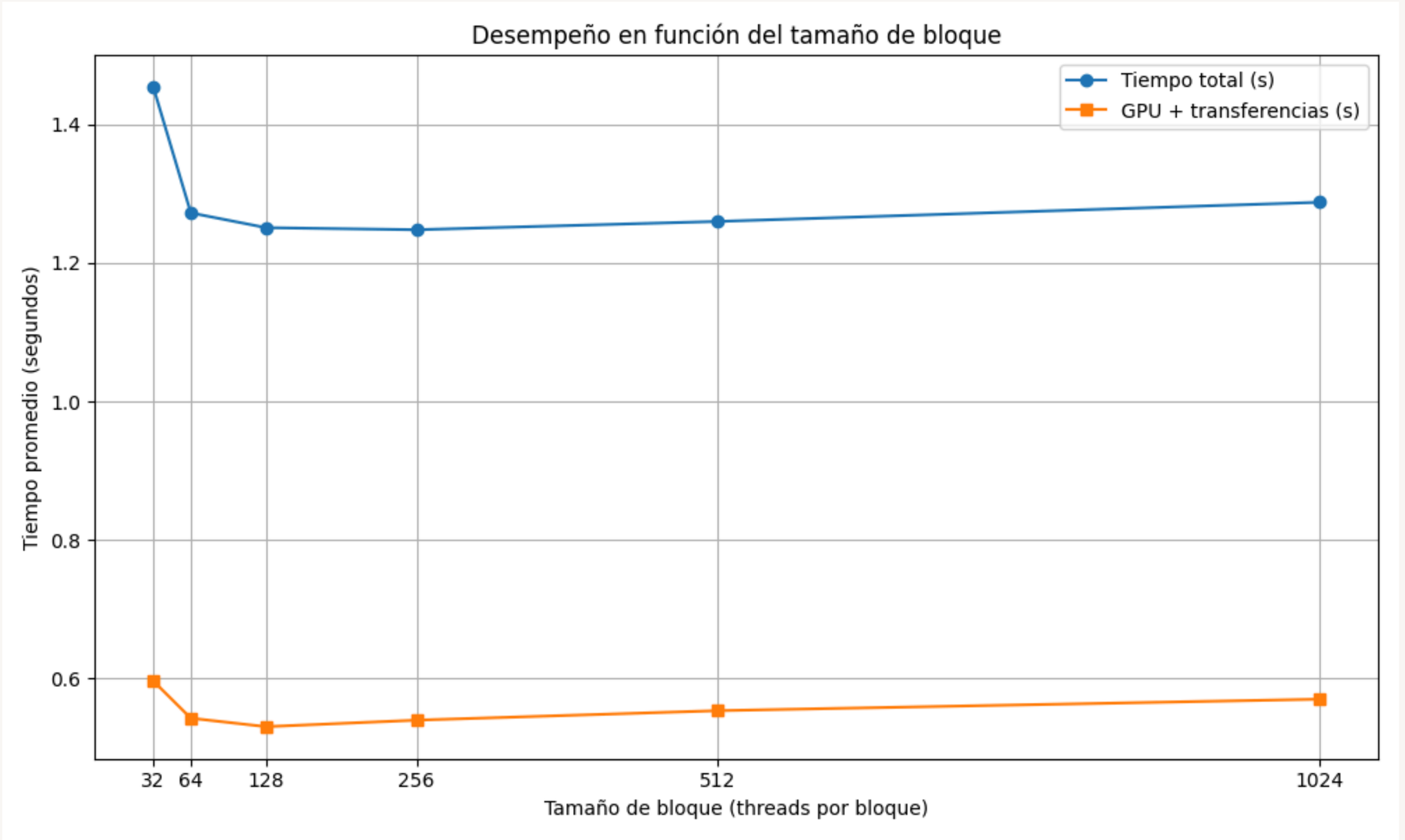
Compute Unified Device Architecture

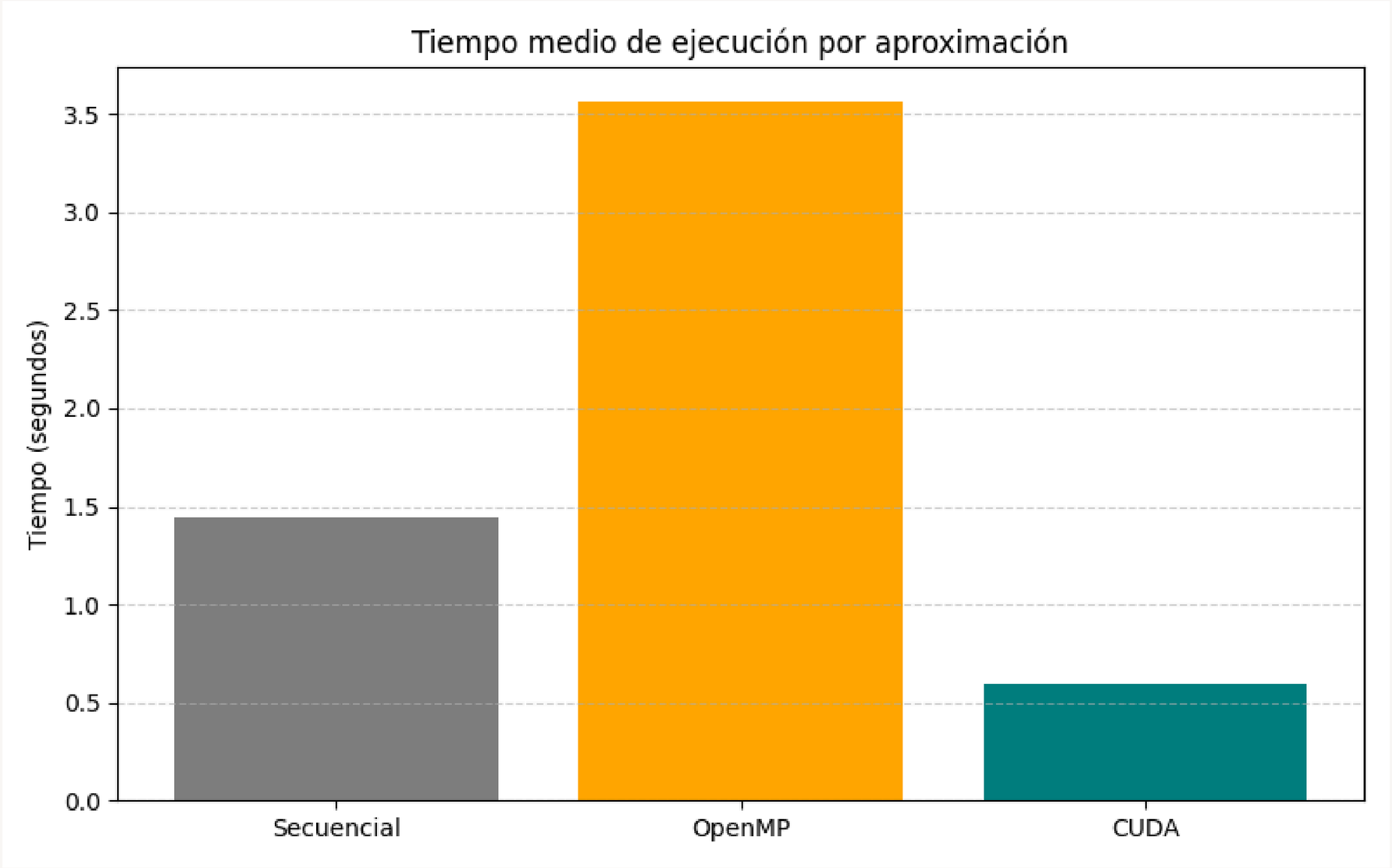
DEVICE

HOST

KERNEL

```cpp
// Kernel CUDA: calcular diagonal diag de la matriz H
__global__ void smith_waterman_kernel(
    char *seq1, char *seq2, int *H, int m, int n, int diag, int start_i, int end_i)
{
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    int i = start_i + idx;
    if (i <= end_i) {
        int j = diag - i + 1;
        int ind1 = caracter(seq1[i - 1]);
        int ind2 = caracter(seq2[j - 1]);
        int s = (ind1 >= 0 && ind2 >= 0) ? d_similitud[ind1 * 4 + ind2] : 0;
```

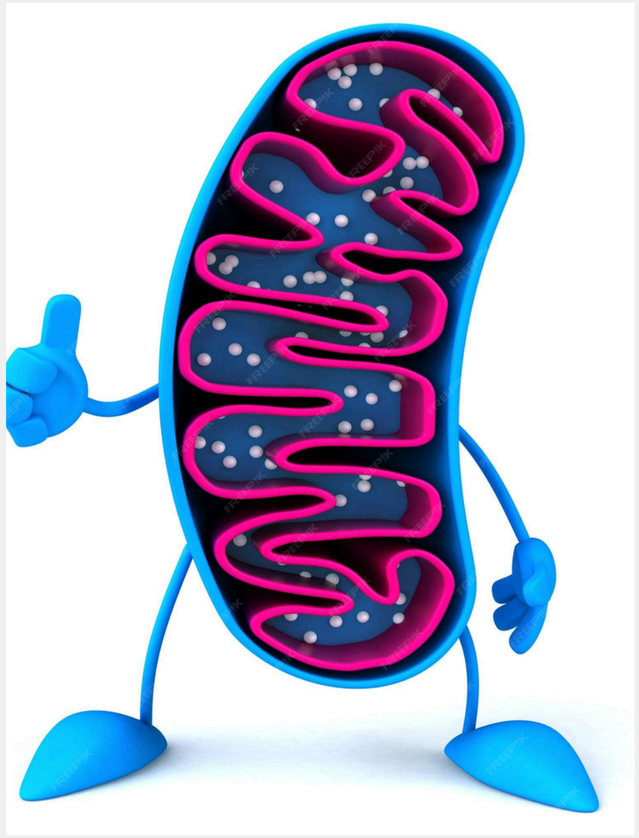

Desempeño en función del tamaño de bloque
Tiempo total (s)
GPU + transferencias (s)
Tiempo promedio (segundos)
1.4
1.2
1.0
0.8
0.6
32
64
128
256
512
1024
Tamaño de bloque (threads por bloque)


Tiempo medio de ejecución por aproximación
Tiempo (segundos)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
Secuencial
OpenMP
CUDA

CONCLUSIÓN

RENDIMIENTO	SIMPLICIDAD
CUDA	SECUENCIAL

DISPONIBILIDAD	¿ENTONCES?
OPENMP	**¡¡SITUACIONAL!!**



GRACIAS