

# Práctica 1: Algoritmos Voraces — String Art

Algoritmia Básica · Grado en Ingeniería Informática · Curso 2025/26

David Puértolas Merenciano (NIA: 900584) · Ibón Castarlenas Cortés (NIA: 900085)

## 1. Diseño del algoritmo

El algoritmo voraz construye una imagen imitando el string art: en cada iteración selecciona los mejores hilos de un conjunto candidato y los dibuja sobre una imagen de trabajo (inicialmente blanca), restando la oscuridad aportada. Se itera hasta convergencia.

### Decisiones de diseño

- Generación de clavos: **n clavos** distribuidos uniformemente en el **perímetro rectangular** de la imagen (no circular), usando **distancia entre clavos = perímetro / n**.
- Selección de candidatos: se generan **p parejas** de clavos aleatorias en cada iteración (heurística del enunciado). Coste: **O(p)**.
- Función de beneficio: suma de oscuridad faltante a lo largo del trayecto del hilo, calculada con **Bresenham:  $\sum \max(0, \text{oscuridad\_original}[x,y] - \text{oscuridad\_actual}[x,y])$** .
- Selección voraz: se ordenan los **p** candidatos por beneficio (**qsort, O(p·log p)**) y se **dibujan los s mejores**.
- **Trazado** de líneas: algoritmo de Bresenham (implementación propia según apéndice). Cada hilo resta 5 unidades de gris por píxel atravesado (opacidad parcial).
- Condiciones de **parada**: (1) sin mejora en 20 iteraciones consecutivas, (2) máximo de 50.000 hilos dibujados.

## 2. Implementación

El programa está implementado en C (estándar C99). Se compila con **gcc -O2 -o hilos code/hilos.c -lm -std=c99**.

La llamada sigue exactamente el formato del enunciado: **./hilos n p s imagen\_entrada imagen\_salida**

Estructuras principales:

- Imagen (matriz dinámica de píxeles en escala de grises)
- Punto (coordenadas de un clavo)
- Hilo (par de clavos + beneficio)
- Parametros (n, p, s).
- La salida por consola incluye: parámetros, nº de hilos dibujados, error final (suma de cuadrados de diferencias) y tiempo de ejecución.

## 3. Experimentación

### 3.1 Análisis teórico de complejidad

Sea **L** la **longitud media de un hilo** en píxeles (Bresenham):  **$L \approx O(\sqrt{W^2 + H^2})$** . El coste por iteración es  **$O(p \cdot L + p \cdot \log p + s \cdot L) \approx O(p \cdot L)$** . Con **T iteraciones** hasta convergencia ( $T \leq 50000/s$ ), el coste total es:

$$O(T \cdot p \cdot L) = O\left(\frac{50000}{s} \cdot p \cdot \sqrt{W^2 + H^2}\right)$$

Por tanto: **aumentar p encarece linealmente cada iteración; aumentar s reduce T** (menos iteraciones) pero baja la calidad; **n afecta indirectamente al rango** de clavos muestreables.

### 3.2 Resultados experimentales

Los experimentos 1,2,3 usan la imagen **messi.pgm**. Error = suma de cuadrados de diferencias (menor es mejor) ->  $\sum (original[x,y] - resultado[x,y])^2$ , x,y son posiciones de cada pixel.

EXP1 — Variación de n (p=500, s=20):

n	Hilos dibujados	Error final	Tiempo (s)
50	29 400	8 994 518 365	1.72
100	36 600	4 777 590 165	2.06
200	37 200	3 489 191 955	1.95
400	37 200	3 177 835 275	1.73
600	37 200	3 152 774 155	1.70
1000	36 600	2 990 129 980	1.75

EXP2 — Variación de p (n=200, s=20):

p	Hilos dibujados	Error final	Tiempo (s)
50	37 800	3 742 193 605	0.38
100	37 800	3 615 324 380	0.58
300	37 200	3 478 670 750	1.26
600	37 200	3 488 028 245	2.34
1000	37 200	3 492 027 245	3.66
2000	37 200	3 495 098 675	6.96

EXP3 — Variación de s (n=200, p=500):

s	Hilos dibujados	Error final	Tiempo (s)
5	28 200	1 868 781 795	5.06
10	31 500	2 299 275 340	3.08
20	37 200	3 484 481 710	1.95
50	50 000	6 290 838 505	1.14
100	50 000	6 311 858 750	0.66
200	50 000	6 383 930 735	0.42

EXP4 — Diversas imágenes (n=200, p=500, s=20):

Imagen	Tamaño	Hilos dibujados	Error final	Tiempo (s)
messi	800x800	37 200	3 485 424 675	1.93
iliaTopuria	675x675	34 200	2 616 019 604	1.41
emoji	512x512	30 600	2 792 400 060	0.86
qr	186x186	16 800	1 221 996 075	0.15

EXP5 — Parámetros del enunciado (n=1000, p=1100, s=30):

Imagen	Tamaño	Hilos dibujados	Error final	Tiempo (s)
messi	800x800	42 300	4 143 639 085	2.53
iliaTopuria	675x675	39 600	3 062 179 164	1.95
emoji	512x512	35 100	2 842 100 160	1.32
qr	186x186	22 500	1 254 206 250	0.28

### 3.3 Conclusiones

- **n (nº de clavos):** más clavos reducen el error final de forma notable **hasta ~400**; por encima la mejora es marginal. El **tiempo apenas varía**.
- **p (candidatos):** el **tiempo crece linealmente con p** (confirmando  $O(p \cdot L)$ ), pero el **error apenas mejora** más allá de  $p=300$ . **Usar p muy alto no compensa**.
- **s (hilos/iteración):** es el parámetro más influyente. **s pequeño (5–10) produce el menor error** (más selectivo) pero **tarda más**. **s grande supera el límite** de 50.000 hilos y **sobreoscurece** la imagen, **disparando el error, tardando poco**.
- La **convergencia y el tiempo de ejecución están fuertemente correlacionados con el tamaño de la imagen**. A mayor tamaño, cada evaluación de Bresenham recorre más píxeles ( $L$  mayor), encareciendo cada iteración. El qr (186×186) converge en 0.15s mientras que messi (800×800) tarda 1.93s con los mismos parámetros. **El contenido también influye:** el emoji, pese a ser más pequeño que iliaTopuria, tiene error similar, probablemente porque sus **áreas planas son más difíciles de aproximar con líneas rectas diagonales**.

Configuraciones recomendadas:

Prioridad	N (número clavos)	P (num parejas clavos)	S (num seleccionados)
Calidad máxima	1000	300	5
Velocidad	200	100	20
Equilibrio calidad/tiempo	400–600	300	10–20

### Referencias

- Demoussel, B., Larboulette, C., & Dattatreya, R. (2022). A greedy algorithm for generative string art. In *Bridges 2022 Conference Proceedings* (pp. 63–70). Aalto, Finland.  
<https://archive.bridgesmathart.org/2022/bridges2022-63.pdf>
- Las matemáticas detrás del hilorama: <https://www.youtube.com/watch?v=1RQMCCSROHA>
- Transparencias de clase.
- LLMs para debugging.