

1. Probar las métricas en su computadora.

Resultados métricas

```
Matriz de brillo:
267 433 81 359 32 350 737 721 460 102
130 983 669 466 582 126 564 745 990 377
24 831 934 422 847 695 569 8 407 972
551 390 580 597 788 413 730 392 907 57
862 944 730 418 112 761 824 900 764 867
813 407 417 613 569 225 648 637 752 896
889 501 747 957 900 568 762 293 52 413
218 209 736 554 721 593 524 589 18 645
901 328 21 172 492 363 397 624 597 950
921 65 809 399 939 225 905 569 61 257
Serial:
Maximo valor: 990
Coordenadas: (1, 8)
Tiempo serial: 0.000000 segundos
Paralelo:
Maximo valor: 990
Coordenadas: (1, 8)
Tiempo paralelo: 0.002000 segundos

--- METRICAS ---
Speedup: 0.000000
Eficiencia: 0.000000
Numero de hilos: 12
Tamaño de la matriz: 10x10
```

Tiempo: serial: 0.000000s

Tiempo paralelo: 0.002000s

Resultados del grupo

```
43 230 604 444 484 800 829 794 730 841 974 902 396 303
382 918 199 206 751 361 812 344 541 926 440 110 752 8
8 642 269 196 938 1 64 54 446 719 543 639 932 61
866 577 47 324 310 631 654 456 565 10 494 578 138 85
406 407 563 21 679 796 606 785 761 132 304 84 861
Serial:
Maximo valor: 999
Coordenadas: (1, 259)
Tiempo serial: 0.001000 segundos
Paralelo:
Maximo valor: 999
Coordenadas: (75, 298)
Tiempo paralelo: 0.002000 segundos

--- METRICAS ---
Speedup: 0.499977
Eficiencia: 0.062497
Numero de hilos: 8
Tamaño de la matriz: 300x300
PS D:\Documentos\Github\PARALELA-13082025> gcc estrella.c -o estrella
PS D:\Documentos\Github\PARALELA-13082025> ./estrella.exe
```

Discusión

Los resultados fueron bastante parecidos esto puede deberse al hardware de la compu pero sobre todo el paralelo fue casi igual, por otro lado algo que se puede destacar es que el valor del paralelizado es casi igual al del valor secuencial.

Una sugerencia que se podría dar es que se haga mas grande el mapa. Tal vez eso nos podría ayudar a ver la diferencia.

2. Realizar una version en Python usando multithreading. Usar misma cantidad de iteraciones o input.

```
PS C:\Users\jjcam\Desktop\Semestre_8\paralela\corto8\PARALELA-13082025> python estrellascorto8.py
Matriz de brillo:
426 750 10 839 845 822 305 875 377 954
198 276 579 446 165 382 127 895 443 267
575 642 178 626 566 191 363 755 981 750
93 543 753 422 599 516 170 151 210 736
79 194 857 349 329 24 469 347 27 990
522 707 762 425 3 6 962 679 891 167
789 594 183 293 101 668 424 108 906 697
591 186 754 384 238 983 536 610 726 935
844 337 826 41 237 122 894 511 221 754
821 774 568 685 190 263 602 444 530 938
Serial:
Maximo valor: 990
Coordenadas: (4, 9)
Tiempo serial: 0.000007 segundos
Paralelo:
Maximo valor: 990
Coordenadas: (4, 9)
Tiempo paralelo: 0.248639 segundos

--- METRICAS ---
Speedup: 0.000029
Eficiencia: 0.000003
Numero de procesos: 10
Tamaño de la matriz: 10x10
```

3. Evaluar y argumentar las limitaciones que tiene trabajar con bajo nivel.

Para encontrar el brillo máximo en la matriz, en C con OpenMP hay que armar la lógica del paralelismo. Por otro lado, en Python eso se hace con pocas líneas. Lo que debería ser buscar el valor más grande termina siendo un código mucho más largo y técnico en bajo nivel.

En el ejercicio, tenemos que usar cosas como: `#pragma omp critical` en C, para que los resultados no sean equivocados mas en estos de que incluyen contadores y comparaciones, porque varios hilos intentarían actualizar la misma variable al mismo tiempo.