A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

19-1-2021

Métodos Numericos para la Computación

Entrega 9

4ºCurso

Grupo Prácticas 11

Several thin, curved lines in dark blue and light grey that sweep upwards from the bottom left towards the center of the page.

Alejandro Daniel Herrera Cardenes
Carlos Eduardo Pacichana Bastidas
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Índice

Practica 1	2
Descripción	2
Trabajo realizado	2
Practica 2	3
Descripción	3
Trabajo realizado	3
Practica 3	5
Descripción	5
Trabajo realizado	5
Practica 4	6
Descripción	6
Trabajo realizado	6
Gráfica	7
Practica 5	8
Descripción	8
Trabajo realizado	8
Practica 6	9
Descripción	9
Trabajo realizado	9
Gráfica	10
Practica 7	10
Descripción	10
Trabajo realizado	10
Practica 8	11
Descripción	11
Trabajo realizado	11
Gráfica	12

Practica 1

Descripción

- Escribe un programa que declare tres vectores de 100 números en coma flotante e inicialice dos de ellos
 - Inicializa cada elemento del primero con el valor de su índice
 - Inicializa cada elemento del segundo con el doble del valor de su índice
- Crea tres vectores equivalentes en el dispositivo CUDA
- Copia en el dispositivo los dos vectores inicializados
- Lanza 100 núcleos, uno para cada elemento, que calculen el producto de los elementos correspondientes de ambos vectores y pongan el resultado en el tercer vector
- Recupera el vector resultado desde el dispositivo, suma sus elementos para obtener el producto escalar de los dos vectores inicializados y muestra el resultado en pantalla; comprueba que sea correcto
- Libera la memoria reservada en el dispositivo y reinícialo para dar por finalizado el programa

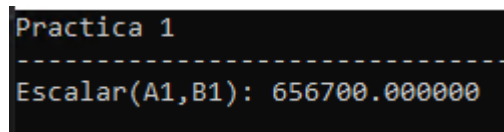
Trabajo realizado

Los tres vectores los creamos de tipo *double* de doble precisión, para el cálculo del producto escalar lanzamos un bloque con 100 hilos que se encargaran de operar cada producto de cada índice guardando el resultado en el tercer vector el cual recuperamos y sumamos todos sus elementos en modo secuencial.

Cuda

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).



```
Practica 1
-----
Escalar(A1,B1): 656700.000000
```

Resultado de la ejecución 1

Practica 2

Descripción

- Modifica el programa que calcula el producto escalar de dos vectores para que cada hilo muestre por pantalla el índice tridimensional del bloque y su índice tridimensional de hilo junto con el elemento que está calculando
- Modifica el programa para que el trabajo se divida en 10 bloques y comprueba que el resultado sigue siendo el correcto
- Verifica si el comportamiento del programa varía entre distintas ejecuciones

Trabajo realizado

Para mostrar los índices de los hilos y de los bloques añadimos la instrucción `printf()` imprimiendo todos ellos así como los valores de los vectores con los que vamos a operar y su resultado.

En el caso de usar 10 bloques debemos pasarle este valor a la función al lanzar el núcleo así como los hilos necesarios para operar los vectores enteros en este caso 10, ($10h \cdot 10b = 100$). También podemos ver esto en las imágenes de ejecución donde vemos que los índices de los bloques dejan de ser todos 0 y toman valores entre 0-9.

Y se hace el mismo procedimiento con los hilos.

Cuda

Resultado Ejecución:

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).

```
Practica 2
-----
H-> [X:96 Y:0 Z:0] B->[X:0 Y:0 Z:0] {96.000000 + 192.000000}={18432.000000}
H-> [X:97 Y:0 Z:0] B->[X:0 Y:0 Z:0] {97.000000 + 194.000000}={18818.000000}
H-> [X:98 Y:0 Z:0] B->[X:0 Y:0 Z:0] {98.000000 + 196.000000}={19208.000000}
H-> [X:99 Y:0 Z:0] B->[X:0 Y:0 Z:0] {99.000000 + 198.000000}={19602.000000}
H-> [X:32 Y:0 Z:0] B->[X:0 Y:0 Z:0] {32.000000 + 64.000000}={2048.000000}
H-> [X:33 Y:0 Z:0] B->[X:0 Y:0 Z:0] {33.000000 + 66.000000}={2178.000000}
H-> [X:34 Y:0 Z:0] B->[X:0 Y:0 Z:0] {34.000000 + 68.000000}={2312.000000}
H-> [X:35 Y:0 Z:0] B->[X:0 Y:0 Z:0] {35.000000 + 70.000000}={2450.000000}
H-> [X:36 Y:0 Z:0] B->[X:0 Y:0 Z:0] {36.000000 + 72.000000}={2592.000000}
H-> [X:37 Y:0 Z:0] B->[X:0 Y:0 Z:0] {37.000000 + 74.000000}={2738.000000}
H-> [X:38 Y:0 Z:0] B->[X:0 Y:0 Z:0] {38.000000 + 76.000000}={2888.000000}
H-> [X:39 Y:0 Z:0] B->[X:0 Y:0 Z:0] {39.000000 + 78.000000}={3042.000000}
H-> [X:40 Y:0 Z:0] B->[X:0 Y:0 Z:0] {40.000000 + 80.000000}={3200.000000}
```

Resultado de la ejecución 1

```

H-> [X:85 Y:0 Z:0] B->[X:0 Y:0 Z:0] {85.000000 + 170.000000}={14450.000000}
H-> [X:86 Y:0 Z:0] B->[X:0 Y:0 Z:0] {86.000000 + 172.000000}={14792.000000}
H-> [X:87 Y:0 Z:0] B->[X:0 Y:0 Z:0] {87.000000 + 174.000000}={15138.000000}
H-> [X:88 Y:0 Z:0] B->[X:0 Y:0 Z:0] {88.000000 + 176.000000}={15488.000000}
H-> [X:89 Y:0 Z:0] B->[X:0 Y:0 Z:0] {89.000000 + 178.000000}={15842.000000}
H-> [X:90 Y:0 Z:0] B->[X:0 Y:0 Z:0] {90.000000 + 180.000000}={16200.000000}
H-> [X:91 Y:0 Z:0] B->[X:0 Y:0 Z:0] {91.000000 + 182.000000}={16562.000000}
H-> [X:92 Y:0 Z:0] B->[X:0 Y:0 Z:0] {92.000000 + 184.000000}={16928.000000}
H-> [X:93 Y:0 Z:0] B->[X:0 Y:0 Z:0] {93.000000 + 186.000000}={17298.000000}
H-> [X:94 Y:0 Z:0] B->[X:0 Y:0 Z:0] {94.000000 + 188.000000}={17672.000000}
H-> [X:95 Y:0 Z:0] B->[X:0 Y:0 Z:0] {95.000000 + 190.000000}={18050.000000}

Escalar(A1,B1): 656700.000000

```

Resultado de la ejecución 1.1

```

Practica 2
-----
H-> [X:0 Y:0 Z:0] B->[X:3 Y:0 Z:0] {30.000000 + 60.000000}={1800.000000}
H-> [X:1 Y:0 Z:0] B->[X:3 Y:0 Z:0] {31.000000 + 62.000000}={1922.000000}
H-> [X:2 Y:0 Z:0] B->[X:3 Y:0 Z:0] {32.000000 + 64.000000}={2048.000000}
H-> [X:3 Y:0 Z:0] B->[X:3 Y:0 Z:0] {33.000000 + 66.000000}={2178.000000}
H-> [X:4 Y:0 Z:0] B->[X:3 Y:0 Z:0] {34.000000 + 68.000000}={2312.000000}
H-> [X:5 Y:0 Z:0] B->[X:3 Y:0 Z:0] {35.000000 + 70.000000}={2450.000000}
H-> [X:6 Y:0 Z:0] B->[X:3 Y:0 Z:0] {36.000000 + 72.000000}={2592.000000}
H-> [X:7 Y:0 Z:0] B->[X:3 Y:0 Z:0] {37.000000 + 74.000000}={2738.000000}
H-> [X:8 Y:0 Z:0] B->[X:3 Y:0 Z:0] {38.000000 + 76.000000}={2888.000000}
H-> [X:9 Y:0 Z:0] B->[X:3 Y:0 Z:0] {39.000000 + 78.000000}={3042.000000}
H-> [X:0 Y:0 Z:0] B->[X:0 Y:0 Z:0] {0.000000 + 0.000000}={0.000000}
H-> [X:1 Y:0 Z:0] B->[X:0 Y:0 Z:0] {1.000000 + 2.000000}={2.000000}
H-> [X:2 Y:0 Z:0] B->[X:0 Y:0 Z:0] {2.000000 + 4.000000}={8.000000}

```

Resultado de la ejecución 2

```

H-> [X:3 Y:0 Z:0] B->[X:9 Y:0 Z:0] {93.000000 + 186.000000}={17298.000000}
H-> [X:4 Y:0 Z:0] B->[X:9 Y:0 Z:0] {94.000000 + 188.000000}={17672.000000}
H-> [X:5 Y:0 Z:0] B->[X:9 Y:0 Z:0] {95.000000 + 190.000000}={18050.000000}
H-> [X:6 Y:0 Z:0] B->[X:9 Y:0 Z:0] {96.000000 + 192.000000}={18432.000000}
H-> [X:7 Y:0 Z:0] B->[X:9 Y:0 Z:0] {97.000000 + 194.000000}={18818.000000}
H-> [X:8 Y:0 Z:0] B->[X:9 Y:0 Z:0] {98.000000 + 196.000000}={19208.000000}
H-> [X:9 Y:0 Z:0] B->[X:9 Y:0 Z:0] {99.000000 + 198.000000}={19602.000000}

Escalar(A1,B1): 656700.000000

```

Resultado de la ejecución 2.1

Practica 3

Descripción

- Escribe un programa que multiplique dos matrices cuadradas de números en coma flotante utilizando la función MatrixMultiplication
 - Las matrices deben declararse como vectores de $N*N$ elementos
 - El valor de cada elemento de la primera matriz será la suma de su número de fila más su número de columna
 - El valor de cada elemento de la segunda matriz será la resta de su número de fila menos su número de columna
- Muestra por pantalla el resultado de una multiplicación con $N = 3$ para comprobar que el programa funciona correctamente

Trabajo realizado

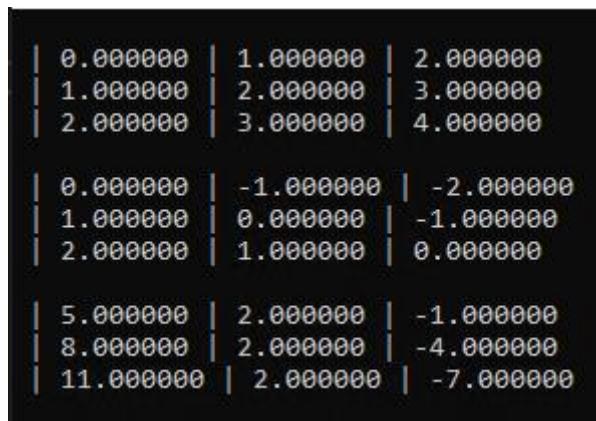
Reservamos la matriz dinámicamente de $N*N$.

Para la prueba declaramos N igual a 3. Asignamos los valores de la matriz siguiente el enunciado un bucle for y se lo pasamos a la función de multiplicar matrices.

Como podemos observar el resultado es el correcto.

Cuda

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).



0.000000	1.000000	2.000000
1.000000	2.000000	3.000000
2.000000	3.000000	4.000000
0.000000	-1.000000	-2.000000
1.000000	0.000000	-1.000000
2.000000	1.000000	0.000000
5.000000	2.000000	-1.000000
8.000000	2.000000	-4.000000
11.000000	2.000000	-7.000000

Resultado de la ejecución 1

Practica 4

Descripción

- Mide el tiempo que tarda en ejecutarse la multiplicación de matrices para valores crecientes del tamaño de la matriz que sean potencia de 2
 - Mide solo el tiempo que tarda la función MatrixMultiplication
 - Dibuja una gráfica con el tiempo que tarda para los tamaños 1, 2, 4, 8, 16 y 32
 - La precisión de los contadores no es suficiente, así que tendrás que ejecutar la función al menos 10000 veces para cada tamaño y calcular la media

Trabajo realizado

Añadimos un temporizador antes y después de hacer la función de multiplicación y realizamos. A pesar de ejecutar el método 10000 veces el tiempo no subía de 0 hasta tamaños superiores a 8 como podemos observar en la gráfica.

Cuda

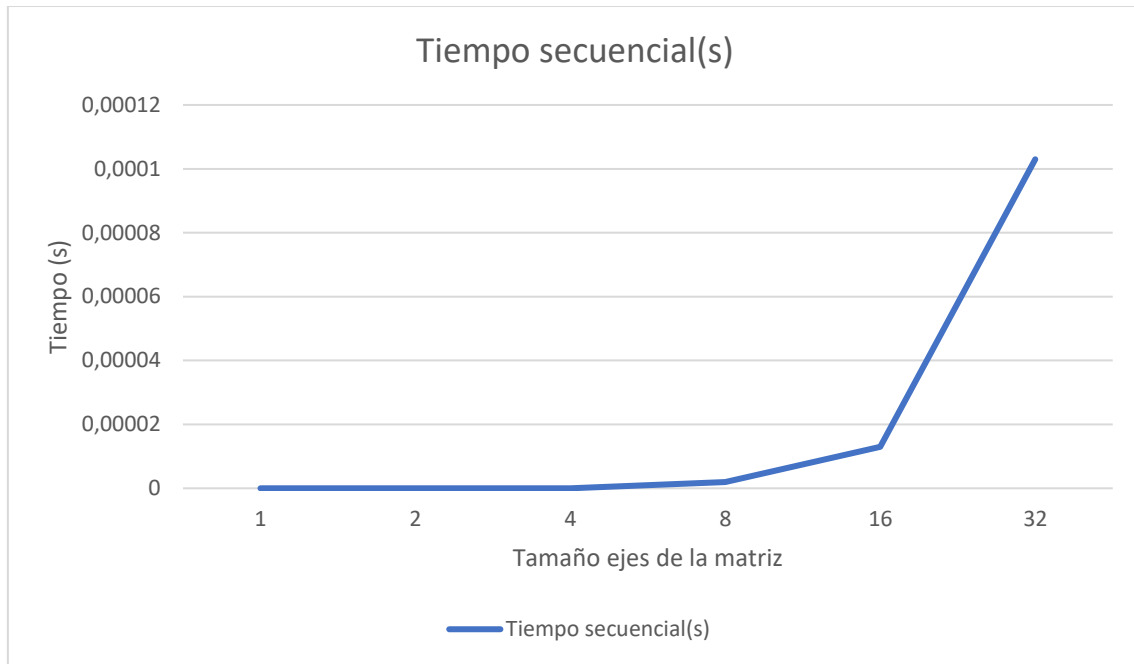
El resultado de la ejecución lo podemos ver en la imagen inferior (***Resultado de ejecución***).

```
Practica 4
-----
Tiempo secuencial: 0.001000 segundos
```

Resultado de la ejecución

Gráfica

Podemos observar que a partir de las 1000 iteraciones los hilos empiezan a aumentar el tiempo de las operaciones al haber más sumas que realizar. (*Gráfica comparación*)



Gráfica tiempo secuencial

Practica 5

Descripción

- Escribe un programa que multiplique dos matrices cuadradas de números en coma flotante utilizando CUDA
 - Transforma la función MatrixMultiplication en un núcleo
- Sustituye la variable i por el índice de hilo threadIdx.x
- Sustituye la variable j por el índice de hilo threadIdx.y
 - El núcleo debe lanzarse usando un único bloque con sus hilos estructurados bidimensionalmente para que coincidan con el tamaño de las matrices
- Muestra por pantalla el resultado de una multiplicación con $N = 3$ para comprobar que el programa funciona correctamente

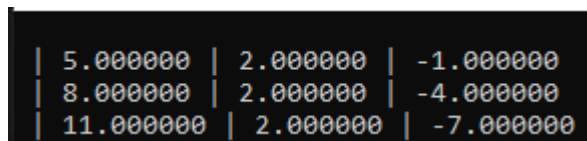
Trabajo realizado

Creamos un nuevo kernel donde realizamos las operaciones de multiplicación. El núcleo recibirá siempre un bloque y el número de hilos necesarios.

Como en este caso la matriz de tamaño máximo es 32×32 lo que daría 1024 hilos, no tenemos problemas a la hora de la ejecución. En caso de que quisiéramos superar este tamaño deberemos añadir más bloques.

Cuda

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).



5.000000	2.000000	-1.000000
8.000000	2.000000	-4.000000
11.000000	2.000000	-7.000000

Resultado de la ejecución

Practica 6

Descripción

- Mide el tiempo que tarda en ejecutarse la multiplicación de matrices para valores crecientes del tamaño de la matriz que sean potencia de 2
 - Mide el tiempo que tarda la copia de datos y la ejecución del núcleo
 - Ejecuta el núcleo al menos 10 veces para cada matriz y calcula el tiempo medio
 - Dibuja una gráfica con el tiempo que tarda para los tamaños 1, 2, 4, 8, 16 y 32
- Compara la gráfica con la que obtuviste para la ejecución secuencial y razona las diferencias entre los dos comportamientos
- Comprueba qué ocurre si intentas ejecutar el programa con tamaños mayores de matriz y determina la causa

Trabajo realizado

Debido a que los contadores de cuda tienen mejor precisión podemos ver los tiempos con mayor exactitud en la siguiente gráfica.

Vemos que cuda obtiene mejores resultados aunque al final no se nota tanta diferencia porque el tamaño de las matrices no es tan significativa.

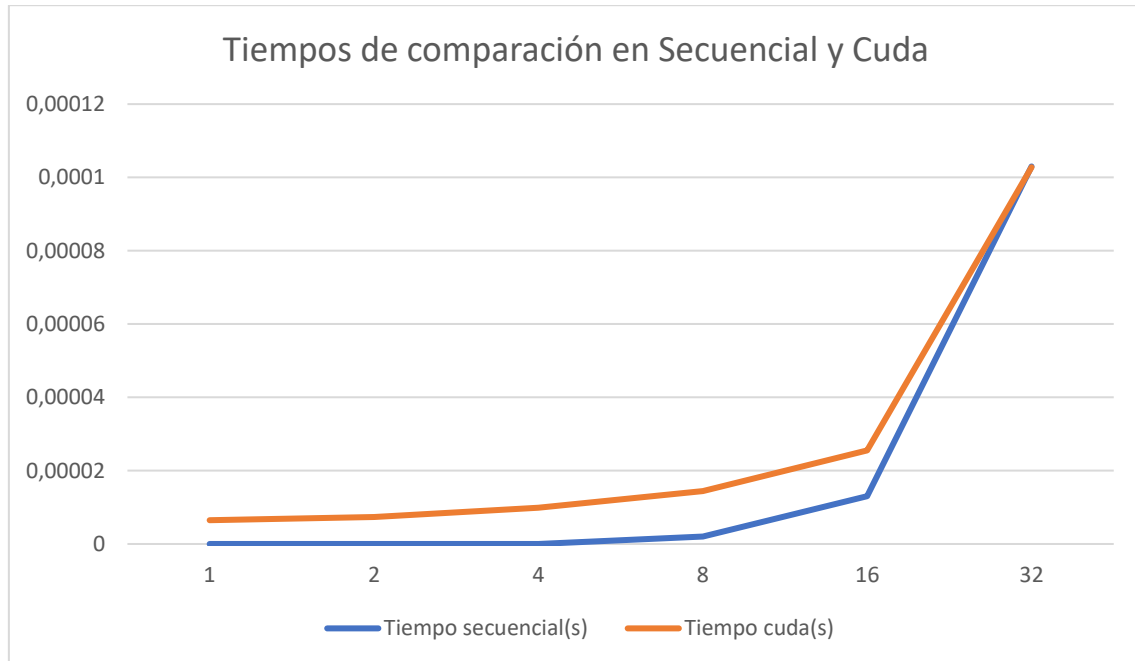
Cuda

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).

```
Practica 6
-----
Tiempo: 0.000102
```

Resultado de la ejecución

Gráfica



Gráfica comparación

Practica 7

Descripción

- Modifica el programa que multiplica matrices cuadradas en CUDA para que las matrices grandes sean divididas en bloques
 - El tamaño de los bloques será siempre 32x32
 - Asumiremos que las dimensiones de las matrices serán potencias de dos
 - Consideraremos matriz grande aquella de dimensiones superiores a 32x32
- Para comprobar que el programa funciona correctamente para matrices grandes, imprime los valores de la diagonal principal de la matriz y compáralos con los valores que da la versión secuencial del programa

Trabajo realizado

Para matrices que superen un tamaño de 32*32 dividiremos este entre el número máximo de hilos por bloque y generaremos todos los bloques que sean necesarios para operar la matriz entera.

Para la prueba hemos usado el kernel creado para operar matrices grandes para multiplicar el ejemplo y observamos que el resultado es el mismo.

Cuda

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).

```
| 5.000000 | 2.000000 | -7.000000
```

Resultado de la ejecución

Practica 8

Descripción

- Extiende la gráfica que compara el tiempo de ejecución de las versiones secuencial y CUDA de la multiplicación de matrices
 - Añade al menos el tiempo para los tamaños 64, 128, 256, 512 y 1024
 - La versión secuencial solo debe ejecutarse 10 veces para estos tamaños
 - Asegúrate de que las matrices están declaradas utilizando memoria dinámica
- Razona las diferencias entre la evolución de las dos líneas de la gráfica a la vista de la nueva información añadida

Trabajo realizado

En la gráfica podemos observar que el crecimiento del tiempo es exponencial para las ejecuciones en secuencial mientras que en cuda se ejecuta mas óptimamente.

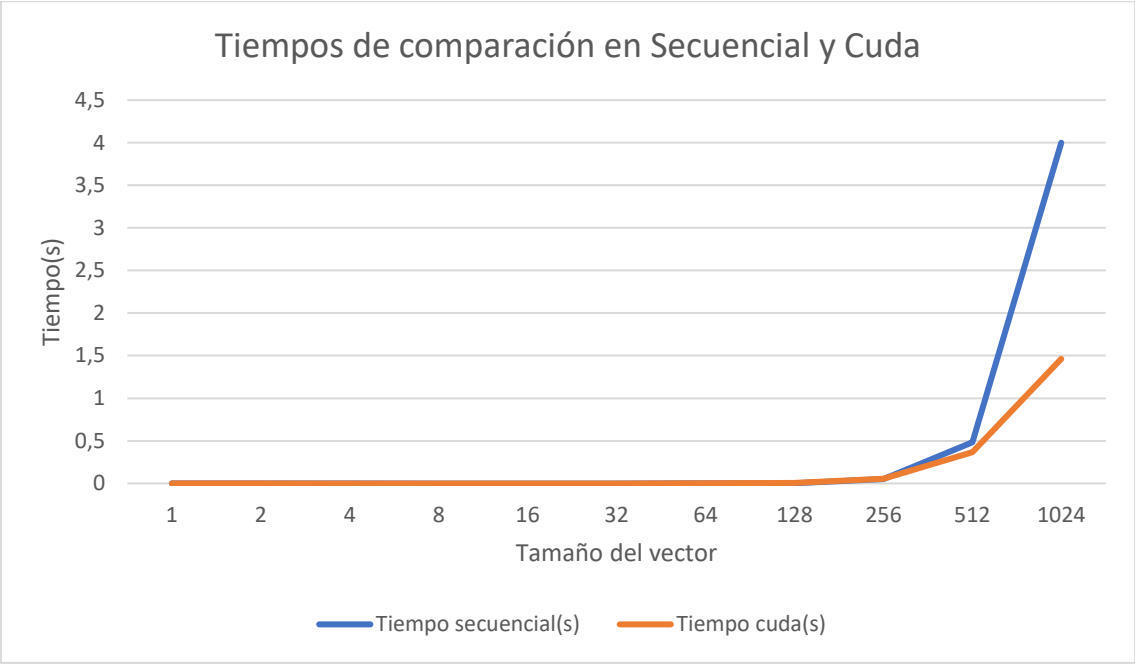
Cuda

El resultado de la ejecución lo podemos ver en la imagen inferior (**Resultado de ejecución**).

```
Practica 8
-----
Tiempo: 1.462456
```

Resultado de la ejecución

Gráfica



Gráfica comparación