High Performance Computing
Departamento de Ingeniería Informática
LAB2 : SIMD-vectorial + SIMD-hebras

Integrantes: Nicolas Alarcón
Sebastian Garay
Profesor: Fernando Rannou
Ayudante: Marcela Rivera

Índice

Utilizando el programa

Para la utilización del programa, se requiere un computador con el compilador gcc instalado, además de la capacidad de utilizar Makefiles.

Compilar

Para compilar el programa, es necesario ubicarse en la carpeta con el código, y escribir “make”. Notar que luego de terminar este comando, se creará el archivo “sort”, el cual posteriormente se ejecutará.
Si se desea hacer una compilación limpia, utilizar el comando “make clean”, con lo que se eliminarán tanto los archivos con extensión .o, como el ejecutable. Cabe destacar que de no encontrarse alguno de estos archivos, se mostrará por pantalla un mensaje de error indicando que no se pudieron eliminar.

Ejecutar

Luego de compilar el programa, es posible ejecutarlo utilizando el comando “./simdsort”, agregando los siguientes argumentos:

- -i: Se utiliza para indicar el nombre del archivo de entrada.
- -o: Se utiliza para indicar el nombre del archivo de salida.
- -N: Se utiliza para indicar la cantidad de valores presentes en el archivo de entrada (múltiplos de 16).
- -d: Se utiliza para indicar el nivel de debug del programa, en donde de ser 0, no se imprimirá nada por pantalla, y de ser 1, se imprimirán los resultados.
- -l: La cantidad de niveles en los que se dividirán los datos del archivo de entrada.
- -h: La cantidad de hebras que se utilizarán.

Un ejemplo para utilizar el programa sería el siguiente:
“./sort -i archivoEntrada.raw -o archivoSalida.raw -N 64 -d 0 -l 2 -h 3”

Especificaciones

El computador donde se ejecutaron las pruebas presenta la siguientes especificaciones:

Ítem	**Valor**
Disco Duro	512Gb SSD
Procesador	Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, 4 Núcleos
Memoria Ram	16 Gb
Tarjeta de Video	Intel UHD Graphics 620

Tabla 1: Especificaciones técnicas

Archivos de prueba

Los archivos de prueba utilizados para probar la ejecución del programa se especifican en la siguiente tabla:

Nombre	**Cantidad de datos**	**Tamaño**
64floats.raw	64	256 bytes
65536floats.raw	65536	262 kilobytes
640000.raw	640000	2,6 megabytes

Tabla 2: Especificación de los archivos de prueba

Nota: El archivo 640000.raw fue creado automáticamente con valores aleatorios, mientras que 64floats.raw y 65536floats.raw fueron proporcionados por la coordinación.

Mediciones

Se realizaron mediciones con tres archivos. El primer archivo de 64 números flotantes, el segundo archivo de 6536 números flotantes y por último un archivo de 640000 números flotantes . Cada uno de estos archivos realizó diferentes pruebas. Primero se aumentó en los niveles de recursividad de 2,3,4,5 en el caso de los archivos de 6536 y 640000 elementos y en el caso del archivo de 64 elementos solo se realizaron pruebas de 2 niveles debido a que no se pueden cada SIMD tendría menos de 16 datos. Junto con esto a cada uno de los niveles se hicieron diferentes pruebas con la cantidad de hebras, para los casos de 6536 y 640000 se realizaron de 1 a 100 hebras y en el caso del archivo de 64 elementos se realizaron pruebas hasta 10000 hebras, debido a que con 100 hebras no habían diferencias de tiempo.
A continuación es posible visualizar las diferentes pruebas junto con los speed up conseguidos con respecto al laboratorio 1.

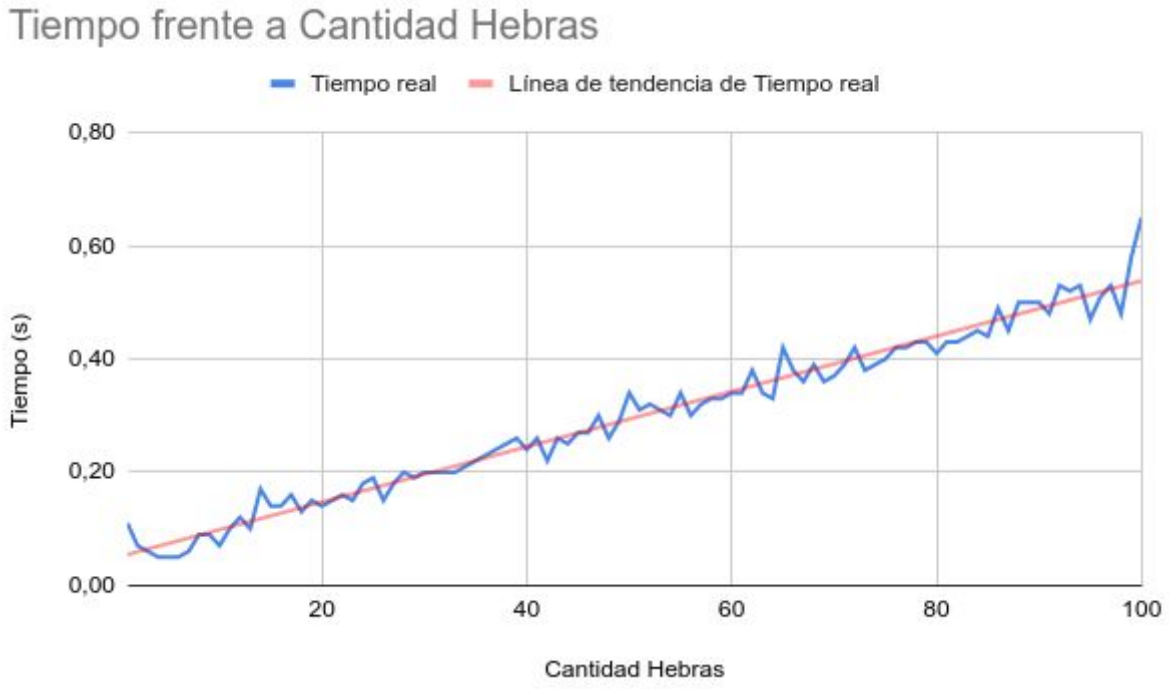

Gráfico 1: Tiempo(s) vs Cantidad de hebras. 2 Niveles de recursividad y 6536 Elementos

	Tiempo	N° Hebras
Mínimo	0,05	4, 5 y 6
Máximo	0,65	100

Tabla 3: Mínimo y máximo tiempo conseguido.2 Niveles de recursividad y 6536 Elementos

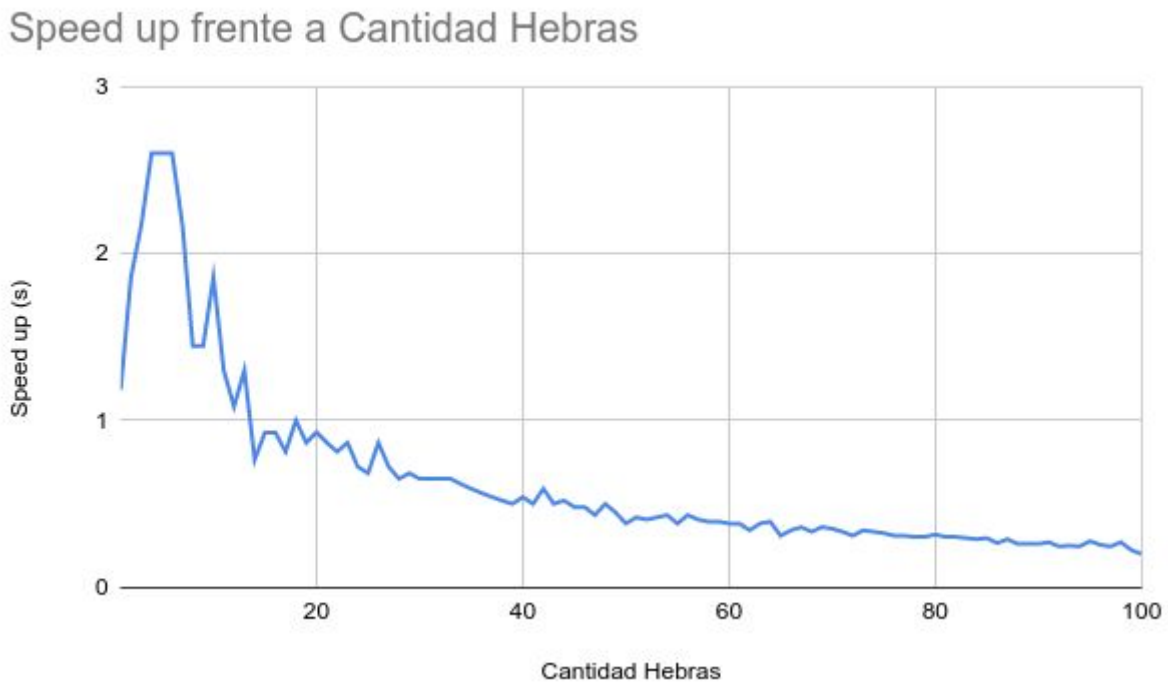

Gráfico 2: Speed up vs Cantidad de hebras.2 Niveles de recursividad y 6536 Elementos

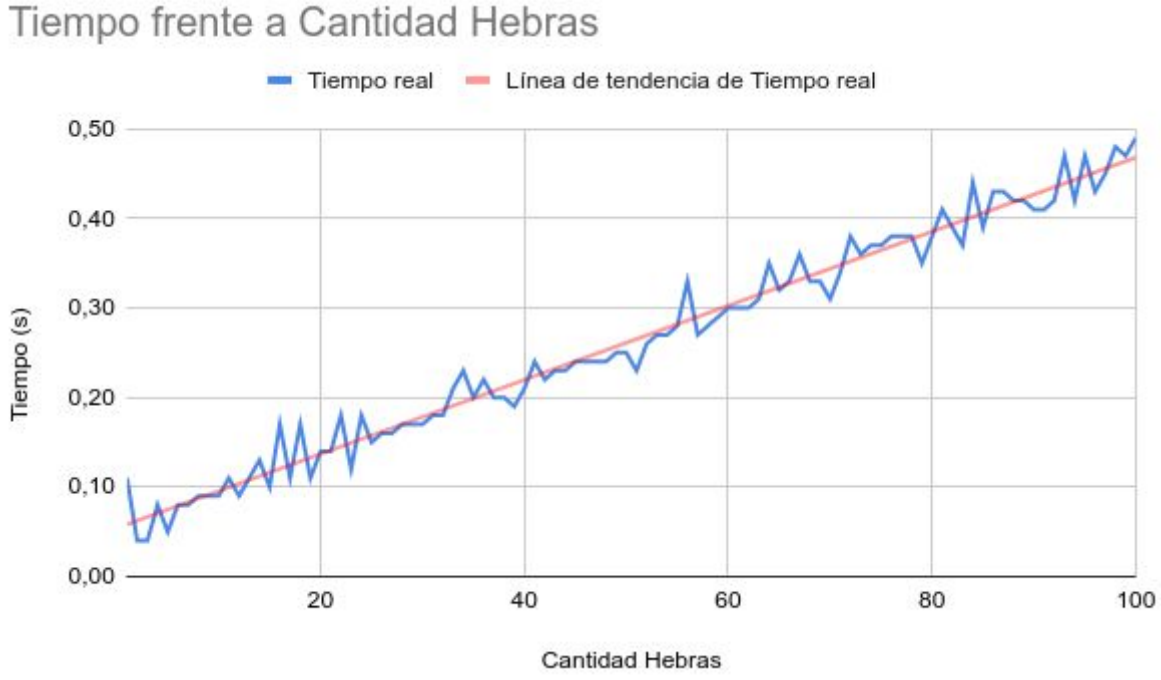

Gráfico 3: Tiempo(s) vs Cantidad de hebras. 3 Niveles de recursividad y 6536 Elementos

	Tiempo	N° Hebras
Mínimo	0,04	2 y3
Máximo	0,49	100

Tabla 4: Mínimo y máximo tiempo conseguido. 3 Niveles de recursividad y 6536 Elementos

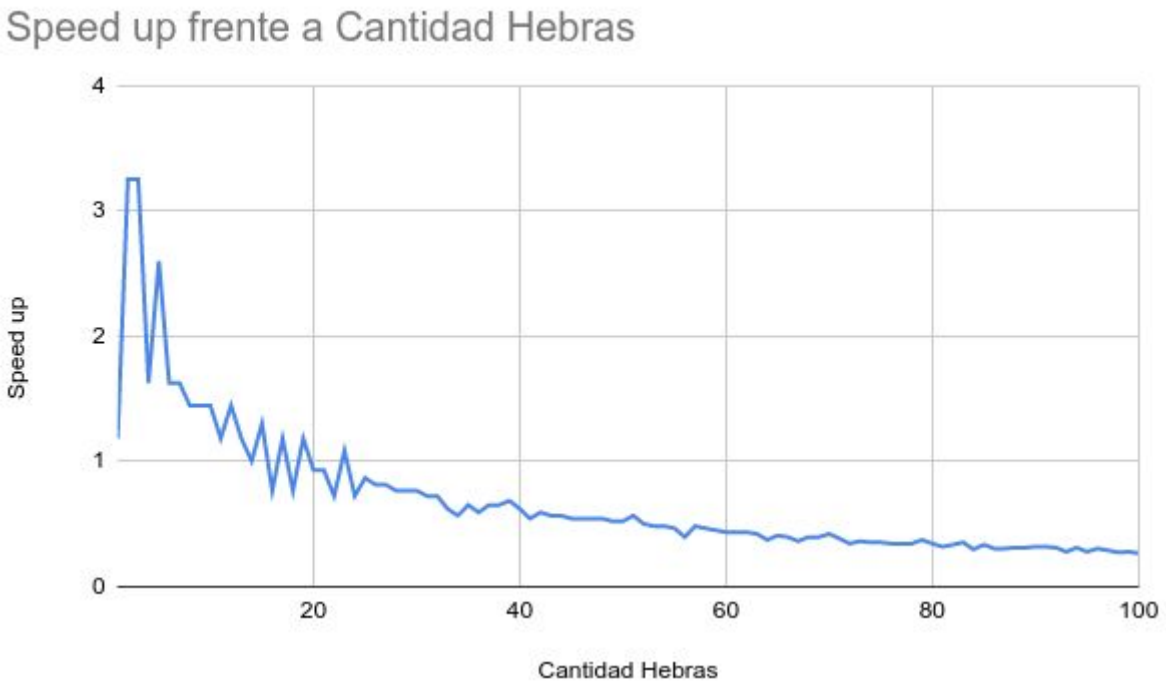

Gráfico 4: Speed up vs Cantidad de hebras.3 Niveles de recursividad y 6536 Elementos

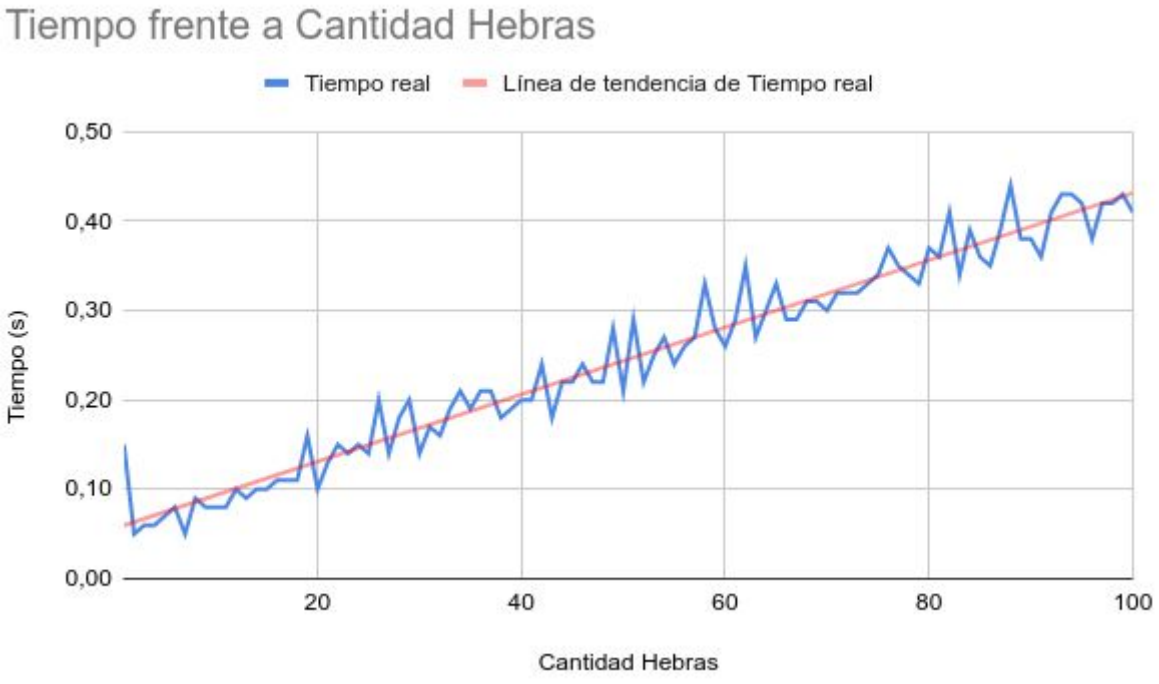

Gráfico 5: Tiempo(s) vs Cantidad de hebras. 4 Niveles de recursividad y 6536 Elementos

	Tiempo	N° Hebras
Mínimo	0,05	2
Máximo	0,44	88

Tabla 5: Mínimo y máximo tiempo conseguido. 4 Niveles de recursividad y 6536 Elementos

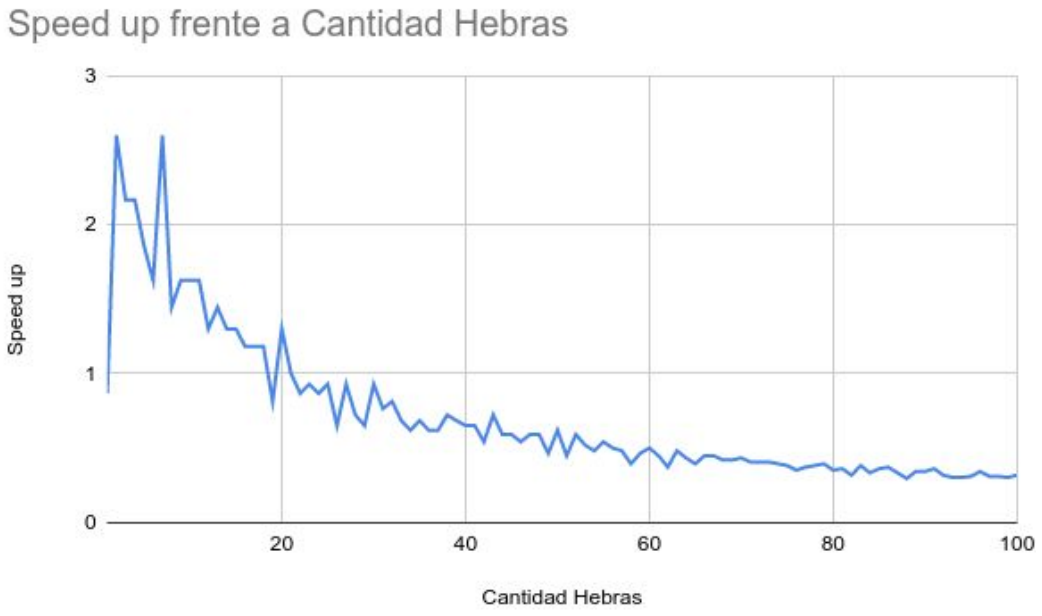

Gráfico 6: Speed up vs Cantidad de hebras.4 Niveles de recursividad y 6536 Elementos

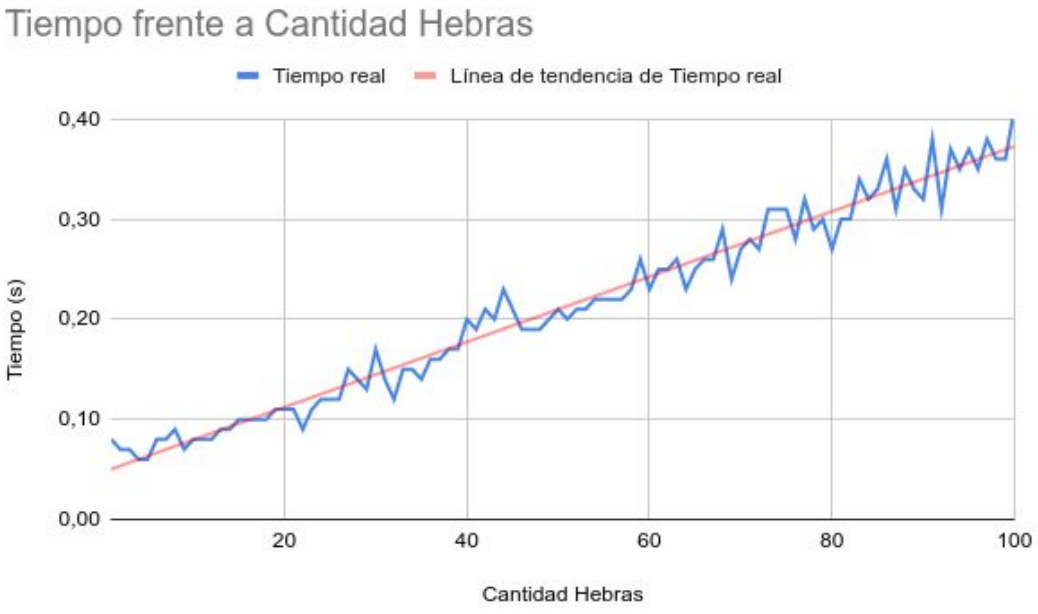

Gráfico 7: Tiempo(s) vs Cantidad de hebras. 5 Niveles de recursividad y 6536 Elementos

	Tiempo	N° Hebras
Mínimo	0,06	4,5
Máximo	0,41	100

Tabla 6: Mínimo y máximo tiempo conseguido. 5 Niveles de recursividad y 6536 Elementos

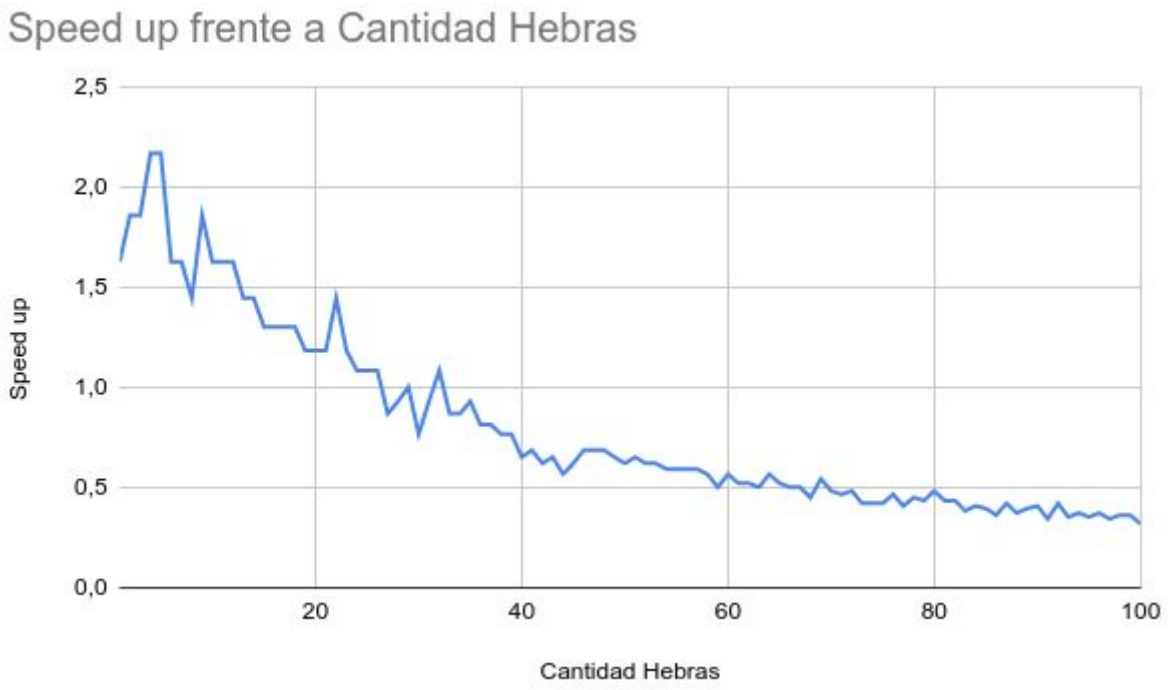

Gráfico 8: Speed up vs Cantidad de hebras.5 Niveles de recursividad y 6536 Elementos

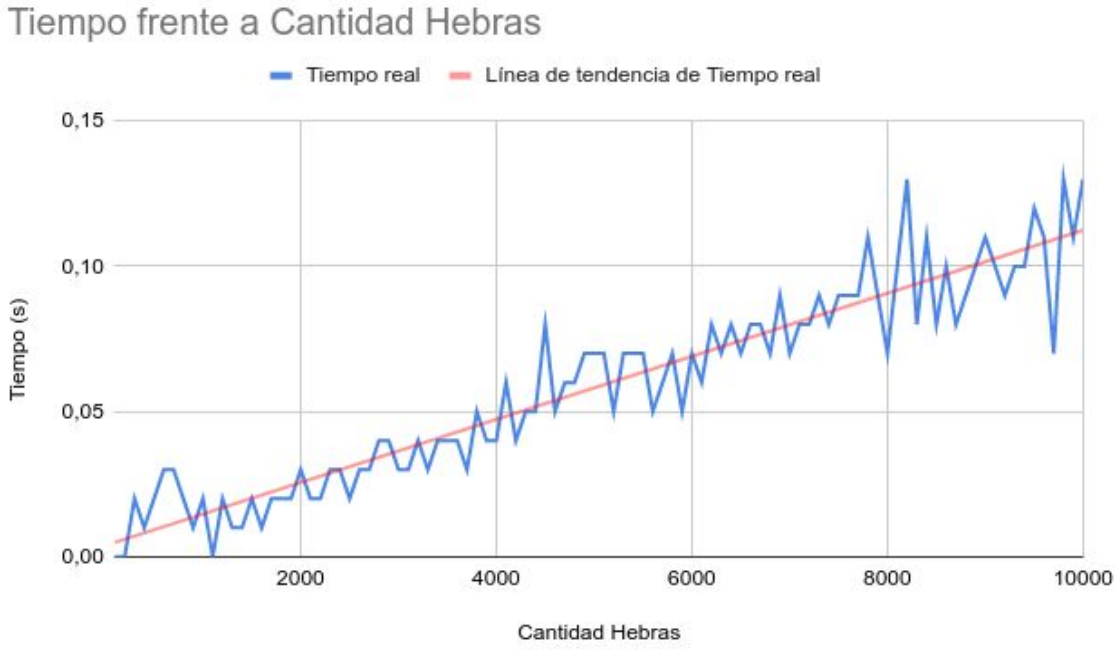

Gráfico 9: Tiempo(s) vs Cantidad de hebras. 2 Niveles de recursividad y 64 Elementos

	Tiempo	N° Hebras
Mínimo	0,00	1,101 y 1001
Máximo	0,13	8101, 9701, 9901 y 10001

Tabla 7: Mínimo y máximo tiempo conseguido. 2 Niveles de recursividad y 64 Elementos

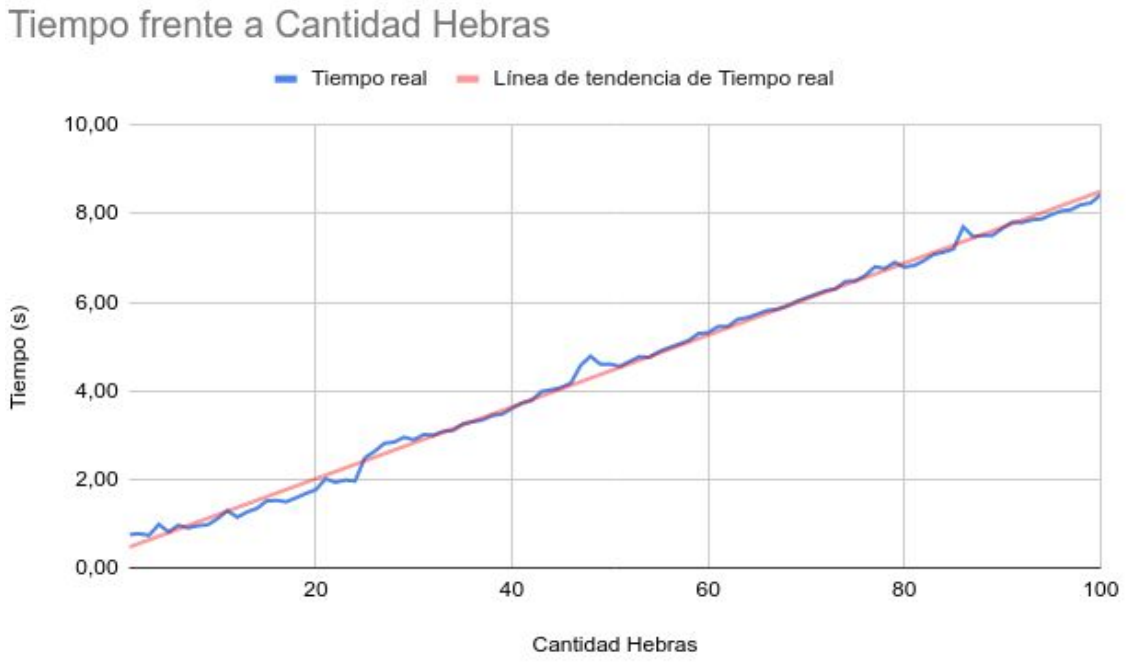

Gráfico 10: Tiempo(s) vs Cantidad de hebras. 2 Niveles de recursividad y 640000 Elementos

	Tiempo	N° Hebras
Mínimo	0,73	3
Máximo	8,43	100

Tabla 8: Mínimo y máximo tiempo conseguido. 2 Niveles de recursividad y 640000 Elementos

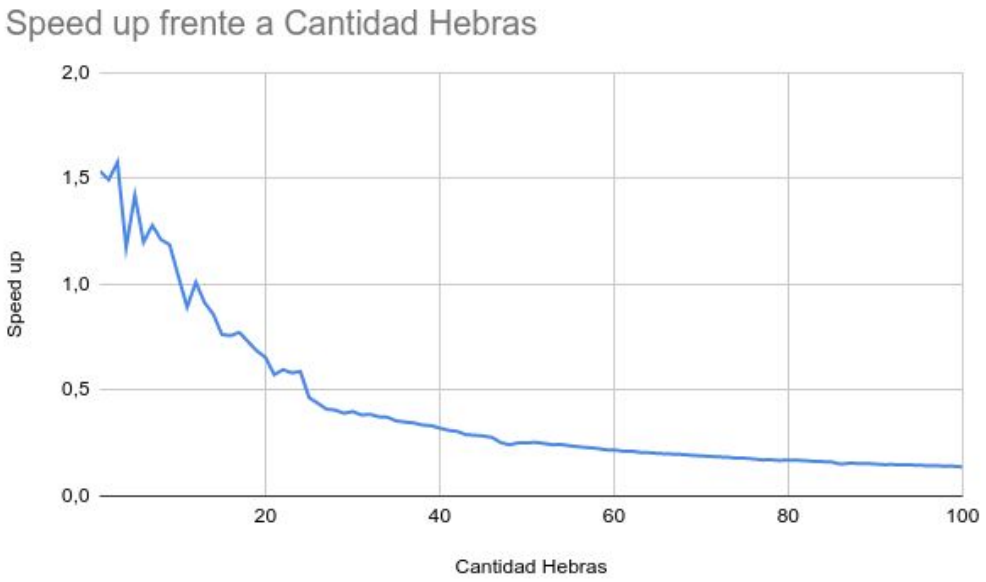

Gráfico 11: Speed up vs Cantidad de hebras. 2 Niveles de recursividad y 640000 Elementos

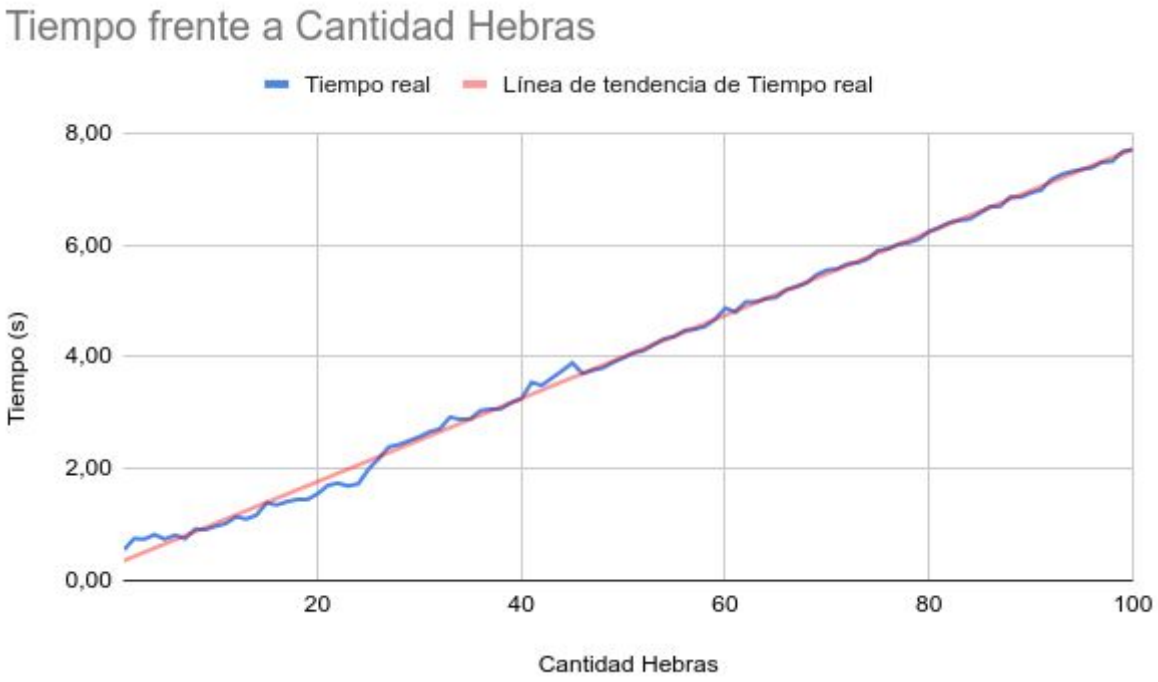

Gráfico 12: Tiempo(s) vs Cantidad de hebras. 3 Niveles de recursividad y 640000 Elementos

	Tiempo	N° Hebras
Mínimo	0,55	1
Máximo	7,70	100

Tabla 9: Mínimo y máximo tiempo conseguido. 3 Niveles de recursividad y 640000 Elementos

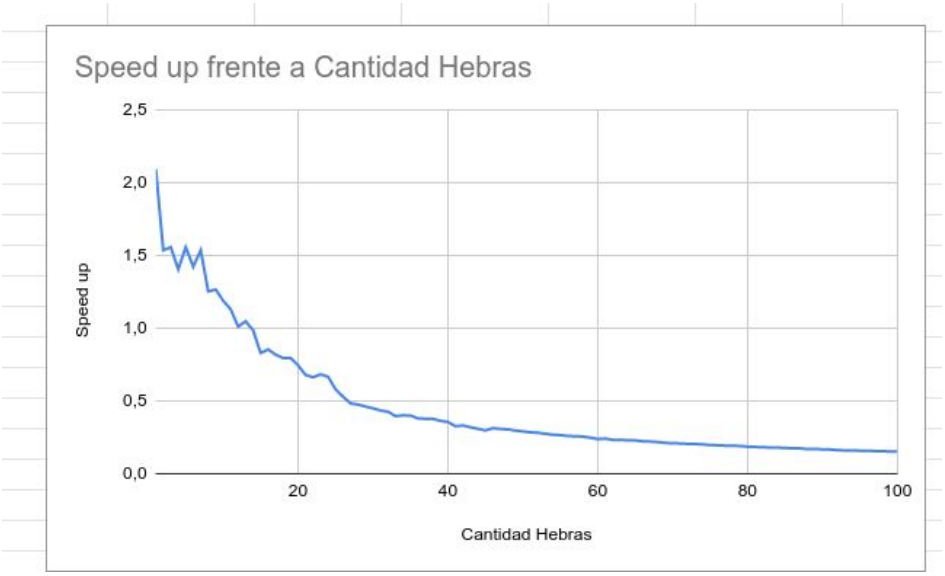

Gráfico 12: Speed up vs Cantidad de hebras. 3 Niveles de recursividad y 640000 Elementos

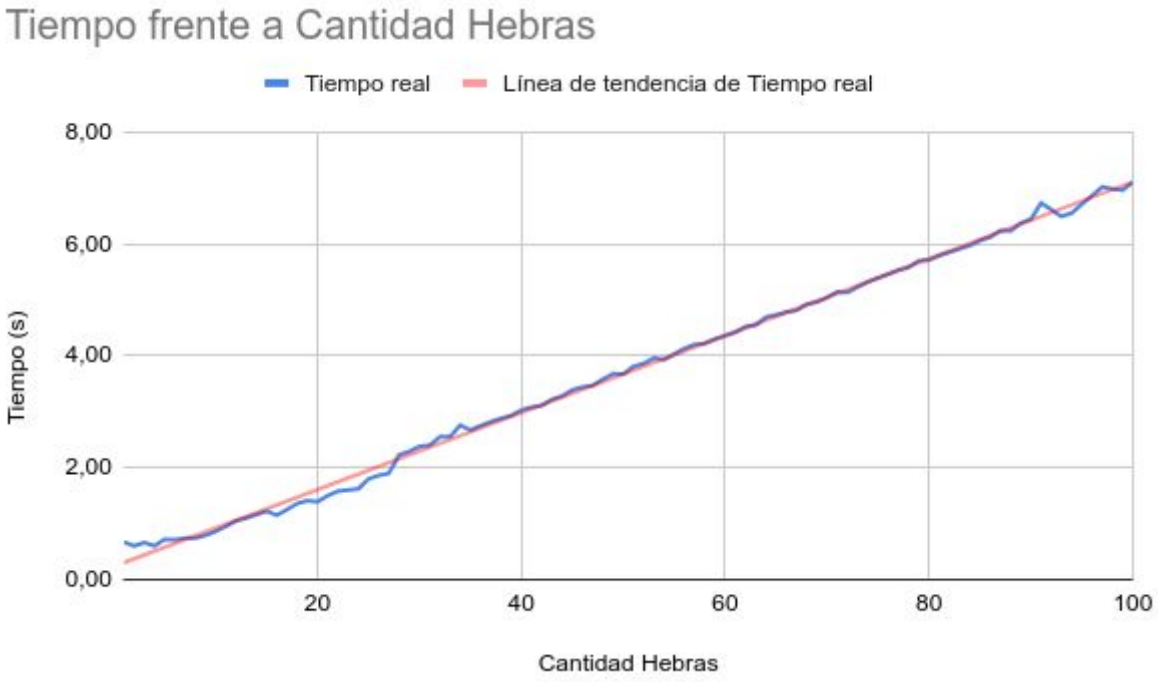

Gráfico 13: Tiempo(s) vs Cantidad de hebras. 4 Niveles de recursividad y 640000 Elementos

	Tiempo	N° Hebras
Mínimo	0,60	2
Máximo	7,10	88

Tabla 10: Mínimo y máximo tiempo conseguido. 4 Niveles de recursividad y 640000 Elementos

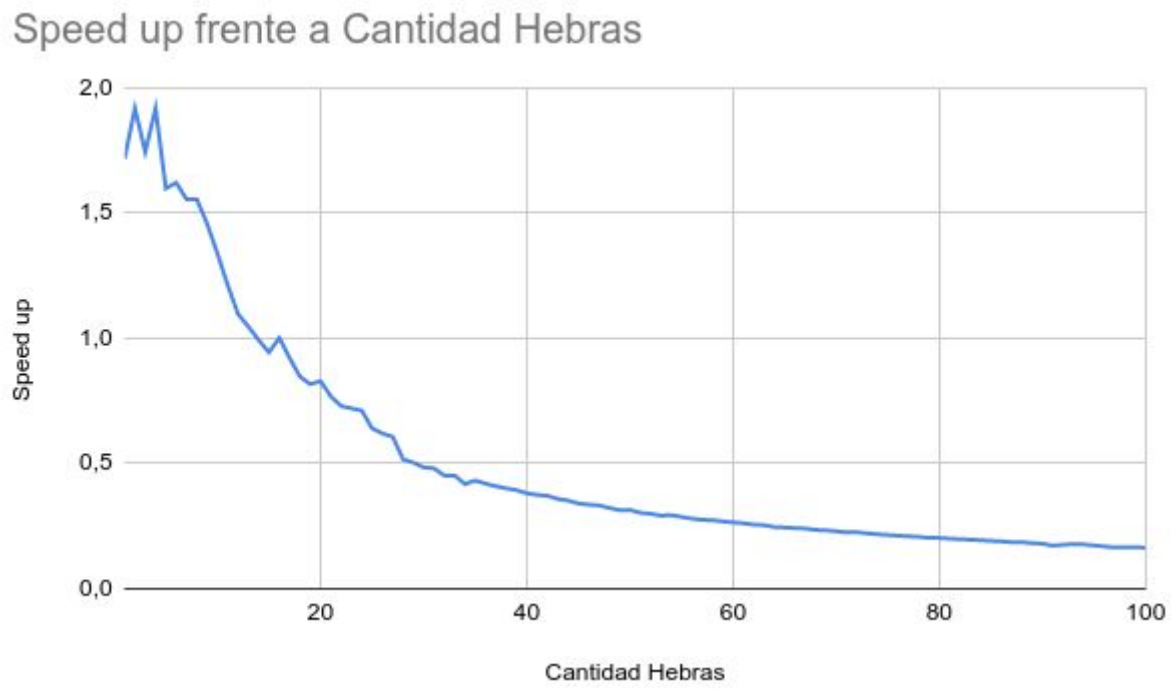

Gráfico 14: Speed up vs Cantidad de hebras. 4 Niveles de recursividad y 640000 Elementos

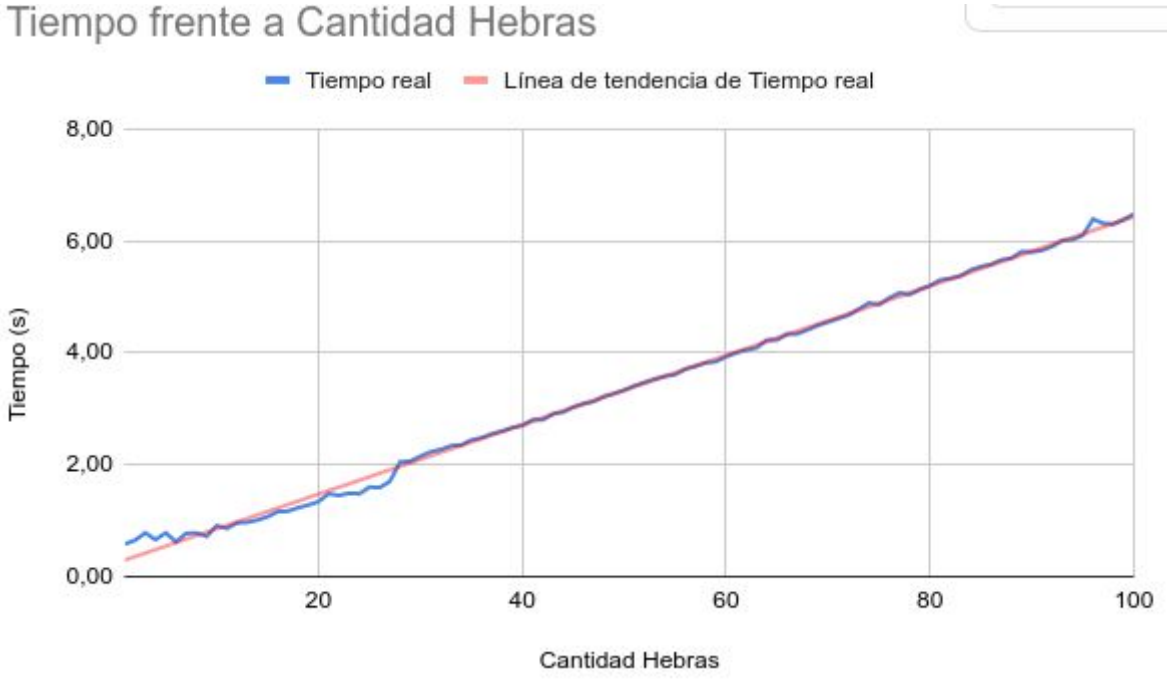

Gráfico 15: Tiempo(s) vs Cantidad de hebras. 4 Niveles de recursividad y 640000 Elementos

	Tiempo	N° Hebras
Mínimo	0,58	4,5
Máximo	6,48	100

Tabla 11: Mínimo y máximo tiempo conseguido. 5 Niveles de recursividad y 640000 Elementos

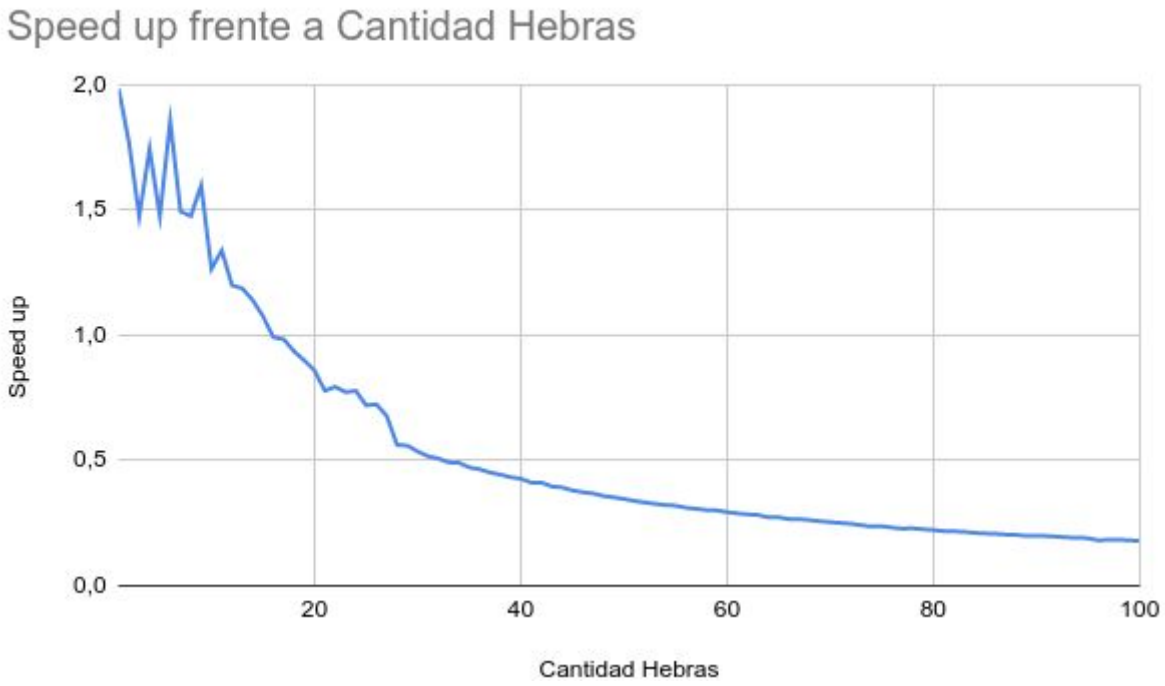

Gráfico 16: Tiempo(s) vs Cantidad de hebras. 5 Niveles de recursividad y 640000 Elementos

Conclusión

En el primer archivo de 6536 elementos el tiempo mínimo con dos niveles conseguido fue con 4,5 y 6 hebras con un tiempo de 0,05. En el gráfico 1 se puede visualizar que al aumentar la cantidad de hebras después de las 6 hebras aumenta el tiempo de ejecución del algoritmo en una manera lineal. En el gráfico 2 se puede observar que las mejores con respecto al laboratorio 1, en un inicio fueron crecientes y considerablemente mejores y luego tuvo un decaimiento exponencial. Con este mismo archivo con los niveles 3 el tiempo mínimo conseguido fue con 2 y 3 hebras, con 4 niveles fue conseguido con 2 y con 5 niveles fue conseguido con 4 y 5. De igual manera que el nivel 2, se puede visualizar que al aumentar la cantidad de hebra en un comiendo decae y luego aumenta de manera lineal. Con respecto al speed up, en el resto de niveles sucede exactamente lo mismo que con el anterior, donde primero aumenta considerablemente y luego tiene un decaimiento exponencial.

En el segundo archivo de 64 elementos solo se realizó la división en 2 niveles, debido a que utiliza SIMD con 16 elementos. Inicialmente se probaron desde 1 hasta 100 hebras, sin embargo no hubo mayores cambios en el tiempo de ejecución por lo que se decidió realizar 100 nuevas pruebas aumentando en 100 las hebras en cada iteración. El tiempo mínimo conseguido fue de 0 float y fue conseguido en 1, 101 y 1001 hebras. sin embargo en el gráfico 10 se puede observar que el rendimiento va disminuyendo de manera lineal.

Por último en el archivo de 640000 elementos se consiguió un tiempo de 0,73 utilizando 3 hebras y 2 niveles de recursión. El aumento de tiempo fue linealmente proporcional al aumento de hebras debido a que era necesario la sincronización de cada una de estas y considerar el cambio de contexto de entre estas. Con respecto al speed up conseguido tiene un decaimiento exponencial a medida que aumenta la cantidad de hebras. Estos comportamientos explicados anteriormente pasan de igual manera al 3,4 y 5 niveles.