Odd-Even Sort

Para paralelizar este algoritmo de ordenamiento se puede hacer de dos formas, la primera es usando un `omp for` y la otra forma es usando un `omp parallel for`, la diferencia es que el `for` se encarga de distribuir la tarea a realizar en los threads que sean indicados, pero no tiene la función de inicializar estos threads, por otro lado, el `omp parallel for`, tiene la función de inicializar los threads y después asignarles trabajo.

El objetivo del trabajo es poder detectar la diferencia en términos de rendimiento entre `omp parallel for` y `omp for`.

Las listas son generadas aleatoriamente y tienen una longitud de 20000.

1. **Parallel For**

   Los tiempos obtenidos son muy similares a los valores que aparecen en el libro, el primer obstáculo que se tiene es el hardware, la computadora en que se hicieron las pruebas tiene 4 núcleos, por lo que la cantidad máxima de threads es 4.

2. **For**

   Después de probar el código con omp parallel for, se prueba con omp for, de igual forma desde 1 hasta 4 threads (limitado por el hardware), en este caso también se mantiene cierta similitud con los resultados mostrados en el libro. Sin embargo en comparación con el omp parallel for, aquí se obtienen resultados ligeramente mejores.
   Esto se produce debido a que omp parallel for tiene que inicializar y asignar threads mientras que omp for únicamente hace una asignación, a esto se le añade que el proceso de reducción en omp for es mejor.

Multiplicación de Matrices

En este programa se aborda la multiplicación de matrices con vectores.

Threads	10	100	1000	10000
1	0.5983	0.9282	1.6709	2.3917
2	0.3484	0.7757	1.4788	2.0593
3	0.2667	0.5486	1.1922	1.7673
4	0.1553	0.3332	0.9431	1.4442

Regla Trapezoidal

En este programa se calcula el área usando trapecios como aproximación, la cantidad de trapecios calculados son las columnas, las filas son la cantidad de threads con los que se ejecutó el programa. Claramente se puede ver que a más trapecios por calcular aumenta el tiempo, sin embargo conforme se aumenta el número de threads, el tiempo disminuye.

Threads	10 Trapecios	100 Trapecios	1000 Trapecios	10000 Trapecios
1	3.3845	4.2244	5.0644	8.4241
2	2.7126	3.3840	4.0564	6.7442
3	2.4059	3.0192	3.6984	6.1036

Cálculo de Pi

Para calcular el número Pi, se utilizó la serie de Leibniz, las filas indican la cantidad de threads que se usaron y las columnas son la cantidad de pasos, fases, iteraciones.

Threads	50	100	250	500
1	0.8957	1.1085	1.3208	2.1712
2	0.7255	0.8955	1.0655	1.7455
3	0.6405	0.7895	0.9381	1.5123
4	0.5555	0.6835	0.8105	1.3205