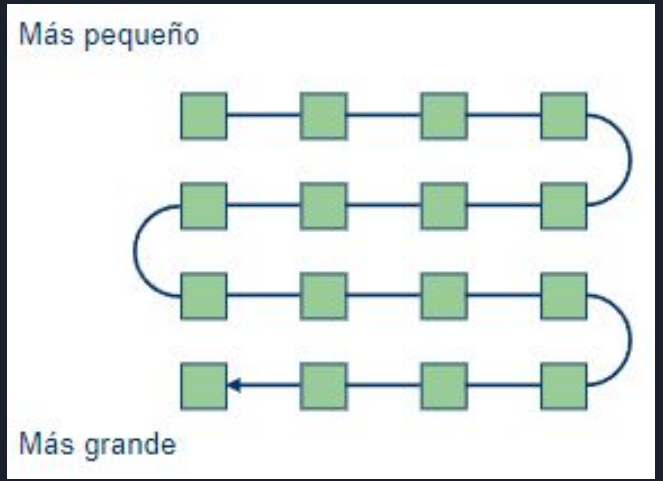
Algoritmos de ordenamiento paralelos

García Lopez Erik
Ramírez Medina Daniel

Snake sort (Shearsort)

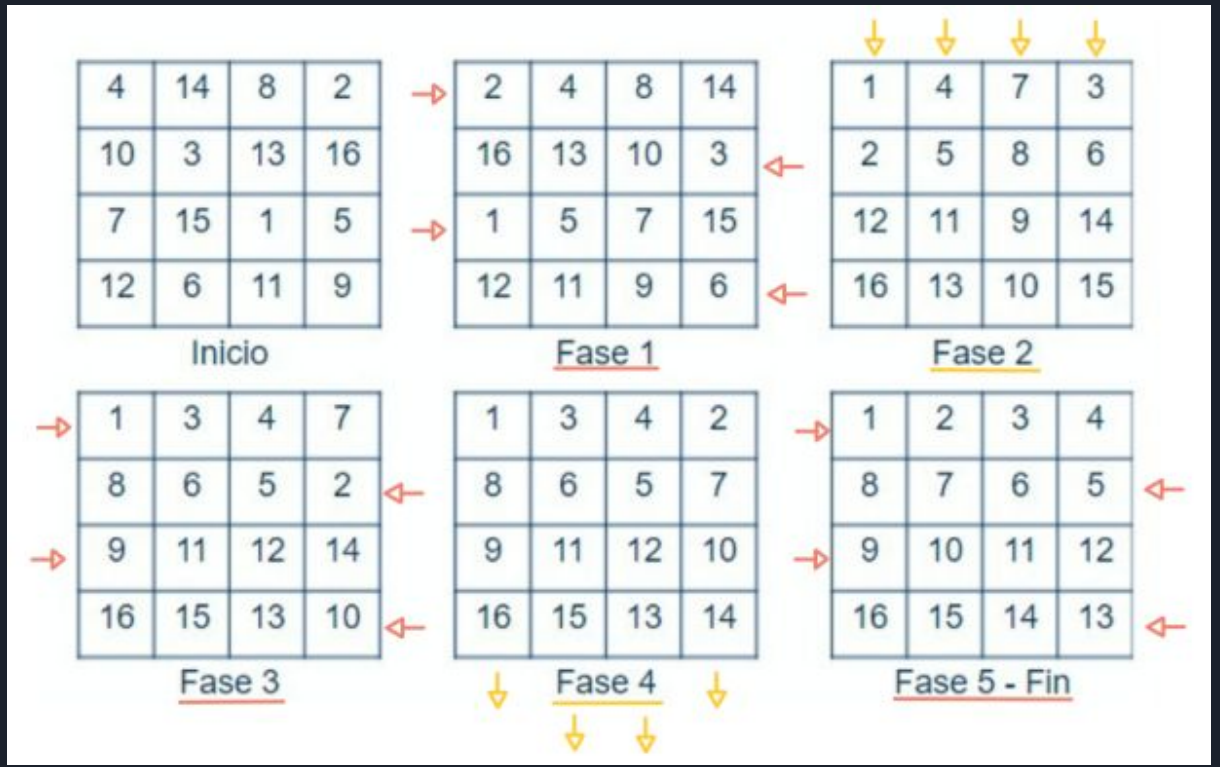
¿Qué es?

Es un algoritmo de ordenación donde los elementos de una matriz son ordenados en patrón de serpiente



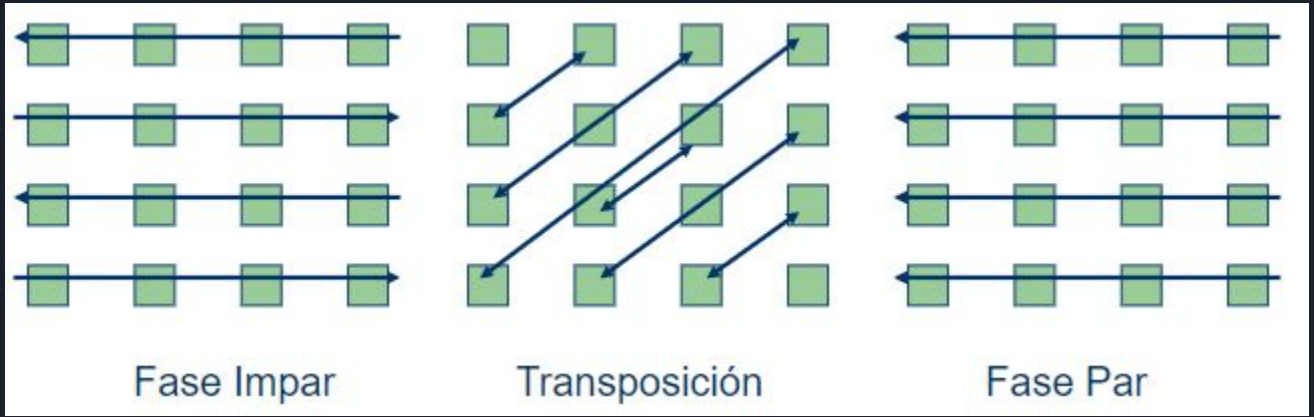
¿Cómo funciona la versión secuencial?

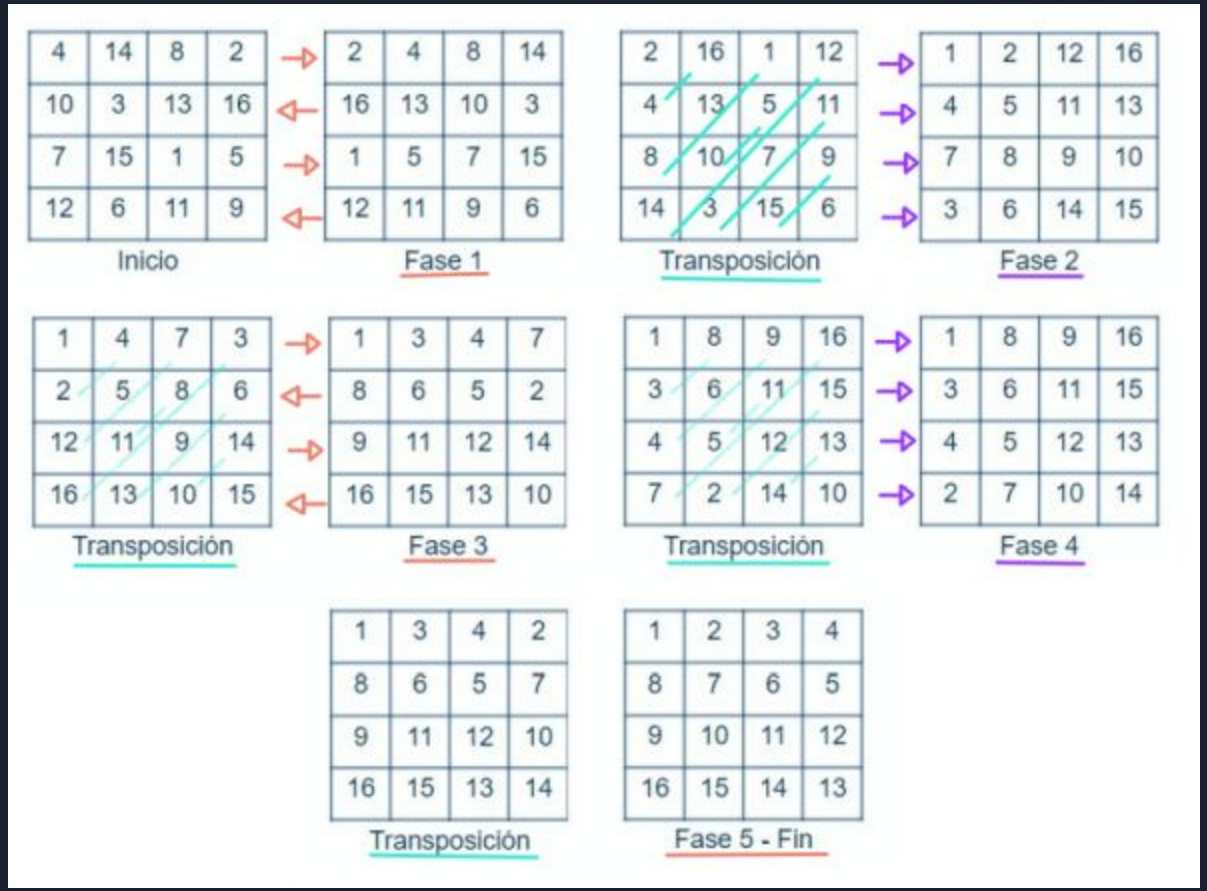
El algoritmo lo que hace es alternar entre ordenar las filas y ordenar las columnas



¿Cómo funciona la versión paralela?

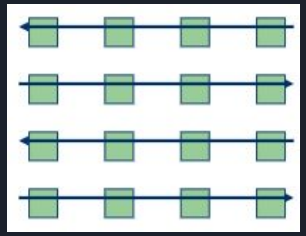
- Teniendo un procesador por fila podemos paralelizar las fases impares
- Usando transposición entre fases podemos paralelizar también las fases impares



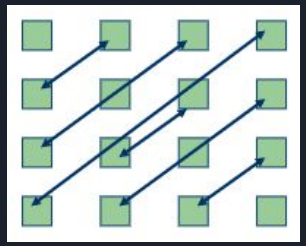

4 14 8 2
10 3 13 16
7 15 1 5
12 6 11 9
Inicio
2 4 8 14
16 13 10 3
1 5 7 15
12 11 9 6
Fase 1
2 16 1 12
4 13 5 11
8 10 7 9
14 3 15 6
Transposición
1 2 12 16
4 5 11 13
7 8 9 10
3 6 14 15
Fase 2
1 4 7 3
2 5 8 6
12 11 9 14
16 13 10 15
Transposición
1 3 4 7
8 6 5 2
9 11 12 14
16 15 13 10
Fase 3
1 8 9 16
3 6 11 15
4 5 12 13
7 2 14 10
Transposición
1 8 9 16
3 6 11 15
4 5 12 13
2 7 10 14
Fase 4
1 3 4 2
8 6 5 7
9 11 12 10
16 15 13 14
Transposición
1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
Fase 5 - Fin

Paralelización

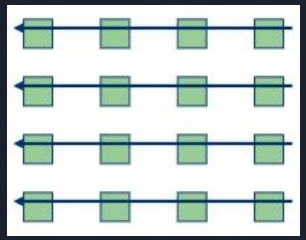
Fase impar: Cada procesador compara y ordena los elementos de su fila. Como cada procesador trabaja de forma independiente en su propia fila, esta fase se puede paralelizar sin necesidad de comunicación entre procesadores.



Fase de transposición: La matriz se transpone para permitir la comparación y ordenamiento de los elementos en las columnas. Para paralelizar esta fase, se pueden usar técnicas de transposición eficientes como la **transposición en bloques**, que permite la transposición paralela de submatrices.

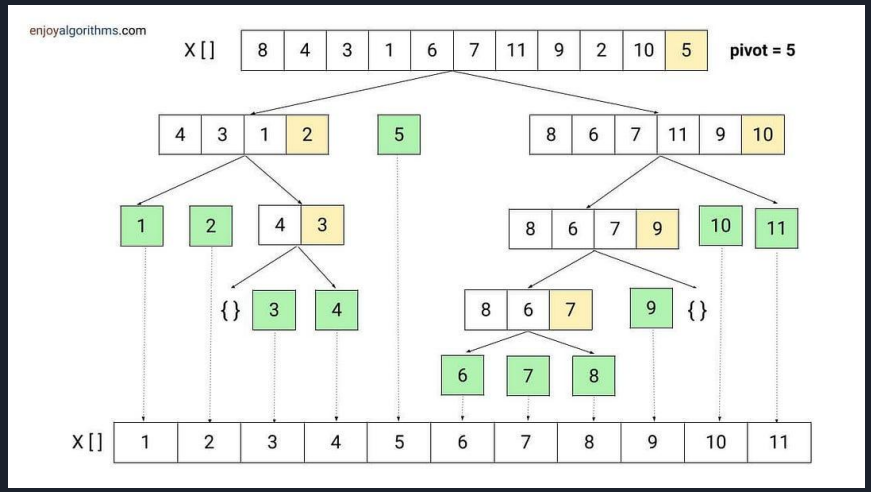


Fase par: los procesadores vuelven a comparar y ordenar los elementos de su fila. Como en la fase impar, esta fase se puede paralelizar sin necesidad de comunicación entre procesadores.



QuickSort

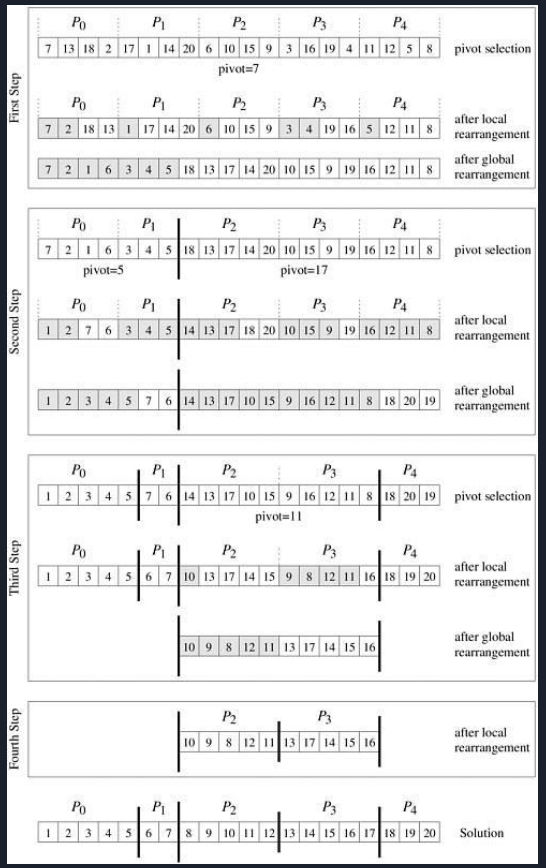
QuickSort es un algoritmo de ordenamiento basado en el algoritmo Divide y Vencerás que elige un elemento como pivote y divide el array quedando los valores menos de un lado y los mayores del otro del pivote.



Parallel Quick Sort

¿Qué es?

El Parallel Quick Sort u “Ordenamiento Rápido Paralelo”, es una variante del algoritmo de ordenamiento rápido (QS) que utiliza técnicas de paralelismo para mejorar el rendimiento en sistemas multi-core o distribuidos.

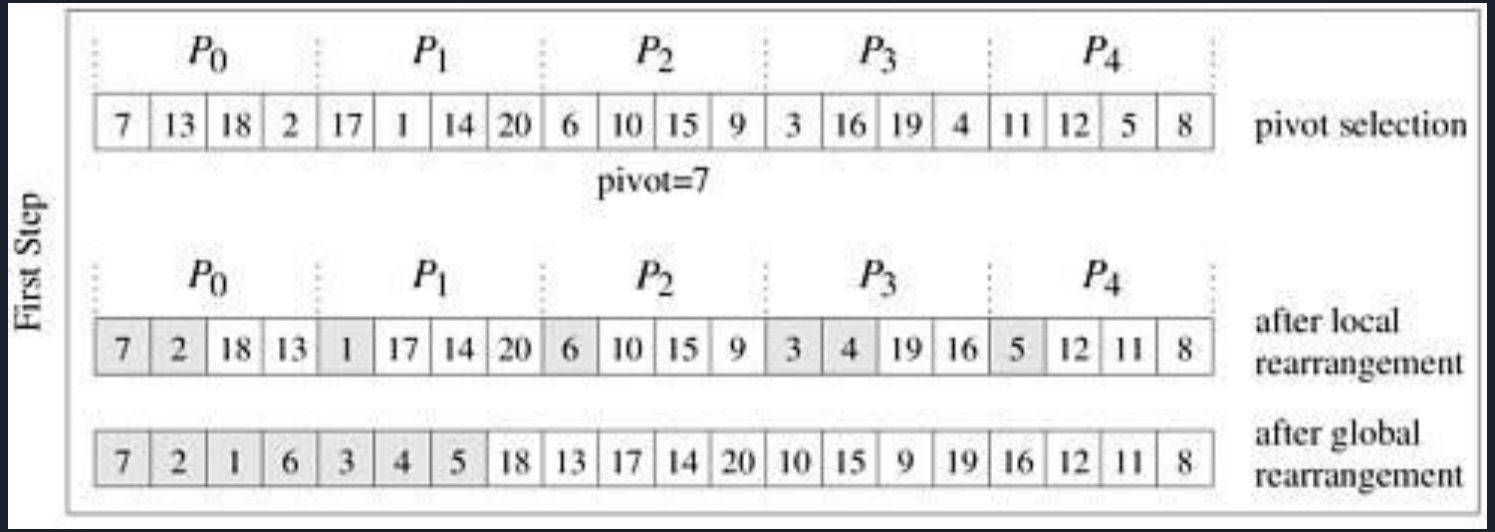


Algoritmo del Parallel Quick Sort

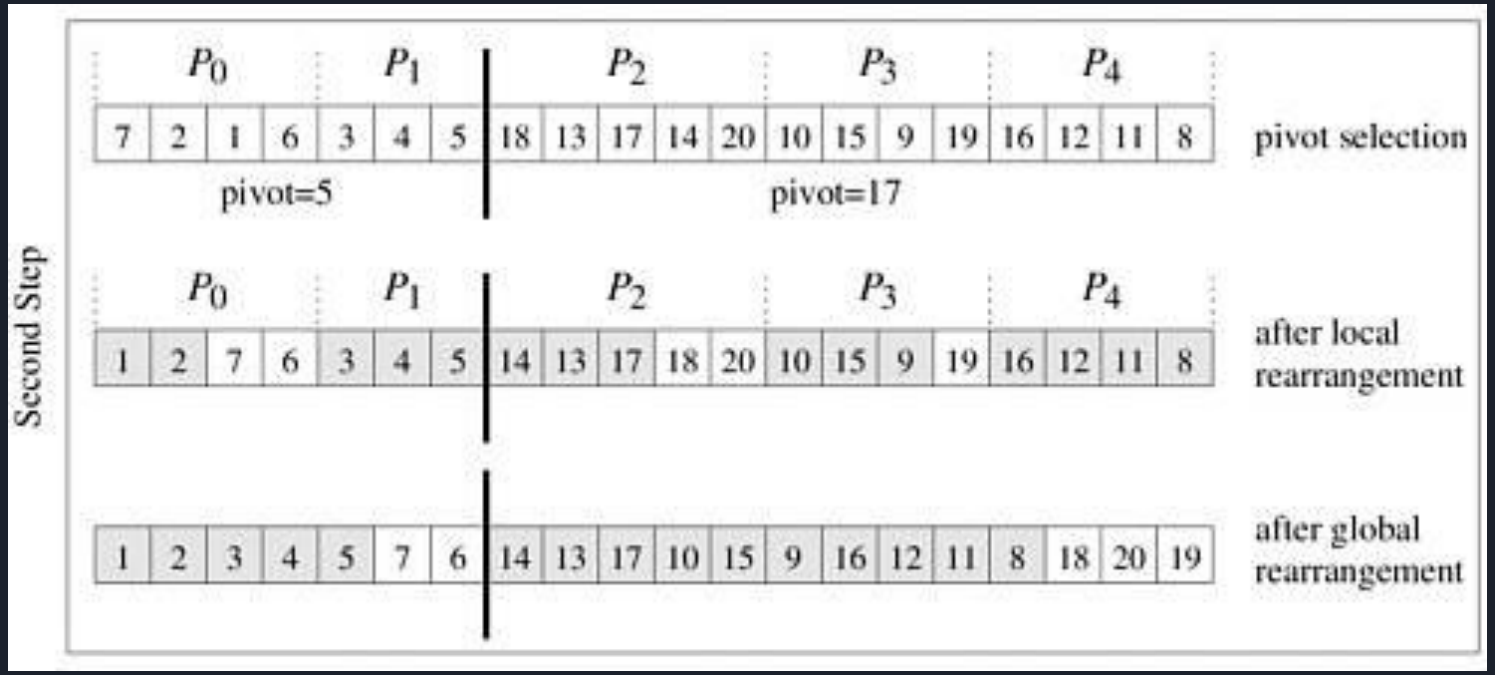
La clave para lograr el paralelismo en el Quick Sort es elegir un pivote adecuado y dividir la lista en subconjuntos que se puedan procesar de forma independiente. Una vez que se hayan ordenado los subconjuntos de manera individual, se puede realizar una etapa de fusión o combinación para obtener la lista finalmente ordenada.

Complejidad: O(n log n).

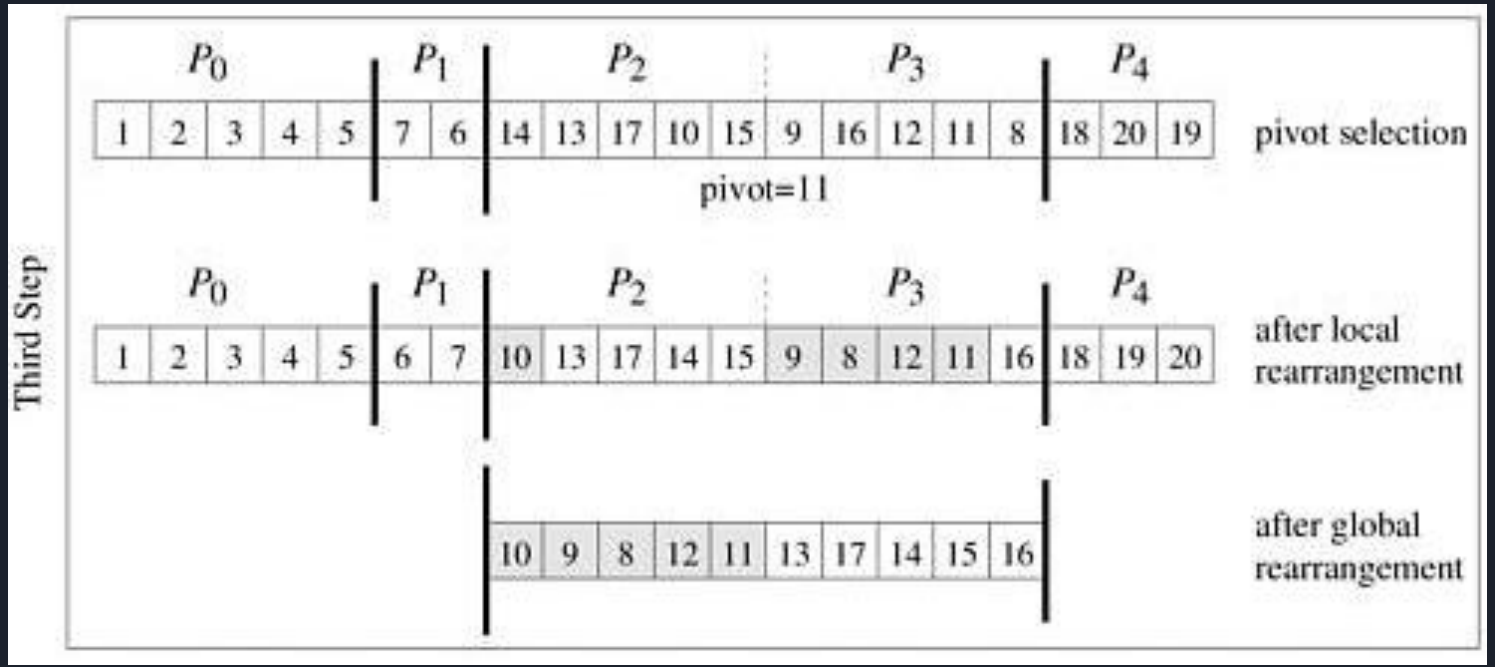
Algoritmo


First Step
P_0 P_1 P_2 P_3 P_4
7 13 18 2 17 1 14 20 6 10 15 9 3 16 19 4 11 12 5 8
pivot selection
pivot=7
P_0 P_1 P_2 P_3 P_4
7 2 18 13 1 17 14 20 6 10 15 9 3 4 19 16 5 12 11 8
after local rearrangement
7 2 1 6 3 4 5 18 13 17 14 20 10 15 9 19 16 12 11 8
after global rearrangement

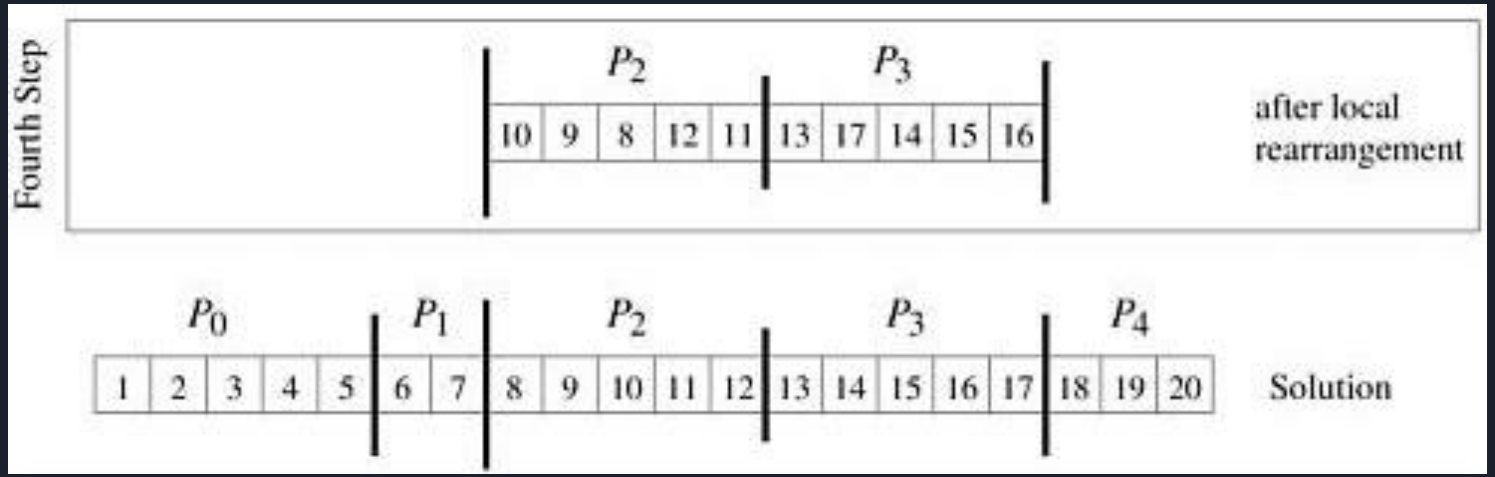
Algoritmo


Second Step
P_0 P_1 P_2 P_3 P_4
7 2 1 6 3 4 5 18 13 17 14 20 10 15 9 19 16 12 11 8
pivot selection
pivot=5
pivot=17
P_0 P_1 P_2 P_3 P_4
1 2 7 6 3 4 5 14 13 17 18 20 10 15 9 19 16 12 11 8
after local rearrangement
1 2 3 4 5 7 6 14 13 17 10 15 9 16 12 11 8 18 20 19
after global rearrangement

Algoritmo


Third Step
P_0
P_1
P_2
P_3
P_4
1 2 3 4 5 7 6 14 13 17 10 15 9 16 12 11 8 18 20 19
pivot selection
pivot=11
P_0
P_1
P_2
P_3
P_4
1 2 3 4 5 6 7 10 13 17 14 15 9 8 12 11 16 18 19 20
after local rearrangement
10 9 8 12 11 13 17 14 15 16
after global rearrangement

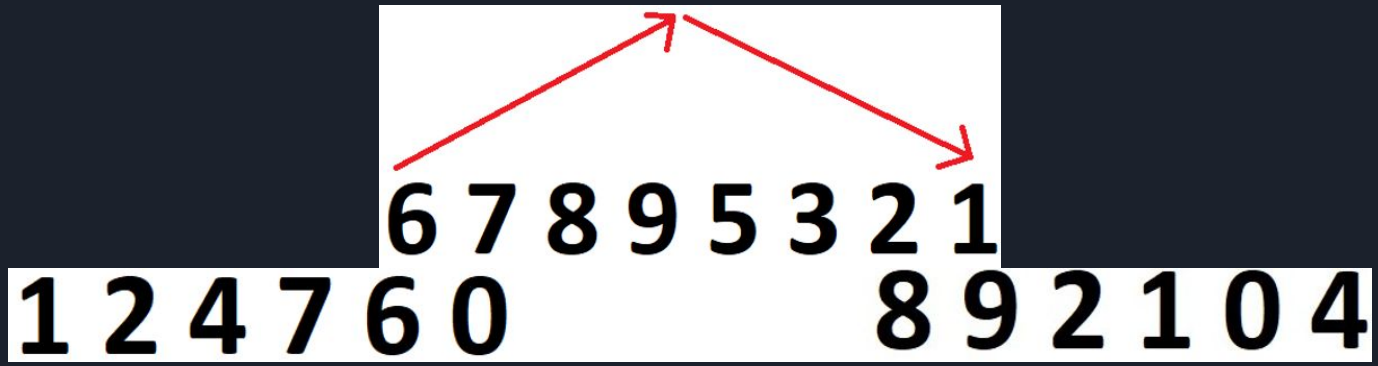
Algoritmo


Fourth Step
P_2
P_3
10 9 8 12 11 13 17 14 15 16
after local rearrangement
P_0
P_1
P_2
P_3
P_4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Solution

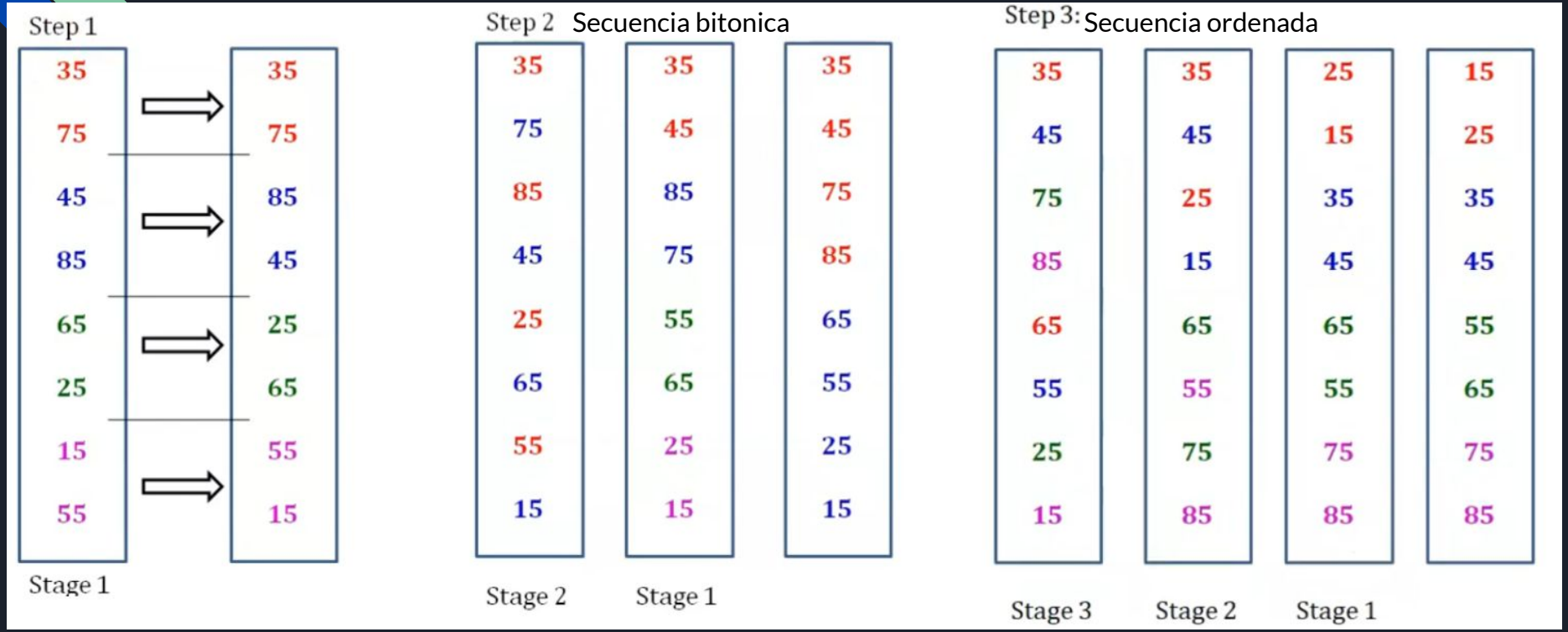
Bitonic Sort

¿Qué es una secuencia bitónica?

Una secuencia es llamada bitónica cuando los números primero aumentan y luego disminuyen o viceversa.

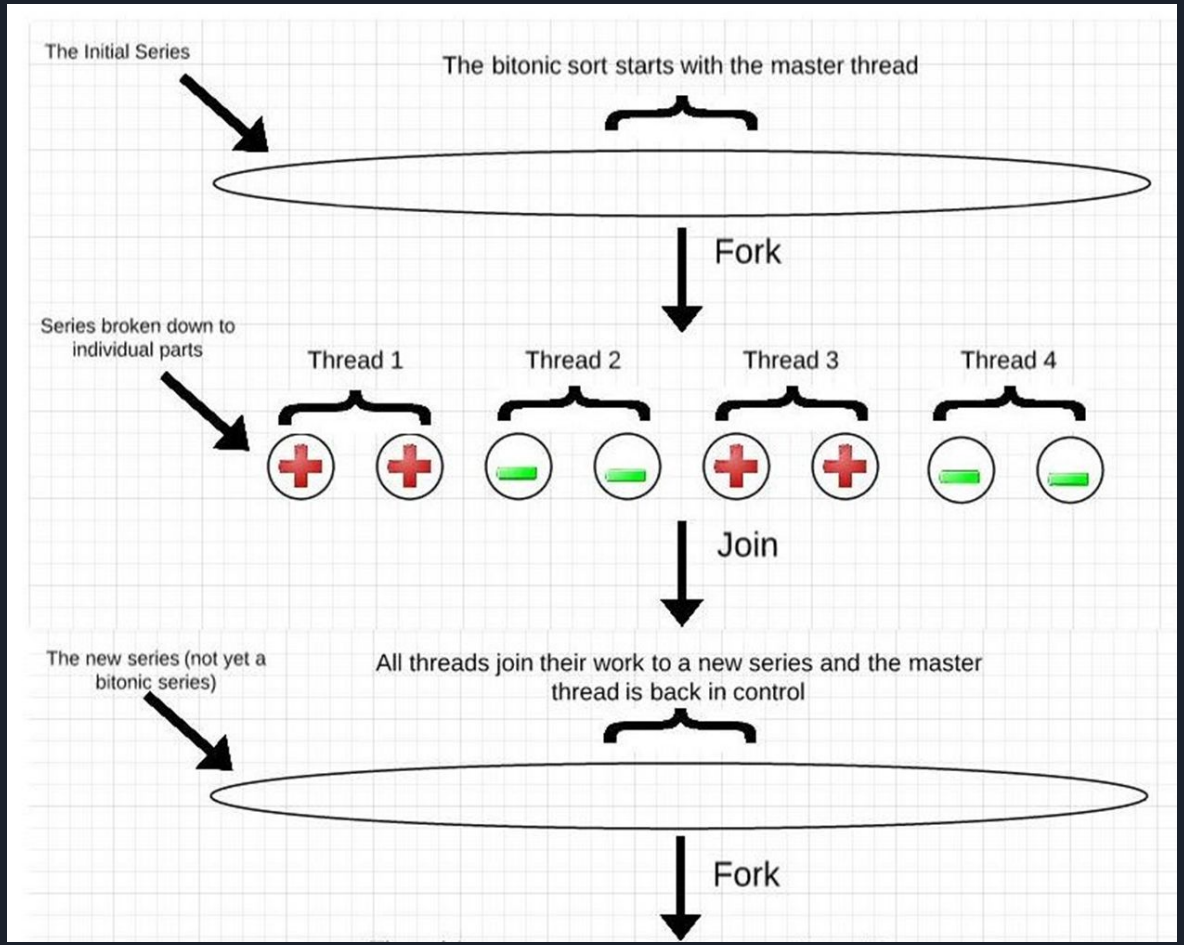


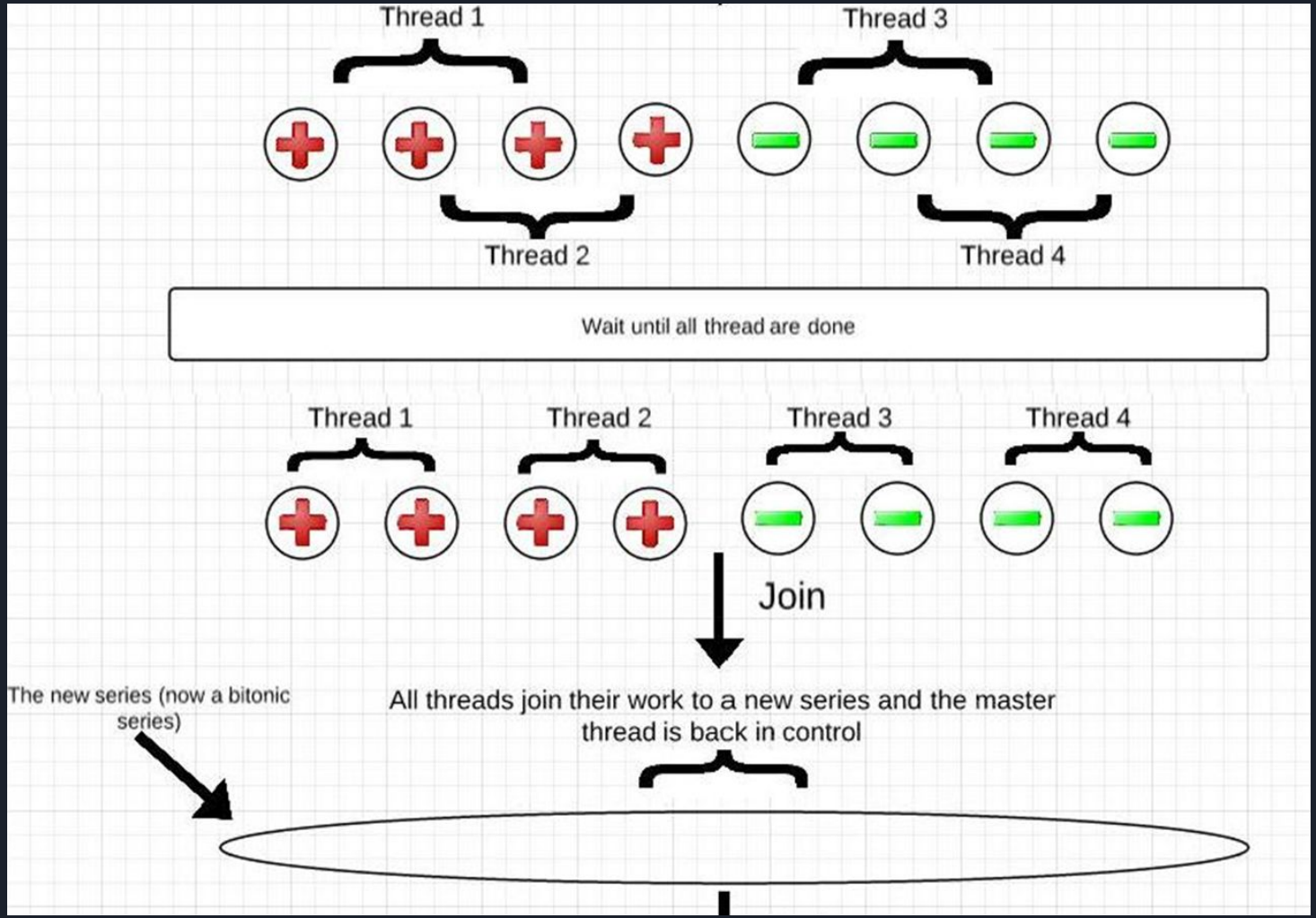
Es un algoritmo de ordenamiento utilizado para convertir una secuencia de números aleatorios a una secuencia bitónica y posteriormente en una secuencia de números ordenada

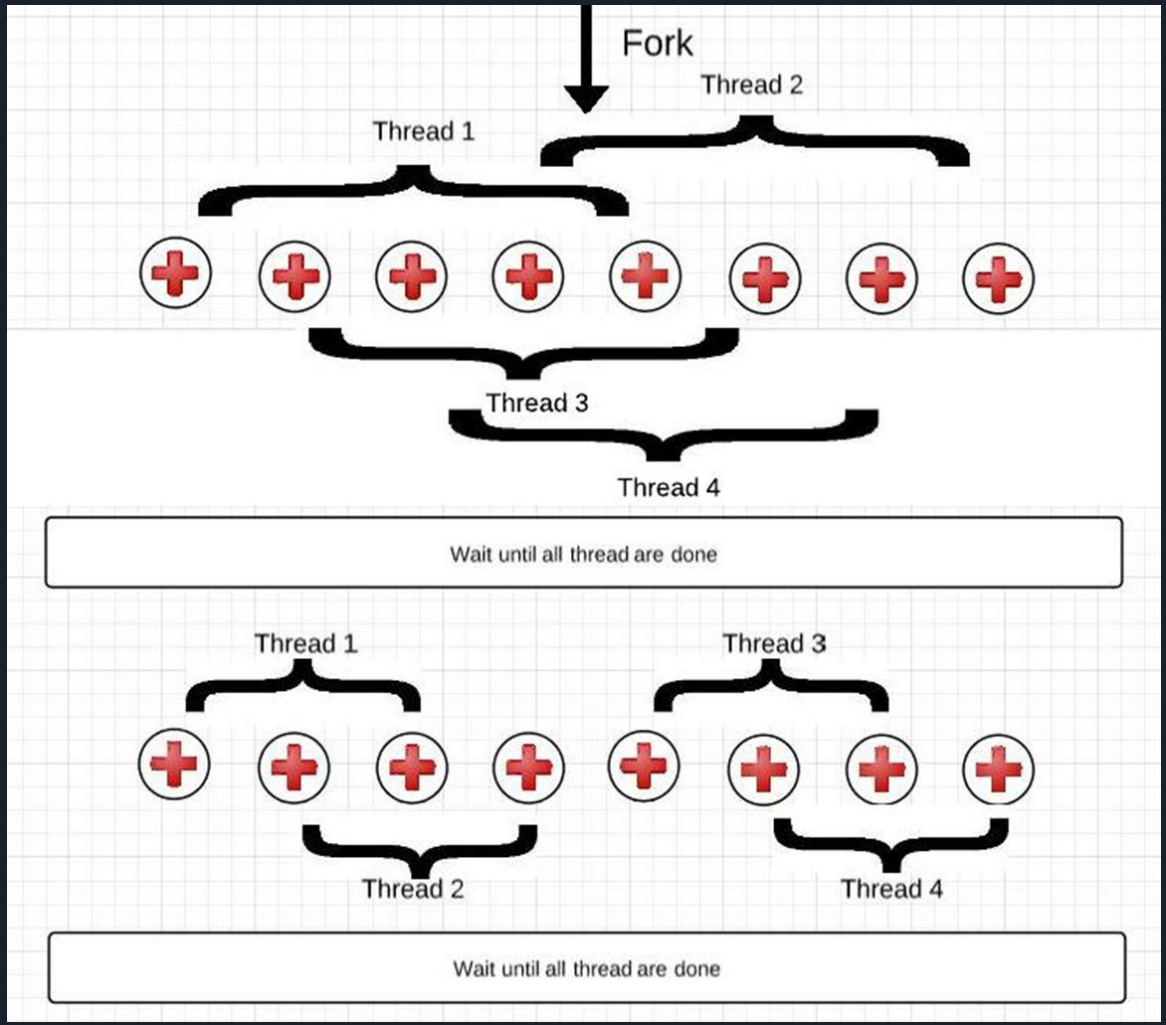

Step 1
35
75
45
85
65
25
15
55
35
75
85
45
25
65
55
15
Stage 1
Step 2
Secuencia bitonica
35
75
85
45
25
65
55
15
35
45
85
75
55
65
25
15
35
45
75
85
65
55
25
15
Stage 2
Stage 1
Step 3:
Secuencia ordenada
35
45
75
85
65
55
25
15
35
45
25
15
65
55
75
85
25
35
45
65
55
75
85
15
35
45
65
55
75
85
Stage 3
Stage 2
Stage 1

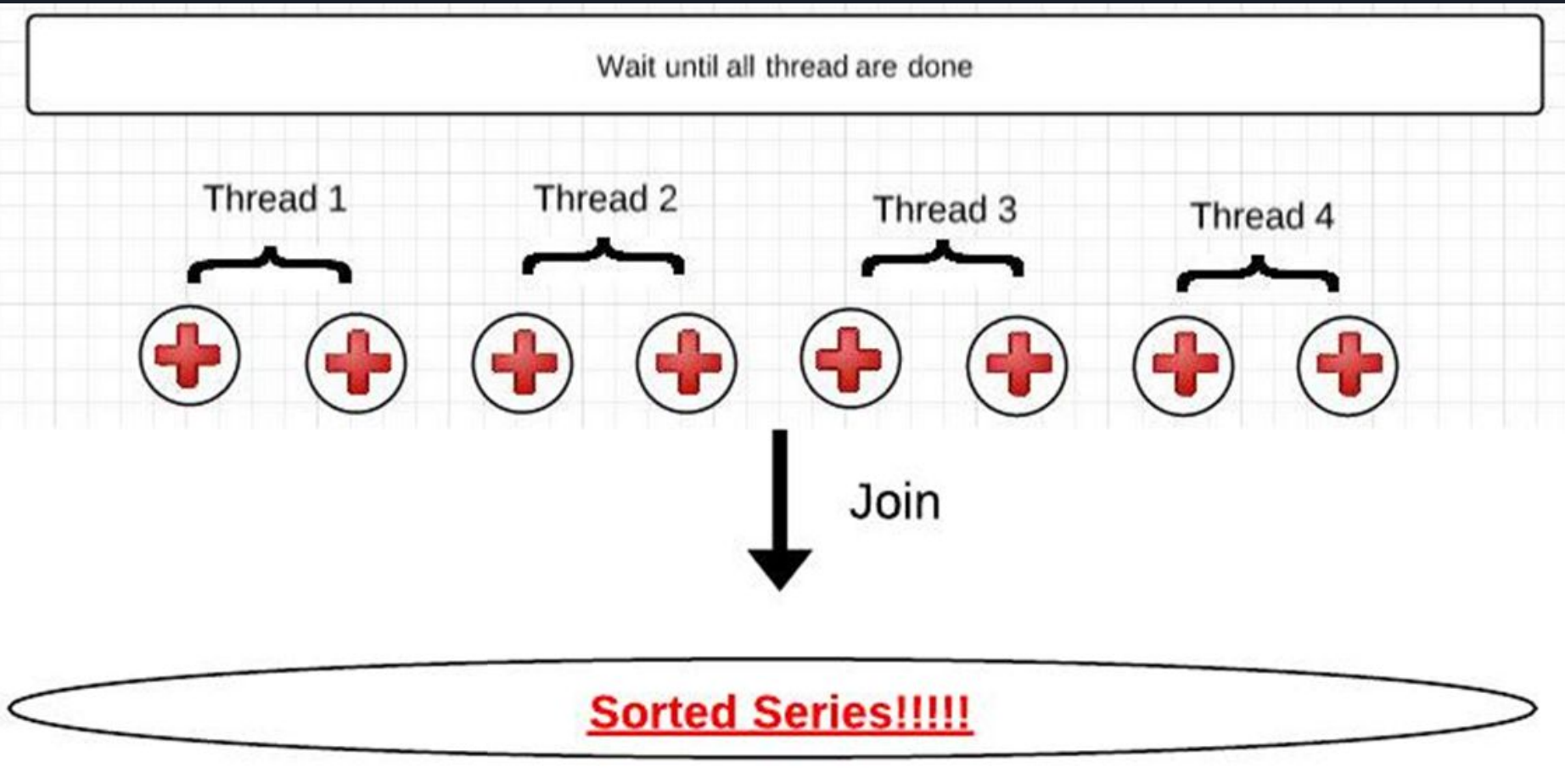
Ejemplo en paralelo: 8 números con 4 hilos

m = n.lenght() /numThreads




Thread 1
Thread 3
Thread 2
Thread 4
Wait until all thread are done
Thread 1
Thread 2
Thread 3
Thread 4
Join
The new series (now a bitonic series)
All threads join their work to a new series and the master thread is back in control


Fork
Thread 2
Thread 1
Thread 3
Thread 4
Wait until all thread are done
Thread 1
Thread 3
Thread 2
Thread 4
Wait until all thread are done


Wait until all thread are done
Thread 1
Thread 2
Thread 3
Thread 4
Join
Sorted Series!!!!!

¿Cuándo conviene paralelizar un algoritmo?

- Grandes conjuntos de datos
- Hardware con múltiples núcleos o procesadores
- Escalabilidad
- Restricciones de tiempo

En resumen, la paralelización de un algoritmo de ordenamiento puede mejorar la eficiencia y el rendimiento del proceso de ordenamiento, especialmente cuando se tienen grandes conjuntos de datos, hardware con múltiples núcleos o procesadores, y se espera una escalabilidad futura.

¿Cuál algoritmo es el mejor?

Es importante tener en cuenta que la eficiencia y el rendimiento de los algoritmos de ordenamiento paralelo pueden variar dependiendo de la arquitectura de hardware utilizada, el tamaño de los datos y otros factores específicos de la implementación. Además, el contexto y los requisitos particulares de tu aplicación también pueden influir en la elección del algoritmo más adecuado.

Bibliografía

SlidePlayer:
Autor: No se proporciona autor en la página.
Título del artículo: [Diapositiva con ID 3173643]
Recuperado de: https://slideplayer.es/slide/3173643/

OpenGenus IQ:
Autor: No se proporciona autor en la página.
Título del artículo: Parallel Merge Sort - OpenGenus IQ
Recuperado de: https://iq.opengenus.org/parallel-merge-sort/

YouTube:
Autor: No se proporciona autor en el enlace.
Título del video: Bitonic Sort Algorithm
Recuperado de: https://www.youtube.com/watch?v=32NLIL_6WJg

GeeksforGeeks:
Autor: No se proporciona autor en la página.
Título del artículo: Bitonic Sort
Recuperado de: https://www.geeksforgeeks.org/bitonic-sort/

Atw.hu:
Autor: No se proporciona autor en la página.
Título del artículo: Parallel Computing: Sorting
Recuperado de: http://users.atw.hu/parallelcomp/ch09lev1sec4.html

Algoritmos de ordenamiento paralelos

García Lopez Erik

Ramírez Medina Daniel