Título del informe

Nombre Apellido1 Apellido 2, email@tld.com

**Resumen:** Un solo párrafo de unas 200 palabras como máximo. El resumen debe ofrecer una descripción general pertinente del trabajo. Se recomiendo el siguiente estilo de resumen estructurado, pero sin encabezados: (1) Antecedentes: presente el trabajo abordado en un contexto amplio y resalte el propósito del estudio; (2) Métodos: describa brevemente los principales métodos utilizados; (3) Resultados: resumir los principales hallazgos del trabajo; (4) Conclusiones: indique las principales conclusiones o interpretaciones. El resumen debe ser una representación objetiva de su trabajo y no debe contener resultados que no estén presentados y fundamentados en el texto principal y no debe exagerar las conclusiones principales.

# Introducción

La programación paralela se puede llevar a cabo de muchas maneras, por ejemplo, con hilos, que estos están dentro de cada Core de un procesador, haciendo que cada hilo haga una tarea para luego unir los resultado, con Cores físicos, que es casi lo mismo que los threads y finalmente lo que utilizaremos ahora con intrínsecas del procesador, en este caso de Intel. Las intrínsecas si bien están consideradas dentro de la programación paralela, no lo son tanto como los threads o Cores de los procesadores, ya que no pueden hacer 2 tareas simultaneas, pero lo que si pueden hacer es realizar una operación sobre un conjunto de datos en 1 solo ciclo de procesador. Entonces por ejemplo antes hacer una suba de 8 números, que nos tomaría 5 ciclos de procesador, ahora con la intrínseca correcta nos puede tomar hasta 1 solo ciclo realizar esta suma, lo que se traduce en optimización de tiempo. Entonces el experimento que haremos será el de ordenar una cantidad de números utilizando estas intrínsecas para “optimizar” el tiempo de ordenamiento.

# Componentes del entorno de pruebas

Para la realización de este trabajo se utilizará el servidor facilitado por la universidad de Valparaíso, que tiene los siguientes componentes técnicos.

CPU 🡪 Intel ® Xeon ® Platinum 8260 CPU @ 2.4GHz

Sockets(s) 🡪 16

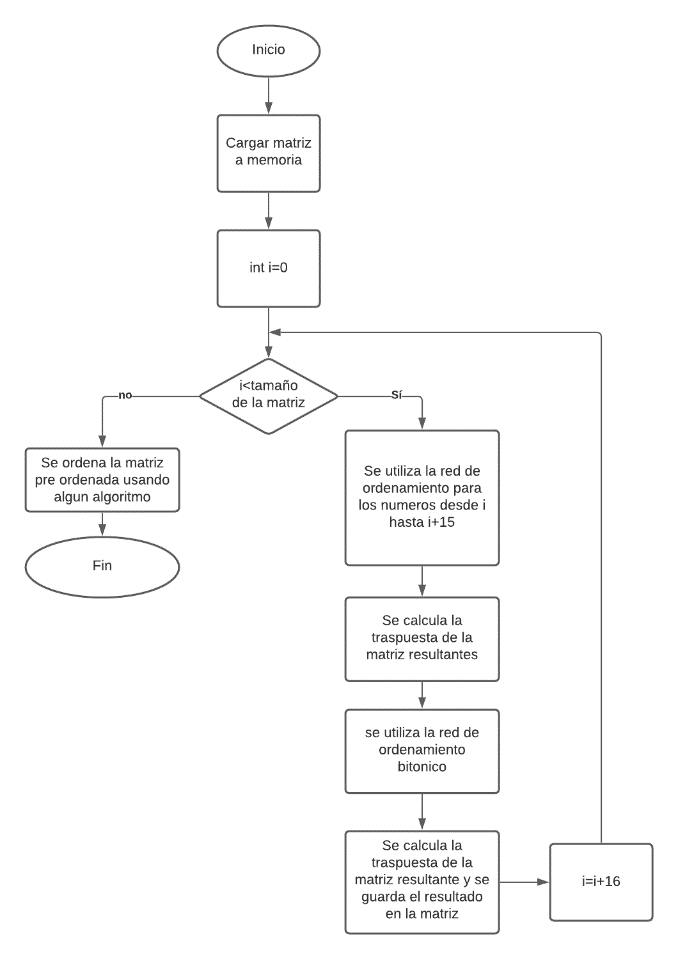
Core(s) por socket 🡪 1

Thread(s) por core 🡪 1

Arquitectura 🡪 x86\_64

# Desarrollo

Para la realización de este algoritmo se utilizaron redes de ordenamiento, ordenadores bitonicos, y redes de ordenamientos bitonico, de esta manera el arreglo total se puede preordenar en grupos de 16 números ordenados entre si, para aligerar la carga de la función que finalmente lo ordenara.



**Figura 1.** En este diagrama de flujo se puede ver el funcionamiento del algoritmo a alto nivel

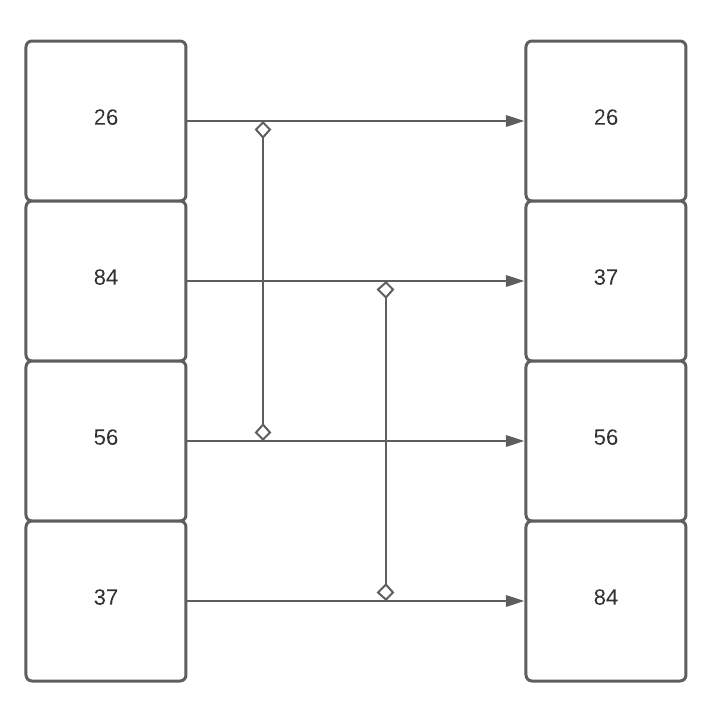
## Sorting network

Una red de ordenamiento es un conjunto de pasos para que un numero fijo de pasos y números sean ordenados. En este caso usaremos una red de ordenamiento de 4 números, pero al momento de implementarla para utilizar vectores ordenaremos 16 números en vez de solo 4, en la misma cantidad de ciclos que si fueran 4 números.

Para explicarlo mejor se utilizará un ejemplo.

### Paso 1

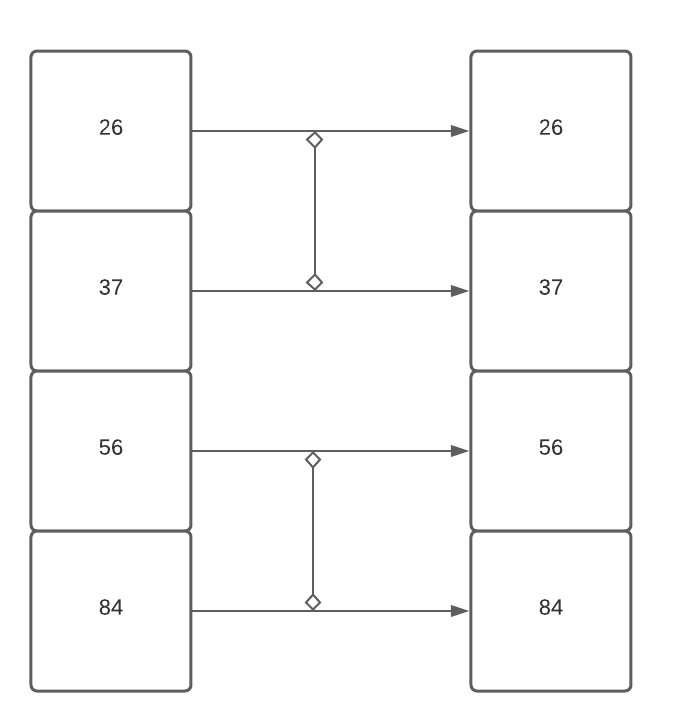
Se tienen 4 números {26,84,56,37} y el paso es comparar la posición 1 con la posición 3, y la posición 2 con la posición 4, con la operación máximo y mínimo, esto quiere decir que a un lado de la comparación siempre dejara el numero menor, y al otro lado siempre dejara el valor mayor, así como se puede observar en la figura a continuación.

****

**Figura 2.** Primer paso sorting network

### Paso 2

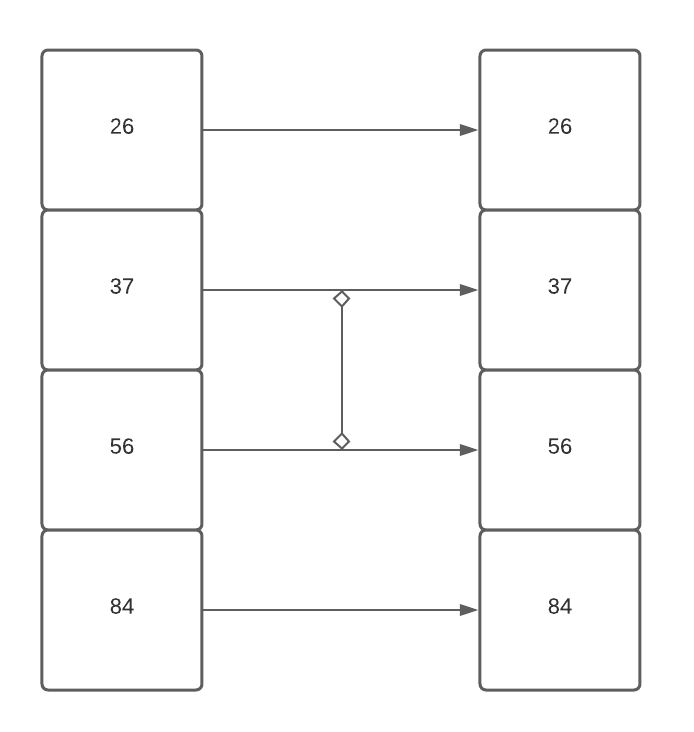
Luego con el resultado hacemos la operación máximo/mínimo entre las posiciones 1 y 2, y entre las posiciones 3 y 4 como se muestra a continuación.



**Figura 3.**  Segundo paso sorting network

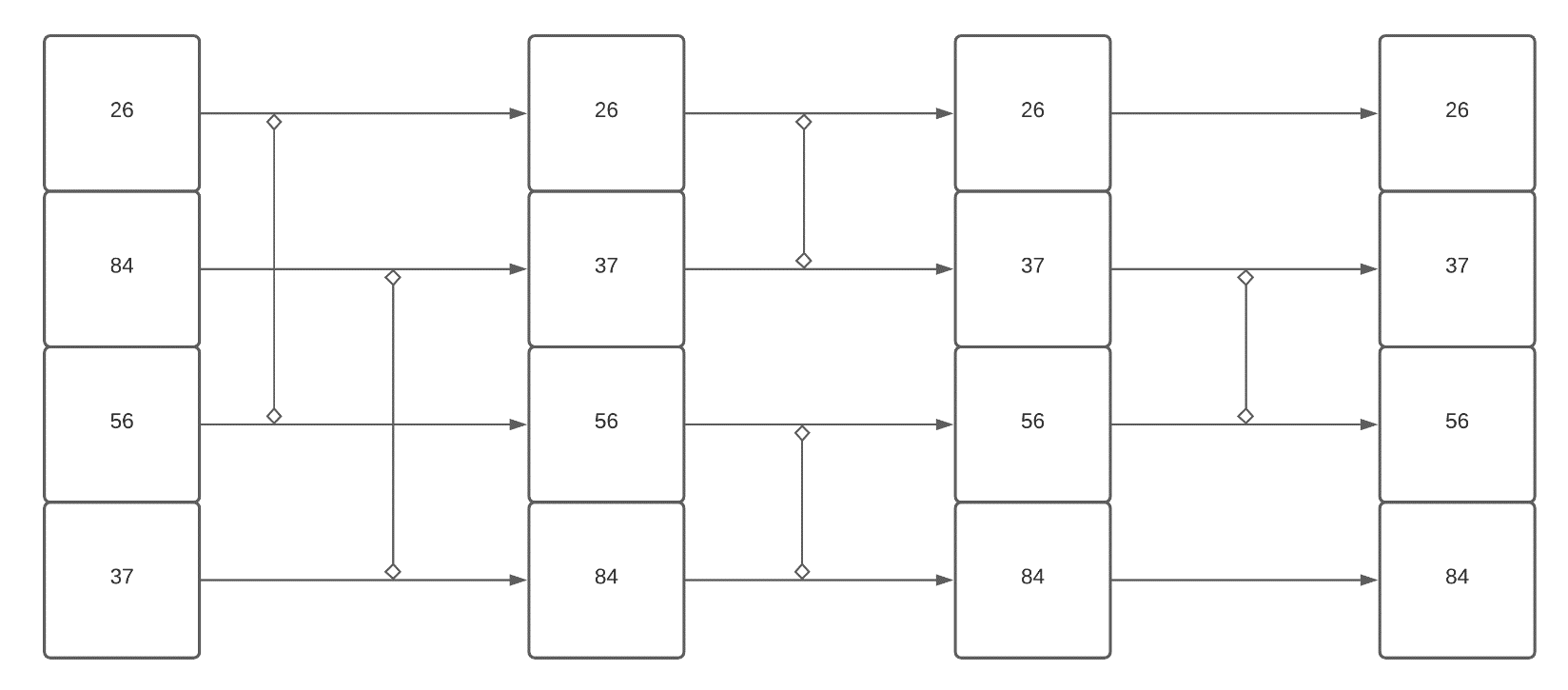
### Paso 3

Finalmente se hace la comparación con los números “del medio” ósea los que están en las posiciones 2 y 3 y con esto nuestro conjunto de 4 números quedaran ordenados.



**Figura 3.**  Tercer paso sorting network

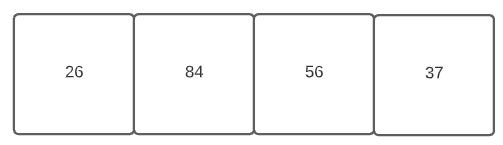
### Vista complete de la sorting network



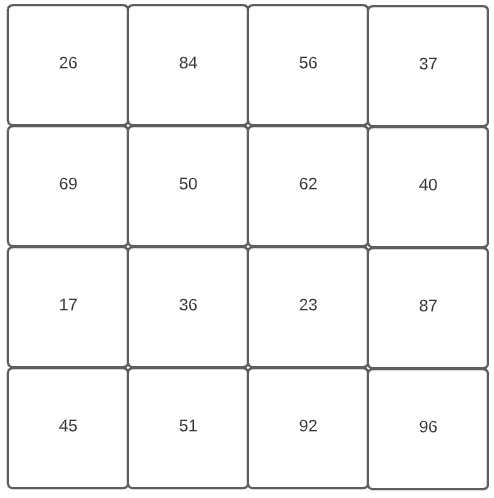
**Figura 3.**  sorting network completa

## Implementation vectorial sorting network

Para implementar esta sorting network con procesamiento vectorial deberíamos tener 1 numero en cada vector para poder hacer las comparaciones, ya que no podemos realizar operaciones sobre el mismo vector, entonces tenemos que tomar 4 vectores para poder realizar estas operaciones. Pero como estos son vectores de 128bits pueden almacenar hasta 4 números, por lo que ordenaremos 4 grupos de 4 números al mismo tiempo.

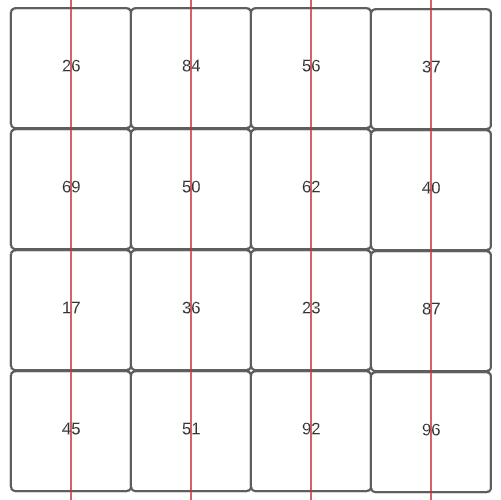


**Figura 3.**  Vector de ejemplo

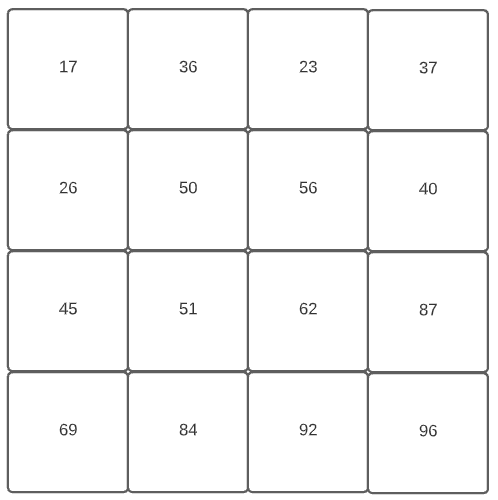


**Figura 3.**  4 vectores posicionados como una matriz 4x4

De esta manera al aplicar la sorting network de manera vectorial ordenaríamos columnas de números y no filas, pero lo haríamos en 5 operaciones siempre.



**Figura 3.**  Matriz con las columnas marcadas por líneas rojas



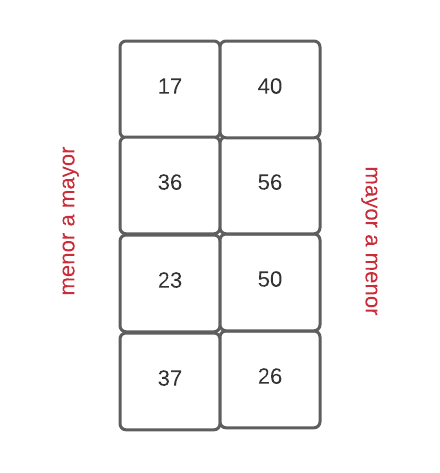
**Figura 3.**  Matriz después de aplicar la sorting network de manera vectorial

Ahora tenemos una matriz con solo las columnas ordenadas de menor a mayor en ellas, sin ninguna relación entre columnas.

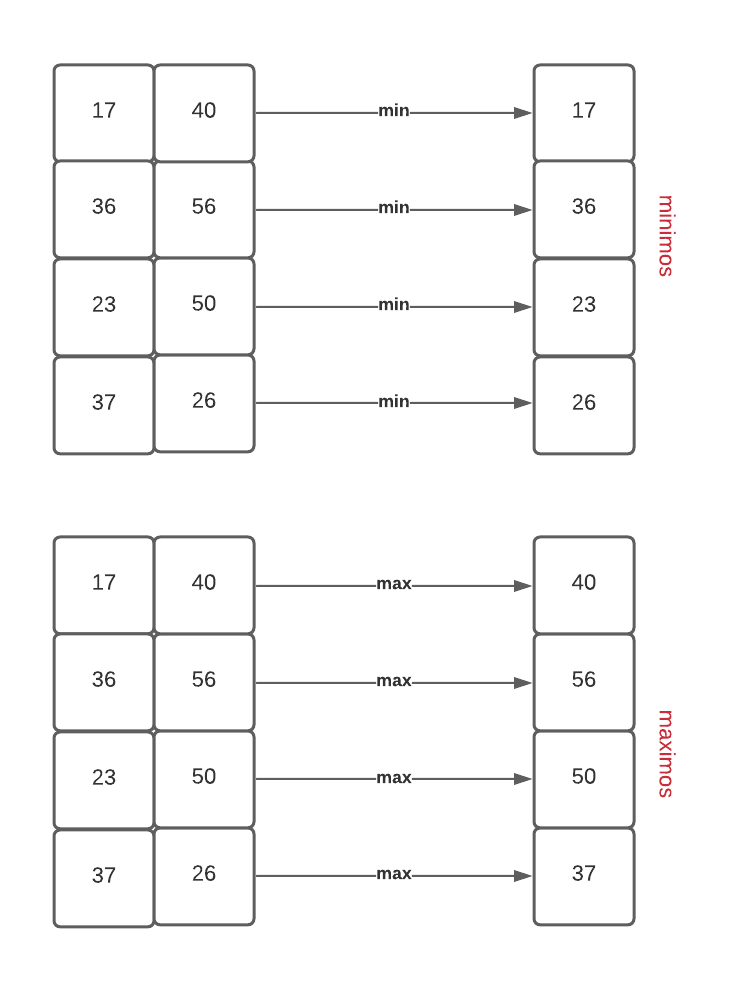
## Implementation de el bitonic sorter

### Implementación bitonic sorter paso 1

Un bitonic sorter ordenador bitonico es un algoritmo que es capaz de ordenar 8 números que estén preordenados, un grupo de 4 de menor a mayor y los otros 4 de mayor a menor. De esta manera este algoritmo será capaz de ordenar los 8 números siempre en la misma cantidad de pasos. Luego de tener los 2 vectores con 4 números cada uno ordenados de la forma correcta hacemos operaciones de máximo y mínimo sobre ambos vectores.



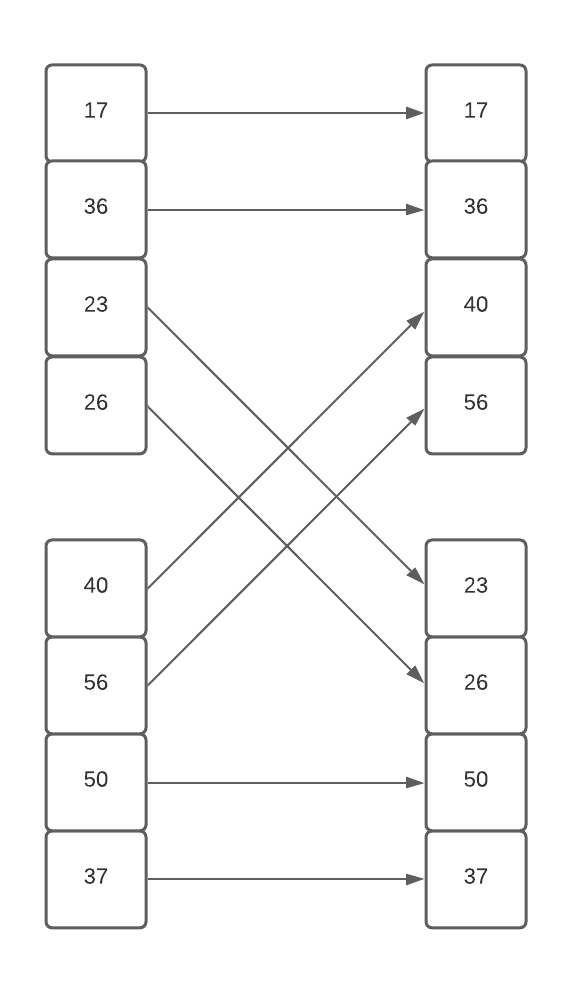
**Figura 3.**  Vectores ordenados



**Figura 3.**  Vectores después de la operación máximo y mínimo

### Implementación bitonic sorter paso 2

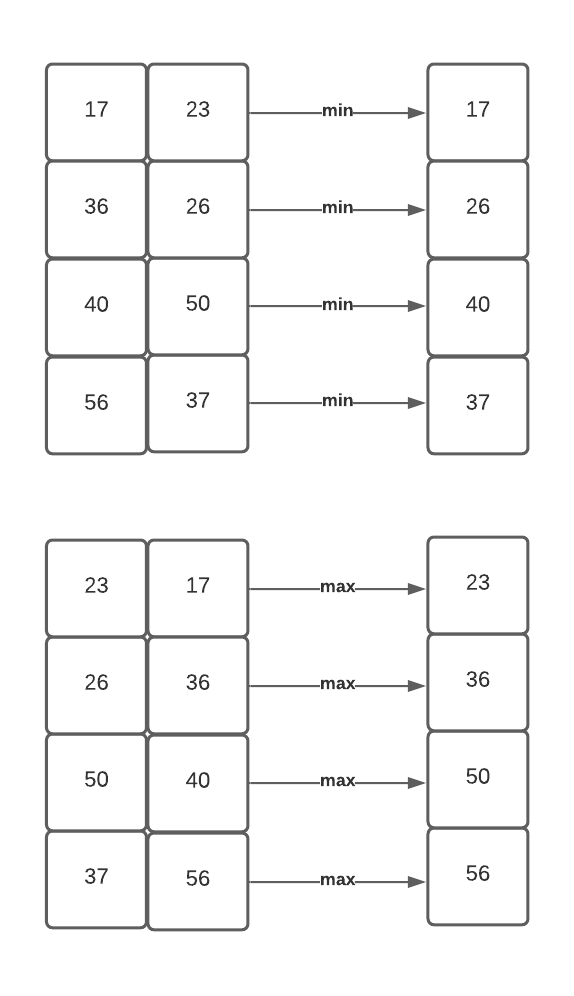
Luego de el primer paso tenemos que colocar los vectores alrevez y mover de posición los vectores, ya que necesitamos comparar valores correspondientes al mismo vector, y esto no lo podemos realizar, así que para trabajar estos vectores solo colocan los primeros 2 números de ambos vectores en un único vector y los últimos 2 números del otro vector en otro único vector.



**Figura 3.**  Imagen de la explicación del paso 2

### Implementación bitonic sorter paso 3

Luego aplicamos la operación máximo/mínimo sobre ambos vectores.



**Figura 3.**  Imagen de la explicación del paso 3

### Implementación bitonic sorter paso 4

Y luego repetimos lo hecho en el paso 2 y 3