# Método de la Ingeniería

**Aplicación para la Solución de un Problema**

**Miguel Angel Romero Rosas y Jonathan Arenas**

# Contexto Problemático

Una empresa de fabricación de microprocesadores está evaluando la posibilidad de implementar varios algoritmos de ordenamiento como instrucciones básicas de su próximo coprocesador matemático. la empresa ha decidido implementar tres (3) algoritmos diferentes de ordenamiento que permitan ordenar, muy rápidamente, números enteros de tamaño arbitrariamente grande y números en formato de coma flotante de cualquier tamaño.

# Desarrollo de la Solución

# Paso 1. Identificación del Problema

## Identificación de necesidades y síntomas

* Una empresa de fabricación de microprocesadores desea implementar tres algoritmos de ordenamientos como una operación nativa de su próximo coprocesador.
* La solución al problema debe garantizar un ordenamiento muy rápidamente de números enteros de tamaño arbitrariamente grande y números en formato de coma flotante de cualquier tamaño.
* La solución del problema debe de ser eficiente ya que esto evitaría que el procesador principal tenga que realizar estas tareas de cómputo intensivo.
* El coprocesador matemático que se quiere implementar debe acelerar el rendimiento del Sistema por el hecho de esta descarga de trabajo en el procesador principal.
* Una empresa de fabricación de microprocesadores desea implementar tres algoritmos de ordenamientos como una operación nativa de su próximo coprocesador.
* Sus anteriores líneas de coprocesadores no implementaban o no soportaban varios algoritmos de ordenamiento.
* los algoritmos deben permitir ordenar muy rápidamente.
* La empresa desea un software para probar los algoritmos qué serán implementados en el hardware.
* No disponen de un software que permita evaluar la eficiencia y eficacia de los algoritmos en el hardware
* La empresa desea poder ingresar los valores que desean ordenar.
* La empresa desea que el programa genere aleatoriamente los valores.
* La empresa desea tener control sobre la cantidad total de valores generados y el intervalo en el cual se generarán los mismos.
* La empresa, además, requiere poder elegir entre números repetidos o ausencia de estos en los números generados.
* La generación aleatoria debe permitir elegir entre unas configuraciones de los valores en la secuencia.
* La empresa requiere que el software ordene los valores ingresados utilizando el algoritmo apropiado de acuerdo con el tipo de número a ordenar y el intervalo de números generados.
* Finalmente, La empresa requiere que la interfaz gráfica muestre el tiempo que se demoró el algoritmo en ordenar las entradas.

## Definición del Problema

Una empresa que fabrica microprocesadores desea incluir en su próximo coprocesador la funcionalidad de ordenar entradas numéricas grandes, ya sean enteros o de coma flotante. Para esto, requieren incluir tres algoritmos de ordenamiento en el hardware que tengan un buen rendimiento dependiendo el tipo de número y la magnitud de números de la entrada. Además, Desean un software que permita evaluar estos algoritmos en el hardware. Así, la empresa requiere que el software lea y genere entradas numéricas enteras y de coma flotante. Para el caso de la generación, el software, precisa brindar control sobre el número total de datos, el intervalo de generación, existencia de números repetidos y elección entre posibles configuraciones de los valores en la secuencia. Finalmente, ordenar la serie de datos y mostrarlos junto al tiempo empleado.

## **Paso 2. Requerimientos Funcionales:**

* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda ingresar los valores que se desean ordenar.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda configurar el tipo de dato a generar.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda configurar la cantidad total de números a generar.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda configurar el intervalo en el cual se generarán los números.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda decidir la ausencia o existencia de números repetidos.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda elegir entre las siguientes configuraciones de los valores en la secuencia: (a)valores ya ordenados (b)valores ordenados inversamente (c)valores en orden completamente aleatorio (d)valores desordenados en un % dado por el usuario.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda generar aleatoriamente valores.
* la interfaz gráfica debe dar al modelo del mundo los valores que se desean generar junto con la configuración deseada para la serie de datos de salida.
* El modelo del mundo debe generar la serie de datos con las especificaciones brindadas por la interfaz del usuario.
* el modelo del mundo debe brindar a la interfaz del usuario la serie de datos generados.
* La interfaz gráfica debe ofrecer una opción mediante la cual el usuario pueda ordenar los valores.
* La interfaz gráfica debe dar al modelo del mundo los valores que se desean ordenar.
* El modelo del mundo debe brindar al programa 3 algoritmos de ordenamiento para entradas grandes enteras y de coma flotante.
* El modelo del mundo debe elegir un algoritmo mediante el cual, dependiendo de la entrada, se llevará a cabo la operación de ordenar.
* El modelo del mundo debe brindar a la interfaz gráfica los valores de la secuencia.
* El modelo del mundo debe brindar a la interfaz gráfica el tiempo empleado en el proceso de ordenamiento.
* La interfaz gráfica debe mostrar al usuario la serie de datos ordenados.
* La interfaz gráfica debe mostrar al usuario el tiempo empleado en el proceso de ordenar.

## 

# Paso 3. Recopilación de Información

## Definiciones

**Números en formato coma flotante:**

es una forma de notación científica usada en los microprocesadores con la cual se pueden representar números racionales extremadamente grandes y pequeños de una manera muy eficiente y compacta, y con la que se pueden realizar operaciones aritméticas.

Sin embargo, hay que tener en cuenta que no se puede evaluar si un coma flotante es mayor, o menor, que otro de la misma forma que se haría con un número entero. Pues, podemos tener A= 1,0 y B=1,0 pero puede que B= 1,00000001.

Cada dato float cuesta 4 bytes.

También, tienen un estándar que convierte los datos de coma flotante en bytes, se llama IEEE 754.

Para comprobar números flotantes no se evalúa si son iguales. Por el contrario, se busca evaluar si su diferencia es muy pequeña. El margen de error que compara esta diferencia normalmente se llama épsilon. En su forma más simple:

if(Math.abs(a-b) < 0.00001)

Pero como épsilon puede ser, o muy grande para floats pequeños o muy pequeño para floats muy grandes, es necesario ver si el error relativo es menor que épsilon:

if(Math.abs((a-b)/b) < 0.00001)

Hay casos especiales donde esto falla:

* Cuando tanto a como b son cero. 0.0/0.0 es NaN, lo que provoca una excepción en algunas plataformas o devuelve falso para todas las comparaciones.
* Cuando solo b es cero, la división devuelve infinito, lo que también puede causar una excepción, o es mayor que épsilon incluso cuando a es más pequeño.
* Devuelve «falso» cuando tanto a como b son muy pequeños, pero a ambos lados del cero, incluso cuando son los números no nulos más pequeños.
* Además, el resultado no es conmutativo (nearlyEquals(a,b) no es siempre lo mismo que nearlyEquals(b,a)). Para solucionar estos problemas, el código tiene que ser mucho más complejo, así que necesitamos meterlo en una función:
* public static boolean nearlyEqual(float a, float b, float epsilon)
* {
* final float absA = Math.abs(a);
* final float absB = Math.abs(b);
* final float diff = Math.abs(a - b);
* if (a == b) { // Atajo, maneja los infinitos
* return true;
* } else if (a \* b == 0) { // a o b o ambos son cero
* // El error relativo no es importante aquí
* return diff < (epsilon \* epsilon);
* } else { // Usar el error relativo
* return diff / (absA + absB) < epsilon;
* }
* }

Este método [pasa las pruebas](http://puntoflotante.org/errors/NearlyEqualsTest.java) para muchos casos especiales importantes, pero como puedes ver, utiliza cierta lógica no trivial. En particular, tiene que utilizar una definición totalmente distinta del margen de error cuando a o b son cero, porque la definición clásica del error relativo es inútil en esos casos.

Hay algunos casos en los que el método de arriba todavía produce resultados inesperados (concretamente, es mucho más estricto cuando un valor es casi cero que cuando es exactamente cero), y algunas de esas pruebas para las que fue desarrollado probablemente especifica un comportamiento que no es apropiado para algunas aplicaciones.

## Comparando valores de punto flotante como enteros

Hay una alternativa a aplicar toda esta complejidad conceptual a una tarea aparentemente tan sencilla: en lugar de comparar a y b como [números reales](http://es.wikipedia.org/wiki/N%C3%BAmero_real), podemos concebirlos como pasos discretos y definir el margen de error como el número máximo de valores de punto flotante posibles entre esos dos números.

Esto es conceptualmente muy evidente y fácil y tiene la ventaja de que escala implícitamente el margen de error relativo con la magnitud de los valores. Técnicamente es un poco más complejo, pero no mucho más de lo que puedas pensar, porque los números de punto flotante del IEEE 754 están diseñados para mantener su orden cuando sus secuencias de bits se interpretan como enteros.

Sin embargo, este método requiere que el lenguaje de programación soporte conversión entre valores de punto flotante y secuencias de bits enteras.

**Coprocesador:** es un microprocesador de un ordenador utilizado como suplemento de las funciones del procesador principal (la CPU). Las operaciones ejecutadas por uno de estos coprocesadores pueden ser operaciones de aritmética en coma flotante, procesamiento gráfico, procesamiento de señales, procesado de texto, criptografía, etc.

# Operaciones con aritmética:

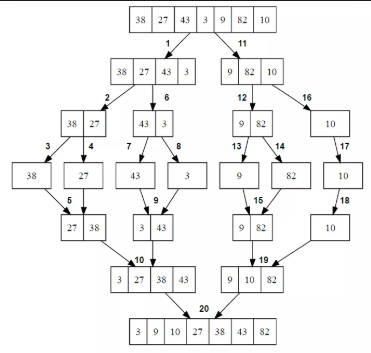
son: suma, resta, multiplicación, división, potenciación, división entera. Los operadores lógicos son: "y", "o", "no" y "o exclusivo".

**Numero entero:**

es un elemento del conjunto numérico que contiene los números naturales N ={1,2,3,4….}, sus opuestos y el cero.1​ Los enteros negativos, como −1 o −3 (se leen «menos uno», «menos tres», etc.), son menores que cero y todos los enteros positivos.

Alternativas:

**Alternativa 1: Método Merge Sort:**

 El algoritmo de ordenación por combinación, Merge Sort, se basa en la técnica Divide y Vencerás, ordena recursivamente un conjunto de elementos dividiéndolo en dos, ordenando cada una de estas partes en forma independiente y combinando los dos resultados. Este a su vez **recibe como entrada un arreglo de números enteros** denominado v, lo parte utilizando el **método copyOfRange** de la clase en java, se llama recursivamente con cada una de las dos partes como argumento y, una vez terminada la ordenación de dichas partes, invoca al proceso de combinación de las dos respuestas implementado en el método combinar, el cual recibe como entrada el arreglo original y las dos mitades de este previamente ordenadas que serán combinadas en el arreglo original. Este método es muy eficiente, comparado con otros métodos de ordenación, **pero solo es para números enteros.**

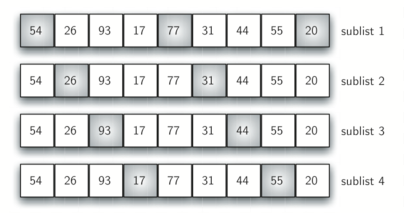
**Alternativa 2: Método Quick Sort:**

A close up of a clock

Description generated with low confidence

El objetivo de este algoritmo es dividir recursivamente el vector en partes iguales, indicando un elemento de inicio, fin y un pivote, que nos permitirá segmentar nuestra lista. Una vez dividida, lo que hace, es dejar todos los mayores que el pivote a su derecha y todos los menores a su izq. Al finalizar el algoritmo, nuestros elementos están ordenados. Esto lo que nos va a permitir en el programa es ordenar de una manera más eficiente y eficaz los números que lleguen como entrada al algoritmo.

**Alternativa 3: Método Shell Sort**

Es una mejora del método de inserción directa, utilizado cuando el array tiene un gran número de elementos.

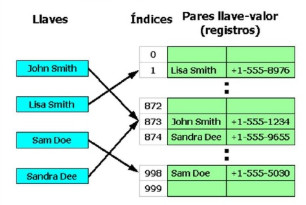
Cualquier algoritmo de ordenación que intercambia elementos adyacentes (como los algoritmos burbuja, selección o inserción) tiene un tiempo promedio de ejecución de orden cuadrático (n2). El método Shell mejora este tiempo comparando cada elemento con el que está a un cierto número de posiciones llamado salto, en lugar de compararlo con el que está justo a su lado. Este salto es constante, y su valor inicial es N/2 (siendo N el número de elementos, y siendo división entera).

**Alternativa 4: Arboles (Estructuras de Datos)**

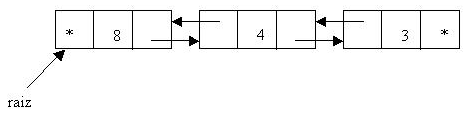


A los arboles ordenados de grado dos se les conoce como arboles binarios ya que cada nodo del árbol no tendrá más de dos descendientes directos, los cuales se pueden llamar izquierda o derecha. Las aplicaciones de los arboles binarios son muy variadas ya que se les puede utilizar para representar una estructura en la cual es posible tomar decisiones con dos opciones en distintos puntos. Esta estructura de datos nos va a facilitar un poco a la hora de introducir los datos al Sistema, ya que nos estará organizando, tanto los enteros como los números de coma flotante en una jerarquía de un árbol.

**Alternativa 5: Tabla Hash**

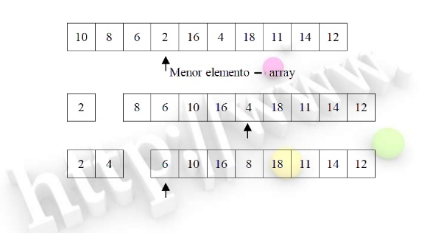
Comparada con otras estructuras de arrays asociadas, las tablas hash son más útiles cuando se almacenan grandes cantidades de información. Las tablas hash almacenan la información en posiciones pseudo-aleatorias, así que el acceso ordenado a su contenido es bastante lento. Otras estructuras como árboles binarios auto-balanceables tienen un tiempo promedio de búsqueda mayor (tiempo de búsqueda O(log n)), pero la información está ordenada en todo momento.

**Alternativa 6: Listas Doblemente Enlazadas**



Observemos que una lista doblemente encadenada tiene dos punteros por cada nodo, uno apunta al nodo siguiente y otro al nodo anterior. Seguimos teniendo un puntero (raíz) que tiene la dirección del primer nodo. El puntero sig del último nodo igual que las listas simplemente encadenadas apunta a null, y el puntero ant del primer nodo apunta a null. Se pueden plantear Listas tipo pila, cola y genéricas con enlace doble. Hay que tener en cuenta que el requerimiento de memoria es mayor en las listas doblemente encadenadas ya que tenemos dos punteros por nodo.

**Alternativa 7: Método de ordenamiento por Selección:**

La ordenación por selección funciona seleccionando el menor elemento de la matriz y llevándolo al principio; a continuación, selecciona el siguiente menor y lo pone en la segunda posición de la matriz y así sucesivamente. Usando el método por selección podremos ordenar los elementos enteros y de coma flotante de una manera más eficaz.

**Paso 4. Elementos relacionados entre las Alternativas creativas escogidas**

Una de las relaciones que Podemos encontrar entre las soluciones creativas es que todas tratan de organizar números. Si bien, cada uno tienen una manera más eficiente para resolver el problema en el peor de los casos, todas ellas están pensadas para satisfacer la necesidad de ordenar una entrada de números. Las practicas que se realizan en cada una varían demasiado entre cada alternativa. Pero algo en común es que se pueden aplicar a arreglos de forma lineal. La otra característica es la facilidad en implementación algorítmica: Las soluciones encontradas son compatibles con las operaciones aritméticas básicas de un equipo de cómputo moderno, hacienda el problema no complejo de tratar, ya que las soluciones escogidas son posibles en todos sus sentidos.

**Paso 5. Fuentes:**

[**http://puntocomnoesunlenguaje.blogspot.com/2014/09/metodo-shell-de-ordenacion.html**](http://puntocomnoesunlenguaje.blogspot.com/2014/09/metodo-shell-de-ordenacion.html)

[**https://jarroba.com/tablas-hash-o-tabla-de-dispersion/**](https://jarroba.com/tablas-hash-o-tabla-de-dispersion/)

[**https://www.monografias.com/trabajos92/arboles-binario/arboles-binario.shtml**](https://www.monografias.com/trabajos92/arboles-binario/arboles-binario.shtml)

[**https://estructuradedatosunivia.wordpress.com/2014/11/18/metodos-de-ordenamiento-mergesort/**](https://estructuradedatosunivia.wordpress.com/2014/11/18/metodos-de-ordenamiento-mergesort/)

http://puntoflotante.org/errors/comparison/