**UNIVERSIDAD TECNOLOGICA DE SANTIAGO  
SISTEMA CORPORATIVO  
FACULTAD DE INGENIERÍA Y ARQUITECTURA  
CARRERA DE SISTEMAS COMPUTACIONALES**



**ALGORITMOS PARALELOS**

**GRUPO 3**

**Presentado a:**

**Iván Mendoza**

**Presentado por:**

**Sugeiri Torres 1-16-0736**

**Dinnibel Azcona 1-16-0788**

**Claudia Báez 2-16-1116**

**Santiago de los Caballeros,   
República Dominicana**

**INDICE PAG.**

**[Introducción 3](#_Toc3713)**

**[Descripción del Proyecto 4](#_Toc6847)**

**[Objetivos 4](#_Toc4263)**

[Objetivo General 4](#_Toc30392)

[Objetivos Específicos 4](#_Toc26136)

**[Definición de Algoritmos Paralelos 4](#_Toc2159)**

**[Etapas de los Algoritmos paralelos 4](#_Toc4218)**

[Partición 4](#_Toc10712)

[Comunicación 5](#_Toc3374)

[Agrupamiento 5](#_Toc18701)

[Asignación 6](#_Toc18234)

**[Técnicas Algorítmicas Paralelas 6](#_Toc19717)**

**[Modelos de Algoritmos Paralelos 7](#_Toc14749)**

**[Algoritmos de Búsquedas y Ordenamiento (Adjuntar PSeudocódigo y código de cada uno) 7](#_Toc15646)**

[Búsqueda Secuencial 7](#_Toc19755)

[Búsqueda Binaria 8](#_Toc10757)

[Algoritmo de Ordenamiento de la Burbuja 9](#_Toc5732)

[Quick Sort 9](#_Toc2742)

[Método de Inserción 10](#_Toc31456)

**[Programa desarrollado 11](#_Toc5808)**

[Explicación de su funcionamiento 11](#_Toc17225)

[Fotos de la aplicación 11](#_Toc11919)

[Link de GitHub y Ejecutable de la aplicación 11](#_Toc1533)

[Resultados (Tiempo en terminar los ordenamientos y búsqueda de cada algoritmo) 11](#_Toc11941)

**[¿Qué tanta memoria se consumió este proceso? 11](#_Toc30659)**

**[¿Cuál fue el algoritmo que realizo la búsqueda y el ordenamiento más rápido? Explique. 11](#_Toc9048)**

**[Conclusión 11](#_Toc23361)**

**[Bibliografías 11](#_Toc21821)**

# Introducción

El presente proyecto tiene como finalidad presentar los diferentes algoritmos de búsqueda y ordenamiento, explicar que son y como se implementa. También explicaremos acerca de que es algoritmo paralelo, sus etapas, técnicas y modelos.

Presentaremos de forma practica varios métodos como el quick sort y el método burbuja. Analizando la duración de los mismos hasta lograr determinar cual es el mas eficiente.

# Descripción del Proyecto

# Objetivos

## Objetivo General

Analizar los diferentes algoritmos de búsqueda y ordenamiento para determinar su eficiencia.

## Objetivos Específicos

* Implementar varios algoritmos de búsqueda y ordenamiento
* Estudiar el comportamiento de los diferentes algoritmos con respecto a un mismo vector
* Determinar, mediante análisis, cual de los algoritmos es el mas eficiente

# Definición de Algoritmos Paralelos

Es un algoritmo que puede ejecutar multiples instrucciones de forma simultanea en diferentes dispositivos de procesamiento, luego combinar las salidas individuales para producir el resultado final.

# Etapas de los Algoritmos paralelos

## Partición

El cómputo y los datos sobre los cuales se opera se descomponen en tareas. Se ignoran aspectos como el número de procesadores de la máquina a usar y se concentra la atención en explotar oportunidades de paralelismo.

En esta etapa se buscan oportunidades de paralelismo y se trata de subdividir el problema lo más finamente posible, es decir; que la granuralidad sea fina.

Un grano es una medida del trabajo Computacional a realizar.

Al particionar se deben tener en cuenta los siguientes aspectos:

* El número de tareas debe ser por lo menos un orden de magnitud superior al número de procesadores disponibles para tener flexibilidad en las etapas siguientes.
* Hay que evitar cómputos y almacenamientos redundantes; de lo contrario el algoritmo puede ser no extensible a problemas más grandes.
* Hay que tratar de que las tareas sean de tamaños equivalentes ya que facilita el balanceo de la carga de los procesadores.
* Considere alternativas de paralelismo en esta etapa ya que pueden flexibilizar etapas subsecuentes.
* El número de tareas debe ser proporcional al tamaño del problema. Así, se podrá resolver problemas más grandes cuando se tenga más procesadores. Es decir, que el algoritmo sea escalable.

## Comunicación

En esta etapa se determina la comunicación requerida para coordinar las tareas y se definen estructuras y algoritmos de comunicación.

La comunicación requerida por un algoritmo puede ser definida en dos fases.

* Primero se definen los canales que conectan las tareas que requieren datos con las que los poseen.
* Segundo se especifica la información o mensajes que deben ser enviado y recibidos en estos canales.

En la etapa de comunicación hay que tener en cuenta los siguientes aspectos:

* Todas las tareas deben efectuar aproximadamente el mismo número de operaciones de comunicación. Si esto no se da, es muy probable que el algoritmo no sea extensible a problemas mayores ya que habrán cuellos de botella.
* La comunicación entre tareas debe ser tan pequeña como sea posible.
* Las operaciones de comunicación deben poder proceder concurrentemente.
* Los cómputos de diferentes tareas deben poder proceder concurrentemente.

## Agrupamiento

El resultado de las dos etapas anteriores es evaluado en términos de eficiencia y costos de implementación. De ser necesario, se agrupan tareas pequeñas en tareas más grandes.

Esta etapa va de lo abstracto a lo concreto y se revisa el algoritmo obtenido tratando de considerar si es útil agrupar tareas y si vale la pena replicar datos y/o cómputos de acuerdo a la plataforma seleccionada.

Mediante la agrupación de tareas se puede reducir la cantidad de datos a enviar y así reducir el número de mensajes a transmitir y por ende el costo de comunicación.

En la etapa de agrupación se debe tener en cuenta los siguientes aspectos:

* Chequear si la agrupación redujo los costos de comunicación.
* Si se han replicado cómputos y/o datos, se debe verificar que los beneficios son superiores a los costos.
* Se debe verificar que las tareas resultantes tengan costos de computo y comunicación similares.
* Hay que revisar si el número de tareas es extensible con el tamaño del problema.
* Si el agrupamiento ha reducido las oportunidades de ejecución concurrente, se debe verificar que aun hay suficiente concurrencia y posiblemente considerar diseños alternativos.
* Analizar si es posible reducir aun más el número de tareas sin introducir des balances de cargas o reducir la extensibilidad.

## Asignación

Cada tarea es asignada a un procesador tratando de maximizar la utilización de los procesadores y de reducir el costo de comunicación.

La asignación puede ser:

* Estática:una tarea es asignada a un procesador desde su inicio hasta su fin
* Dinámica: una tarea puede ser migrada durante su ejecución. Esto puede agregar un costo adicional

Aquí se trata de establecer el mayor número posible de tareas con la intensión de explorar al máximo las oportunidades de paralelismo.

En esta etapa se determina en que procesador se ejecutará cada tarea

# Técnicas Algorítmicas Paralelas

# Modelos de Algoritmos Paralelos

**Paralelismo de datos**

Aquí Las tareas son mapeadas de manera cuasi-estática. Las operaciones paralelas son muy similares. Ademas de esto, necesita poca sincronización y asignación.

**Grafo de tareas**

Generalmente se usa cuando tenemos grandes volúmenes de datos. Y Usualmente se mapean estáticamente

**Conjunto de trabajadores (work pool)**

Mapeo dinámico de tareas a procesos.Es muy útil para el balance de carga. En este el mapeo puede ser centralizado o descentralizado

**Maestro-esclavo (master-slave)**

Un maestro genera trabajo y los esclavos ejecutan. Ayuda a resolver problemas de cuellos de botella

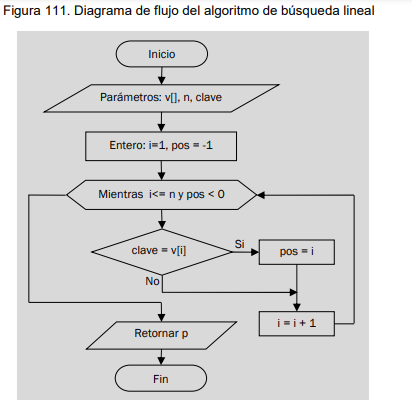
**Productor-consumidor (pipeline producer-consumer)**

Una secuencia de datos pasa a través de una serie de procesos. Aquí cada proceso realiza una tarea particular y todos los procesos actúan a la vez formando un pipeline

# Algoritmos de Búsquedas y Ordenamiento (Adjuntar PSeudocódigo y código de cada uno)

## Búsqueda Secuencial

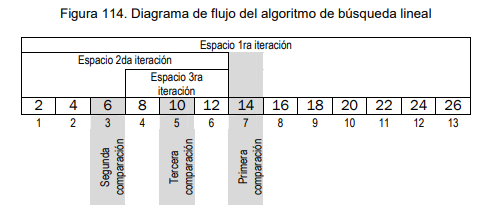
Consiste en tomar un dato clave que identifica al elemento que se busca y hacer un recorrido a través de todo el arreglo comparando el dato de referencia con el dato de cada posición.



## Búsqueda Binaria

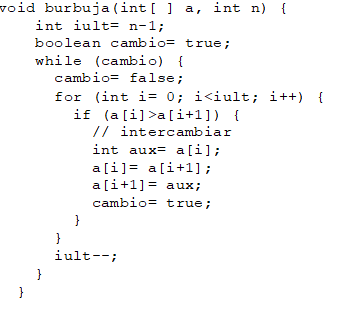
es más eficiente que la búsqueda secuencial pero sólo se puede aplicar sobre vectores o listas de datos ordenados.

En la búsqueda binaria no se hace un recorrido de principio a fin, sino que se delimita progresivamente el espacio de búsqueda hasta llegar al elemento buscado. La primera comparación se hace con el elemento de la mitad del arreglo, si aquel no es el dato buscado, se decide si buscar en la mitad inferior o en la mitad superior según la clave sea menor o mayor del elemento de la mitad. Se toma como espacio de búsqueda la mitad del vector que corresponda y se procede de igual forma, se compara con el elemento del centro, si ese no es el que se busca, se toma un nuevo espacio de búsqueda correspondiente a la mitad inferior o superior del espacio anterior, se compara nuevamente con el elemento del centro, y así sucesivamente hasta que se encuentre el elemento o el espacio de búsqueda se haya reducido un elemento.



## Algoritmo de Ordenamiento de la Burbuja

La idea consiste en realizar varios recorridos secuenciales en el arreglo intercambiando los elementos adyacentes que estén desordenados. en la primera iteración se lleva el máximo a su posición definitiva al final del arreglo. En la segunda iteración se lleva el segundo máximo a su posición definitiva. Y así continúa hasta ordenar todo el arreglo.



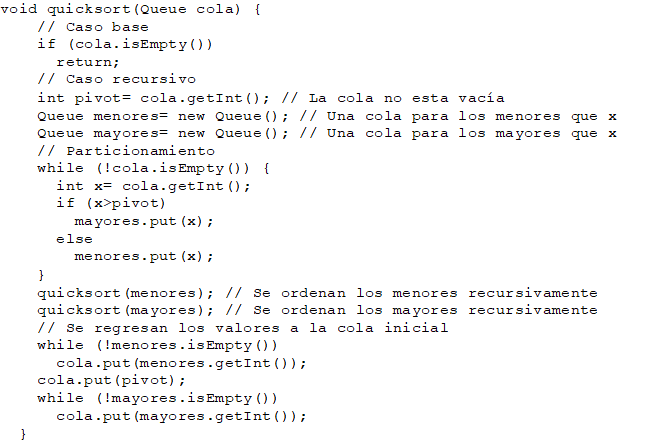
## Quick Sort

El algoritmo trabaja de la siguiente forma:

Elegir un elemento del conjunto de elementos a ordenar, al que llamaremos pivote.Re-situar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.

La lista queda separada en dos sub listas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.

Repetir este proceso de forma recursiva para cada sub lista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.



## Método de Inserción

# Programa desarrollado

## Explicación de su funcionamiento

## Fotos de la aplicación

## Link de GitHub y Ejecutable de la aplicación

## Resultados (Tiempo en terminar los ordenamientos y búsqueda de cada algoritmo)

# ¿Qué tanta memoria se consumió este proceso?

# ¿Cuál fue el algoritmo que realizo la búsqueda y el ordenamiento más rápido? Explique.

# Conclusión

# Bibliografías

<https://es.wikipedia.org/wiki/Algoritmo_paralelo#:~:text=En%20las%20ciencias%20de%20la,y%20obtener%20el%20resultado%20correcto.>

<https://www.hebergementwebs.com/tutorial-sobre-el-algoritmo-paralelo/algoritmo-paralelo-guia-rapida>

<https://www.inf.utfsm.cl/~noell/IWI-131-p1/Tema8b.pdf>

<http://biolab.uspceu.com/aotero/recursos/docencia/TEMA%208.pdf>

<http://webdelprofesor.ula.ve/ingenieria/gilberto/paralela/08_DisenoDeAlgoritmosParalelos.pdf>

<https://www.frbb.utn.edu.ar/hpc/lib/exe/fetch.php?media=2016-03-disenno-algoritmos-paralelos.pdf>