

Act 1.2 - Algoritmos de Búsqueda y Ordenamiento

Empezar tarea

Fecha de entrega Miércoles a las 23:59 **Puntos** 100

Entregando una carga de archivo **Tipos de archivo** zip y cpp

Disponible hasta el 6 de jul en 23:59

¿Qué niveles de dominio de subcompetencias voy a demostrar con esta evidencia?

SICT0301B - Evalúa los componentes que integran una problemática de acuerdo a principios y procesos computacionales.

SICT0302B - Toma decisiones en la solución de problemas en condiciones de incertidumbre y diferentes niveles de complejidad con base metodologías de investigación y de cómputo.

SICT0303B - Implementa acciones científicas e ingenieriles o procesos computacionales que cumplen con el tipo de solución requerida.

SEG0702A - Tecnologías de Vanguardia. Evalúa diversas tecnologías con apertura a la búsqueda e implementación de alternativas relevantes en la transformación de la práctica profesional.

¿Qué tengo que hacer?

Realiza una aplicación en C++, en donde dado **n** números de entrada que son almacenados en un **vector<int>** realice las siguientes funciones de ordenamiento y búsqueda:

ordenaBurbuja	Descripción	Ordene en forma ascendente los datos con el método de Burbuja
	Entrada	Un vector<int> con los n números
	Salida	Nada
	Precondición	El vector<int> debe contener los n números
	Postcondición	El vector<int> contendrá los datos ya ordenados

ordenaMerge	Descripción	Ordene en forma ascendente los datos con el método de Merge
	Entrada	Un vector<int> con los n números
	Salida	Nada
	Precondición	El vector<int> debe contener los n números
	Postcondición	El vector<int> contendrá los datos ya ordenados


ordenaQuick	Descripción	Ordene en forma ascendente los datos con el método de Quicksort
	Entrada	Un vector<int> con los n números
	Salida	Nada
	Precondición	El vector<int> debe contener los n números
	Postcondición	El vector<int> contendrá los datos ya ordenados

busquedaSecuencialOrd	Descripción	Realiza la búsqueda secuencial de un dato entero dentro del vector ordenado.
	Entrada	El vector <int> del espacio de búsqueda y el dato entero que se desea buscar.
	Salida	El índice donde se encuentra el dato o -1 en caso de que no se localice.
	Precondición	El vector<int> debe contener los n números ordenados en forma ascendente
	Postcondición	Ninguna

busquedaBinaria	Descripción	Realiza la búsqueda binaria de un dato entero dentro del vector ordenado.
	Entrada	El vector <int> ordenado del espacio de búsqueda y el dato entero que se desea

	buscar.
Salida	El índice donde se encuentra el dato o -1 en caso de que no se localice.
Precondición	El vector<int> debe contener los n números ordenados en forma ascendente
Postcondición	Ninguna

La aplicación debe cumplir con los siguientes requerimientos:

1. Documentar en el encabezado del archivo fuente las instrucciones detalladas para compilar (banderas de compilación necesarias) y ejecutar su programa.
2. Recibe por redirección de la línea de comandos cada uno de los casos de prueba siguientes:
[TestCases-SortingAlgorithms.zip](https://experiencia21.tec.mx/courses/274974/files/100397065/download?download_frd=1)  (https://experiencia21.tec.mx/courses/274974/files/100397065/download?download_frd=1)
3. Los casos de prueba tienen el formato siguiente:
 - Línea 1 contiene un entero con el tamaño del arreglo a ordenar.
 - Línea 2 contiene un entero que indica el algoritmo de ordenamiento a emplear (1: burbuja, 2: merge, 3: quicksort).
 - Línea 3 contiene un entero que indica el algoritmo de búsqueda a emplear (1: búsqueda secuencial, 2: búsqueda binaria).
 - Línea 4 contiene un entero que indica el número a buscar en el arreglo ya ordenado.
 - Líneas 5 en adelante contienen los elementos del arreglo a ordenar.
4. Para cada caso de prueba la aplicación imprime en la salida estándar únicamente la siguiente información:
 - El arreglo ordenado
 - El número total de comparaciones efectuadas por el algoritmo de ordenamiento
 - El número total de intercambios (swaps) efectuados por el algoritmo de ordenamiento
 - El resultado del algoritmo de búsqueda
 - El número total de comparaciones efectuadas por el algoritmo de búsqueda
5. Todas las funcionalidades deberán de estar correctamente alineadas y documentadas. Como parte de la documentación deberá incluirse la complejidad de cada una de ellas.

¿Bajo qué criterios se evalúa mi evidencia?

- **70%** - Lista de 4 casos de prueba para cada una de las funcionalidades donde para cada una se evaluará:

- **Excelente (70%)** - evalúa correctamente los 4 casos de prueba.
- **Muy Bien (55%)** - evalúa correctamente 3 casos de prueba.
- **Bien (40%)** - evalúa correctamente 2 casos de prueba
- **Insuficiente (25%)** - evalúa correctamente 1 o 0 casos de prueba.
- **10%** - El código deberá seguir los lineamientos estipulados en el estándar de codificación:
[liga_estándar_codificación](https://experiencia21.tec.mx/courses/274974/files/100374188/download?download_frd=1) ↓ (https://experiencia21.tec.mx/courses/274974/files/100374188/download?download_frd=1)
- **10%** - Se respetaron los nombres de las funciones en la aplicación.
- **10%** - Especifican en cada una de las funcionalidades **la complejidad computacional** como parte de su documentación.

¿Dónde la entrego?

Entrega los archivos correspondientes de la actividad, en la sección correspondiente dentro de esta plataforma.