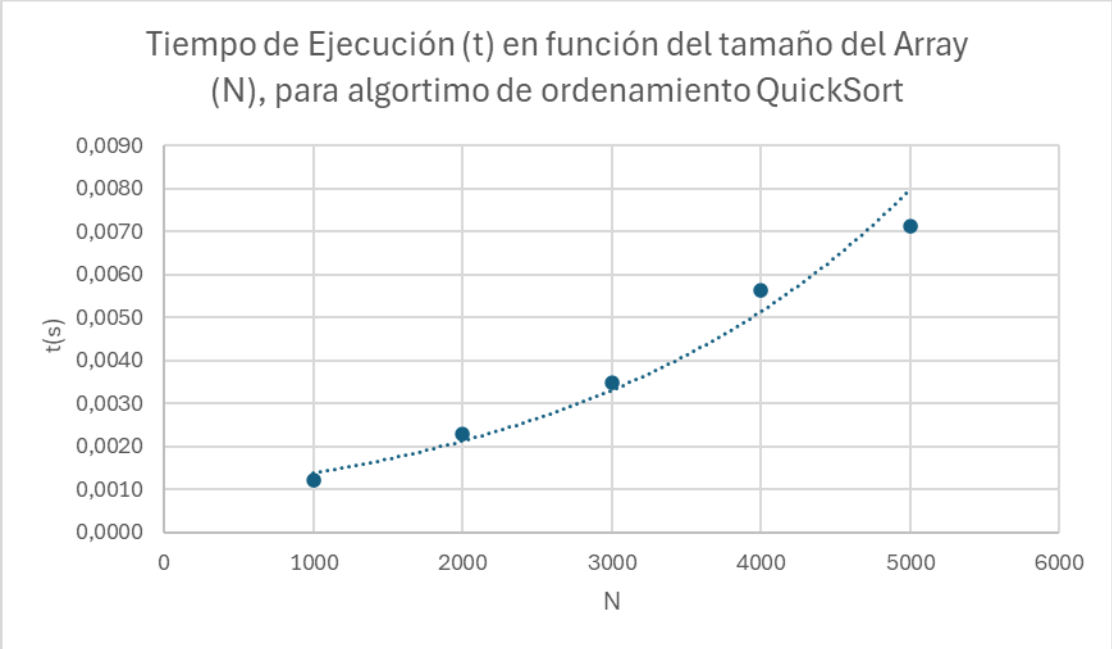
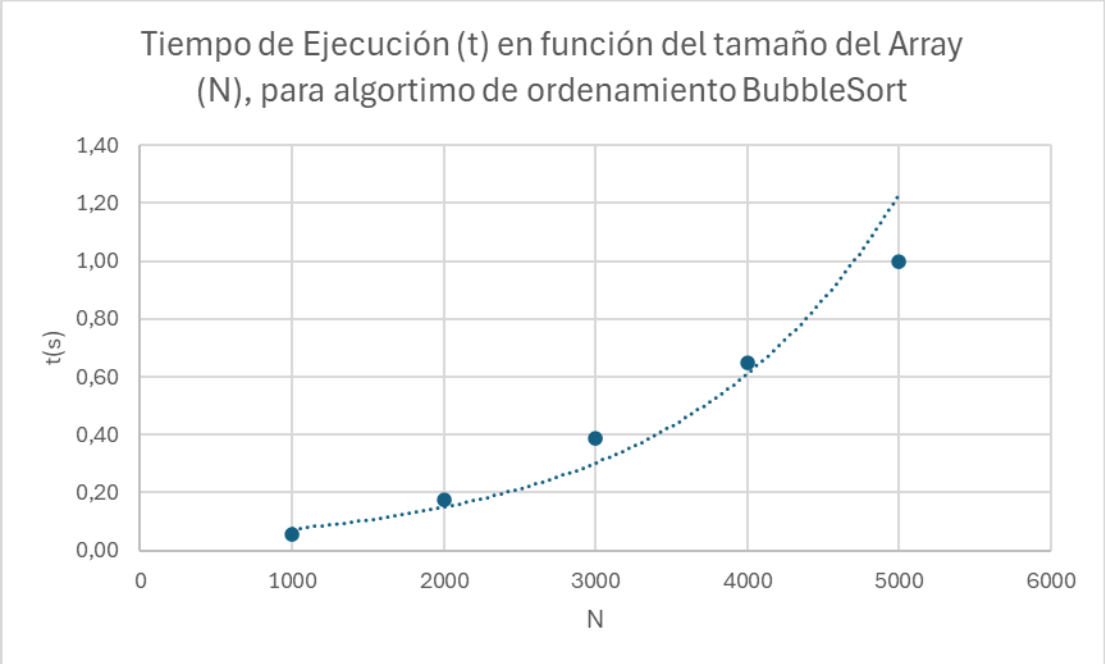
**GRAFICA QUICK SORT**




**GRAFICA BUBBLE SORT**



**PROBLEMA 1**

5. Responda las siguientes preguntas

a. ¿Cuál algoritmo es más rápido y por qué?

El algoritmo más rápido entre ambos es el quick sort, esto lo podemos apreciar primeramente por su complejidad la cual es O(n log n) a diferencia que bubble sort que es O(n^2), esto nos indica como aumenta el tiempo de ejecución a comparación del tamaño del arreglo, siendo el bubble sort cuadrático, además también podemos notar por medio de sus implementaciones cual va a durar más, esto debido a que el bubble sort debe de realizar una cantidad mayor de comparaciones, debido a que va recorriendo la lista y haciendo comparaciones y tendrá que hacer esto hasta que la lista este completamente ordenada, por otro lado el quick sort, elige un pivot y va ordenando de izquierda del pivot más pequeños y derecha más grandes, aplicando recursividad y ordenando eficientemente.

b. ¿El tiempo de ejecución será el mismo si la implementación del algoritmo es iterativa o recursiva?

En teoría el hecho de que el algoritmo sea recursivo o iterativo no afecta en la complejidad de este, en realidad se implementa de una u otra forma por facilidad de entender el código, como el quick sort utiliza el modelo de divide y vencerás, la recursividad es entendible para dividir el problema principal en subproblemas e ir ordenando pedacitos, por otro lado el bubblesort es recorrer haciendo comparaciones, fuerza bruta, por lo que la iteración funciona bien, el único cambio entre las 2 implementaciones es la cantidad de memoria que requiere su ejecución.

c. ¿Es posible que exista un algoritmo de ordenamiento que sea muy eficiente en consumo de recursos pero que a la vez sea relativamente rápido?

Si, de hecho, el quick sort ya es un ejemplo de esto, en su implementación iterativa, ya que debería de utilizar el mismo espacio en memoria que cualquier función iterativa y su complejidad es bastante buena, por lo que en realidad aquí tendríamos un algoritmo eficiente y no tan caro, otro ejemplo seria el heap sort que posee la complejidad del quick sort y ocupa una cantidad accesible de memoria.

d. Suponga que se planea ejecutar el algoritmo en un sistema computacional con extremadamente bajos recursos de memoria. ¿Cuál de los dos algoritmos de ordenamiento escogería y por qué?

Aquí habría que valorar los usos que se les vayan a dar al algoritmo ya que como mencione anteriormente el hecho de que el código sea recursivo consume mas memoria, por lo que la respuesta simple seria que el bubble sort sería mejor para un sistema de escasa

memoria debido a que es iterativo, pero también hay que tener en cuenta que el quick sort se puede implementar de manera iterativa, entonces por ejemplo si ocupamos manejar grandes volúmenes de datos, podríamos considerar esta implementación o por ejemplo si los demás recursos del sistema son bajos también, hay que tener en cuenta si puede aguantar o no todas las comparaciones del bubble sort.

**PROBLEMA 2**

1. ¿Cuál es la diferencia entre el algoritmo de búsqueda lineal y búsqueda por interpolación?

La búsqueda lineal consiste en encontrar un elemento especifico de la lista recorriendo la lista por completo, esto por medio de iteración. La búsqueda por interpolación consiste en ya teniendo una lista de números ordenada poder estimar la posición del valor a encontrar mediante comparaciones generales, por ejemplo, comparar si el elemento central es mayor o menor al elemento a buscar nos hace por descartar la mitad de la lista y así consecutivamente.

2. Suponga que se tiene que buscar un elemento en una lista desordenada, pero se desea optimizar el tiempo de búsqueda por sobre cualquier otra métrica ¿Cómo se podría hacer eso?

Lo más recomendable para realizar esta tarea es antes de realizar la búsqueda, ordenar esta lista desordenada por medio de algún algoritmo de ordenamiento (por ejemplo quick sort que es el más eficiente) y si lo que vamos a priorizar por encima de todo es la velocidad de búsqueda, entonces debemos utilizar el algoritmo de búsqueda hash lookup que tiene más complejidad que el binary search pero mejora el tiempo del anterior, si no podemos o queremos ordenar la lista, la única forma que tenemos de ordenarlo es recorrer al menos una vez la lista ya que no conocemos donde puede estar.

3. Busque y explique alguna aplicación de la vida real donde el tiempo de búsqueda en una lista o en un arreglo sea crítico para que la aplicación se pueda dar.

El tiempo de búsqueda sobre una lista se vuelve un factor crítico cuando el sistema requiere ya sea, manejar grandes volúmenes de datos haciendo que la velocidad de búsqueda entre estos no se ralentice o por ejemplo en los videojuegos donde los datos están actualizándose siempre en tiempo real, por lo que si no se ejecutaran rápidamente generarían cuello de botellas afectando de gran medida la experiencia del jugador.