**Objetivos**

U1.2. Aplicar conceptos de análisis de complejidad de algoritmos para describir y comparar formalmente (utilizando elementos matemáticos como funciones y notación asintótica) diferentes estrategias de solución al mismo problema.

U1.3. Evaluar los resultados de un proceso de análisis computacional e identificar los elementos relevantes que soportan la elección e implementación de una solución en particular.

U1.4. Implementar pruebas unitarias automáticas utilizando el nuevo lenguaje de programación.

U1.6. Llevar a cabo experimentos relacionados con el desarrollo de programas.

U1.9. Utilizar PSP para el seguimiento durante las etapas de desarrollo de programas.

U2.6. Analizar la información resultante de la ejecución de experimentos.

**Enunciado**

El ordenamiento de elementos en una estructura de datos lineal es un problema que ha sido ampliamente estudiado. Se han utilizado diversas técnicas de diseño de algoritmos para abordarlo e incluso se han encontrado y demostrado límites de eficiencia para los algoritmos de propósito general basados en comparación.

Uno de los algoritmos más eficientes encontrados hasta el momento es el famoso QuickSort, y de ahí justamente su nombre. Este algoritmo tiene varias versiones, de las cuales presentamos a continuación dos de ellas.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Línea** | **Instrucción** | **# veces que se ejecuta** | |  | **QuickSort**(A,p,r) |  | | 1 | **if** p < r |  | | 2 | q = Partition(A,p,r) |  | | 3 | QuickSort(A,p,q-1) |  | | 4 | QuickSort(A,q+1,r) |  |  |  |  |  | | --- | --- | --- | | **Línea** | **Instrucción** | **# veces que se ejecuta** | |  | **Partition**(A,p,r) |  | | 1 | x = A[r] |  | | 2 | i = p - 1 |  | | 3 | **for** j = p to r - 1 |  | | 4 | **if** A[j] < x |  | | 5 | i = i + 1 |  | | 6 | A[i] A[j] |  | | 7 | A[i + 1] A[r] |  | | 8 | **return** i + 1 |  |   ***QuickSort No Randomized*** | |  |  |  | | --- | --- | --- | | **Línea** | **Instrucción** | **# veces que se ejecuta** | |  | **Randomized-QS**(A,p,r) |  | | 1 | **if** p < r |  | | 2 | q = Rand-Parti(A,p,r) |  | | 3 | Randomized-QS(A,p,q-1) |  | | 4 | Randomized-QS(A,q+1,r) |  |  |  |  |  | | --- | --- | --- | | **Línea** | **Instrucción** | **# veces que se ejecuta** | |  | **Rand-Parti**(A,p,r) |  | | 1 | i = Random(p,r) |  | | 2 | A[r] A[i] |  | | 3 | **return** Partition(A,p,r) |  |   ***Variante del QuickSort Randomized*** |

Usted debe llevar a cabo el análisis de complejidad temporal de los algoritmos: QuickSort y Randomized-QS y presentar el procedimiento y sus resultados tal como se trabajó en su curso de Algoritmos y Estructuras de Datos. Puede apoyarse en el Capítulo 7 del libro Introducción a los Algoritmos de Cormen et al. Tenga en cuenta que apoyarse no quiere decir que su respuesta va a ser un copy-paste del libro.

Como resultado del análisis anterior usted obtiene el comportamiento de cada algoritmo para el caso mejor, peor y promedio. Luego, usted implementará, de la manera más fiel posible, estos algoritmos en su nuevo lenguaje de programación favorito: C#.

Antes o durante (pero no después) de la implementación de estos algoritmos, usted deberá llevar a cabo el diseño de pruebas unitarias automáticas para validar la correctitud del algoritmo frente a entradas de diversos tamaños. Pruebe para entradas pequeñas, grandes y aleatorias. Luego, implemente en C# estas pruebas unitarias automáticas.

Luego de esto, implementará una serie de pruebas (estas no son pruebas unitarias automáticas) para cada uno de los algoritmos. Estas pruebas tienen como objetivo experimentar sobre el comportamiento del algoritmo para validar los resultados del análisis de complejidad temporal. Para las siguientes pruebas, consideraremos el QuickSort como un solo algoritmo pero parametrizado, de tal manera que con un parámetro se llevará a cabo Randomized y con otro no.

La ejecución básica de las pruebas en un experimento implica el establecimiento de unas condiciones bajo las cuales se llevará a cabo éste. Dentro de estas condiciones se encuentran una serie de factores de entrada que vamos a controlar y unas características de salida que evaluaremos como resultado de hacer la prueba con distintos valores para esos factores de entrada.

A continuación se muestran los factores de entrada que se tendrán en cuenta (columna 2 a la 4) para cada tipo de prueba que se llevará a cabo sobre el algoritmo, y la última columna es la variable de salida que vamos a probar.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Factores de Entrada** | |  |
| **# de Prueba** | **Variante del QS**  **X1** | **Estado del Arreglo**  **X2** | **Tiempo (en ms)**  **Y** |
| 1 | No Randomized | Ordenado No Descendente | Aquí debería escribir la hipótesis a validar, resultado del análisis de complejidad temporal previo.  Debería ser una función en términos del tamaño de la entrada y algunas constantes |
| 2 | Randomized | Ordenado No Descendente |
| 3 | No Randomized | Ordenado No Ascendente |
| 4 | Randomized | Ordenado No Ascendente |
| 5 | No Randomized | No ordenado (en orden aleatorio) |
| 6 | Randomized | No ordenado (en orden aleatorio) |

Para cada una de las anteriores pruebas usted generará arreglos con valores enteros aleatorios positivos entre 1 y 10 ^ 9 en cada posición. Además para cada una de las pruebas usted debe llevar a cabo una serie de repeticiones. El número de repeticiones será de 1000. Es decir, cada prueba debe ser repetida 10 ^ 3 veces y registrar los resultados.

Cada una de las seis pruebas anteriores para cada una de las repeticiones debe llevarlas a cabo con los siguientes tamaños de arreglos: 10 ^ 1, 10 ^ 2, 10 ^ 3, 10 ^ 4 y 10 ^ 5.

***Aclaración***

Para la ejecución del experimento debe tener en cuenta lo siguiente: Ya que son 10 ^ 3 repeticiones, usted deberá generar 10 ^ 3 arreglos de cada uno de los tamaños indicados anteriormente (10 ^ 1, 10 ^ 2, …, 10 ^ 5) y son esos mismos arreglos los que usted ingresará como entrada al algoritmo para cada una de las 6 pruebas indicadas en la tabla. **Importante:** para asegurar que se cumpla con las condiciones establecidas en cada uno de los valores de entrada, esos 10 ^ 3 arreglos de cada tamaño (10 ^ 1, 10 ^ 2, …, 10 ^ 5) usted deberá tenerlos (los mismos arreglos) en 3 formas: Ordenado No Descendente, Ordenado No Ascendente y No ordenado, para suministrarlos como entradas a cada variante del algoritmo en cada una de las pruebas.

Cada uno de los integrantes del grupo debe leer el texto publicado en moodle sobre Introducción al Diseño de Experimentos (Capítulo 1 del libro Análisis y Diseño de Experimentos), y luego de esto con base en la ejecución del experimento anterior:

1. Identifique:
   1. Unidad experimental
   2. Variables de respuesta
   3. Factores controlables
   4. Factores no controlables
   5. Factores estudiados
   6. Niveles
   7. Tratamientos
2. ¿Qué etapas del análisis y diseño de experimentos se han llevado a cabo hasta el momento? ¿Cuáles hacen falta?
3. El objetivo del anterior experimento en cual categoría de objetivos de experimentos clasificaría? Explique brevemente.
4. Con cada configuración se realizaron 1000 ejecuciones, y se almacenaron 1000 tiempos de ejecución correspondientes. Compruebe desde el punto de vista estadístico, con un nivel de confianza del 99%, si se cumple que la versión aleatoria (randomized) es más rápida que la no aleatoria. Utilice ANOVA para responder a esta pregunta, indicando el valor P correspondiente a cada prueba de hipótesis.

**Condiciones de Entrega**

1. El problema debe ser resuelto en grupos de **máximo** 4 personas.
2. A través de moodle a más tardar en la fecha y hora indicadas en el enlace en moodle.
3. Los entregables son:
   1. Un informe en formato pdf el cual se consigne el análisis de los algoritmos y los resultados del experimento.
   2. La implementación en C# de los algoritmos, las pruebas unitarias automáticas y las pruebas del experimento.
   3. Un video en el cual, todos los integrantes, expliquen todos los entregables del taller. El video debe durar máximo 5 minutos. Cada integrante debe explicar una parte de la solución del taller, y cada vez que aparezca un integrante, este debe verse en video, al menos en una miniatura en una esquina de la pantalla.