**EXPERIMENTO PARA COMPARAR LA EFICIENCIA DEL ALGORITMO QICKSORT Y QUICKSORT RANDOMIZED**

Leonardo Franco. Brayan Grueso. Anderson Ramírez.

Universidad Icesi, Facultad de Ingeniería, Proyecto Integrador I.

Santiago de Cali, Colombia – septiembre 10 de 2018.

**RESUMEN**

El ordenamiento de elementos en una estructura de datos es un problema que ha inquietado a la comunidad científica y se ha estudiado de forma rigurosa como abordar esa problemática de acuerdo con factores específicos que intervienen directamente en la estructura que almacena esos datos, como lo es la dimensión de esta, el número de elementos y en el orden en que se encuentren. De acuerdo con los distintos factores, se han diseñado diferentes algoritmos, donde cada uno funciona mejor en distintos casos y también donde existen variantes de los algoritmos de ordenamiento que buscan mejorar su comportamiento en situaciones que son las menos favorables para el algoritmo (peor caso). En este experimento se busca comparar el algoritmo de ordenamiento *QuickSort* y su variante *Randomized-QuickSort* para determinar cuál de los dos es más eficiente de acuerdo a diferentes factores del entorno y contrastar los datos con el análisis de complejidad temporal de ambos algoritmos.

1. **INTRODUCCIÓN**

*QuickSort* es un algoritmo recursivo de ordenamiento *inplace*, que funciona para ordenar elementos en estructuras de datos lineales, donde el tiempo de ejecución para el caso promedio está dado por Θ*(n log n)* y para el peor caso O*(n2).* Es un algoritmo recursivo que va particionando el arreglo de datos tomando como referencia un *pivot* y los va ordenando los menores a la izquierda del *pivot* y los mayores a la derecha hasta llegar al caso base donde el arreglo es de tamaño 1 y ya está ordenado. Al ser un algoritmo *inplace* es eficiente en cuanto a memoria se refiere, debido a que no utiliza estructuras auxiliares para realizar el ordenamiento, sino que lo hace mediante intercambios.

De igual forma, el *Randomized-QuickSort* es un algoritmo recursivo *inplace*, donde su única variación con el *QuickSort* es que elige el elemento *pivot* de forma aleatoria y luego se hace el particionado del arreglo tomando como referencia dicho elemento *pivot*, logrando para el peor de los casos una complejidad asintótica de O*(n log n).*

1. **ANALISIS Y CARACTERIZACIÓN**

Inicialmente, se realiza un análisis de la complejidad temporal del algoritmo QuickSort y de su variante para determinar matemáticamente como se comportan cada uno para el peor caso y para el caso promedio, utilizando técnicas de conteo de líneas para algoritmos recursivos.

Con base en ese análisis se realizó una caracterización, donde se establece como se espera que se comporte cada algoritmo para un tamaño de entrada y se establece una hipótesis que establece qué algoritmo se demora menos tiempo en ejecutarse de acuerdo con los factores que influyen en su conducta.

1. **MARCO PROCEDIMENTAL**

**Materiales.**

Visual Studio 2017

Computador

**Procedimiento**

Se obtienen las ecuaciones que describen el comportamiento de cada algoritmo con la relación tamaño-tiempo junto con su notación asintótica.

Haciendo uso del entorno de desarrollo de Visual Studio se implementa los métodos de ordenamiento de *QuickSort* y *Randomized-QuickSort,* y se realiza pruebas unitarias automatizadas para verificar la correctitud de los algoritmos para posteriormente realizar las pruebas del experimento.

Se prepara el entorno de ejecución de las pruebas, creando arreglos de tamaño 10 ^ 1, 10 ^ 2, 10 ^ 3, 10 ^ 4 y 10 ^ 5, cada uno con tres variaciones: ordenado no ascendente, ordenado no descendente y no ordenado.

Se realiza la ejecución de las pruebas para cada tamaño de entrada y se repite 1000 veces, se toma el tiempo de ejecución y se promedia en cada variante de la prueba.

Con los datos recolectados de los promedios se procede a evaluar la hipótesis utilizando la *prueba de hipótesis para comparar la media de 3 o mas poblaciones ANOVA*.

1. **RESULTADOS**

Unidad experimental:

Para realizar el correcto funcionamiento de los algoritmos Quicksort  y Randomized-QuickSort, se realizaron varias pruebas sobre arreglos, los cuales son las unidades experimentales, cuyo tamaño máximo será 10^5, en los cuales, se medirá el tiempo en que estos se demoran ordenando los arreglos. Dicho experimento se realizará 10^n veces, donde n será el exponente al que se elevará el 10 (n máximo es 5). Además, para cada posición del arreglo, se generará un número entre 1 y 10^9. También, cada vez que se haga una repetición, se cambiarán los números de las posiciones del arreglo, lo cual ayudará a ver cuál es el mejor algoritmo para el peor caso, puesto que el mejor caso sería que el arreglo fuera el mismo para las n repeticiones.

Variables de respuesta:

La complejidad temporal de cada algoritmo, que determina cuál tiene más eficiencia debido a que define cuál tiene menor tiempo de ejecución y nos indica, para nuestro problema, cuál sería mejor. Por lo tanto, las variables de respuestas seria el tiempo de ejecución del algoritmo. Durante el experimento, se tratará de mejorar para poder dar con el algoritmo que mejor se adapte al número de casos con los que se trabajará.

Factores controlables: Para este problema, el factor controlable sería el tamaño de los arreglos, que ya están definidos, al igual que el tipo de dato que se va a analizar durante el experimento. El tiempo también se puede considerar un factor controlado, puesto que siempre se tratará de que la duración del proceso sea la menor posible.

Factores no controlables: Dentro del experimento, una de las cosas que podría afectar el proceso, sería la capacidad del equipo, (características de procesador, cantidad de memoria ram) puesto que cada equipo cuenta con diferentes capacidades de hardware. Así pues, no todos podrán contar con una buena máquina para poner a prueba la solución y mucho menos cuando se trata de problemas tan grandes como este. No es lo mismo utilizar un computador “gaming” que un computador estándar, la diferencia, en este caso, de tiempo, será enorme.

También, otro factor no controlable, sería la temperatura en el ambiente que se trabaja. Puesto que, los computadores se calientan, sumándole la temperatura externa, podría resultar en un daño en la computadora, ya sea interno o externo, al igual que el mal funcionamiento de esta.

Factores estudiados: Los factores estudiados en el experimento son aquellos algoritmos que están no ordenado no ascendente, ordenado no descendente, y ordenado aleatoriamente.

Niveles y tratamientos:

|  |  |  |  |
| --- | --- | --- | --- |
| # de prueba | Tipo de Quick Sort | Estado del arreglo | Tratamiento |
| 1 | Normal | Ordenado no descendente | 1 |
| 2 | Random | Ordenado no descendente | 2 |
| 3 | Normal | Ordenado no ascendente | 3 |
| 4 | Random | Ordenado no ascendente | 4 |
| 5 | Normal | No ordenado(en orden aleatorio) | 5 |
| 6 | Random | No ordenado(en orden aleatorio) | 6 |

***Tabla 1. Niveles y tratamientos.*** *Muestra los factores de entrada para cada tratamiento*

**Las etapas que se han llevado a cabo hasta ahora son:**

**Planeación y realización.**

Delimitar el problema y objeto de estudio: En este caso, ya se llevó a cabo la identificación de lo que se quiere estudiar. Se quiere saber, cuál de los 2 algoritmos es el más eficiente para el mejor y peor caso. Además de que se quiere tener un tiempo estimado de cuanto se demoran los algoritmos al realizar el proceso de ordenamiento de arreglos de grandes tamaños con tipos de dato entero que van desde 1 hasta 10^9.

**Elegir la(s) variable(s) de respuesta que será medida en cada punto del diseño y verificar que se mide de manera confiable.**

Las variables que afectan a la respuesta son; el tamaño del arreglo, la cantidad de veces que se repetirá el proceso, y también, el tipo de arreglo ( No ordenado ascendentemente, no ordenado descendentemente, no ordenado aleatorio).

**Determinar cuáles factores deben estudiarse o investigarse, de acuerdo con la supuesta influencia que tienen sobre la respuesta.**

Aquí, ellos determinan que una de las cosas que se debe de investigar, que influye en el experimento, es el factor del tamaño del arreglo, al igual que la generación aleatoria de los números del arreglo.

**Seleccionar los niveles de cada factor, así como el diseño experimental adecuado a los factores que se tienen y al objetivo del experimento.**

Se sabe cuantas repeticiones se realizará para cada nivel, serán 10^3 repeticiones, con las cuales, se podrá determinar el tiempo en el que se ejecutará.

Lo que hace falta del diseño de experimentos sería “**Planear y organizar el trabajo experimental**”, “**Realizar el experimento**”, “**Análisis**”, “**Interpretación**” y “**Control y conclusiones finales**”.

c)

Se podría clasificar como:

* Diseños para determinar el punto óptimo de operación del proceso.
* Diseños para estudiar el efecto de varios factores sobre la(s) respuesta(s).

**Análisis de varianza de un factor:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESUMEN |  |  |  |  |
| Grupos | Cuenta | Suma | Promedio | Varianza |
| Columna 1 | 6 | 0,0599679 | 0,00999465 | 7,72087E-05 |
| Columna 2 | 6 | 0,6710203 | 0,111836717 | 0,005024627 |
| Columna 3 | 6 | 15,6259042 | 2,604317367 | 3,388199754 |
| Columna 4 | 6 | 1025,446529 | 170,9077548 | 57784,20456 |

***Tabla 2. Datos de las medias de los tratamientos***.

**Análisis de varianza:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ANÁLISIS DE VARIANZA |  |  |  |  |  |  |
| Origen de las variaciones | Suma de cuadrados | Grados de libertad | Promedio de los cuadrados | F | Probabilidad | Valor crítico para F |
| Entre grupos | 130074,4357 | 3 | 43358,14523 | 3,001207652 | 0,054795978 | 4,938193382 |
| Dentro de los grupos | 288937,9893 | 20 | 14446,89947 |  |  |  |
| Total | 419012,425 | 23 |  |  |  |  |

***Tabla 3. Analisis de la varianza con la anova***

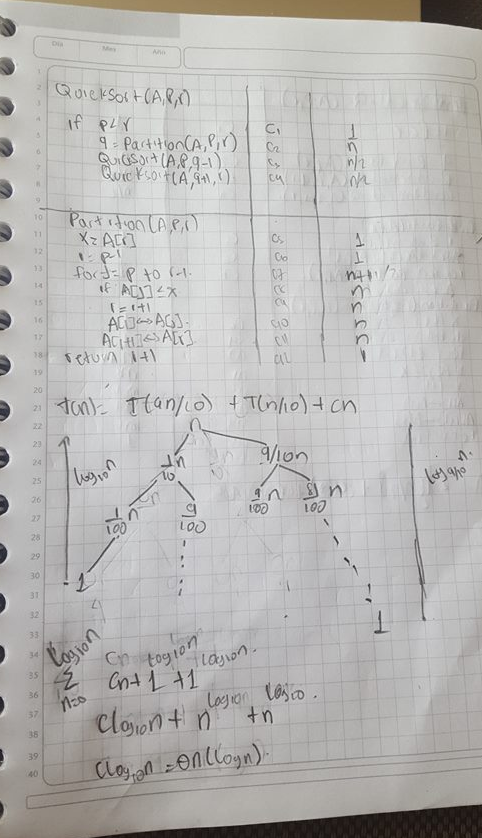
1. **CONCLUSIONES**

**Complejidad temporal Quick-Sort**

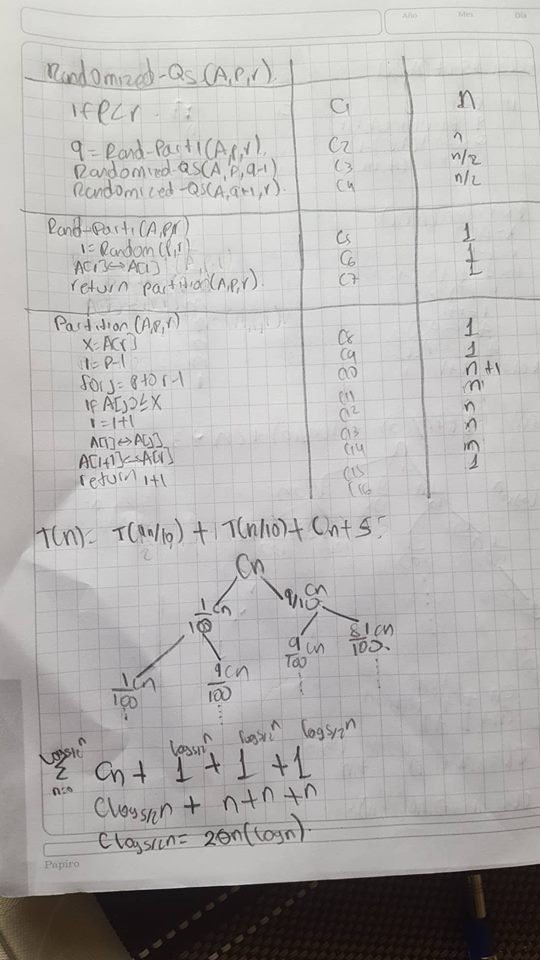
Teniendo en cuenta los resultados del experimento, se puede concluir que ambos algoritmos funcionan para resolver el problema de ordenamiento, pero, dependiendo de las condiciones en las que cada algoritmo se ejecute (tamaño del arreglo de entrada) será más eficiente. De las pruebas, obtuvimos que el *QuickSort* se demora mucho más en entradas muy grandes y que su variante, el *Randomized-QuickSort* funciona mucho mejor en esos casos y que en datos medianos y pequeños funciona mejor el QuickSort que su variante randomizado.

Para fines de este experimento, en computadores caseros, no alcanzó a superar las pruebas con entradas de 10^5 porque se desborda debido a sus recursos limitados. También se obtuvo que, usando métodos de linq de c# para la preparación de los arreglos el tiempo disminuyó un poco.

En definitiva, omitiendo las características físicas de los computadores, se puede concluir que para entradas grandes es mejor el *Randomized-QuickSort* porque en la mayoría de casos se demora para el peor de los casos O*(n log n)*, contrario al *QuickSort* que se demora para el peor de los casos O*(n2)*, lo que se pudo constatar con las pruebas.



Complejidad Temporal Randomized Quick-Sort



**REFERENCIAS**

Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein (2009) Introduction to algorithms