Universidad Nacional de San Agustín de Arequipa
Escuela Profesional de Ciencia de la Computación
Curso: Análisis y Diseño de Algoritmos

LABORATORIO 6

Quicksort

Docente: Rolando Jesús Cárdenas Talavera

1 Competencia del Curso

El alumno comprenderá e identificará el uso adecuado de diferentes algoritmos para dar solución a problemas de manera eficiente teniendo en consideración el tiempo de procesamiento y la cantidad de recursos empleados.

2 Competencia del Laboratorio

El alumno deberá de analizar y comprender las diferentes técnicas de diseño de Algoritmos

3 Equipos y Materiales

- Un computador.
- Compilador del lenguaje C++

4 Marco Teórico

En la siguiente tabla se muestra en resumen el tiempo de ordenar elementos en el orden de miles de datos a billones de datos utilizando los algoritmos de ordenamiento vistos en clase.

	Insertion Sort(n^2)			Mergesort ($n \log n$)			Quicksort ($n \log n$)		
Computadora	Miles	Millones	Billones	Miles	Millones	Billones	Miles	Millones	Billones
Laptop	instantáneo	2.8 hs	317 años	instantáneo	1 seg	18 min	instantáneo	0.6 sec	12 min
Super computador	instantáneo	1 seg	1 semana	instantáneo	instantáneo	instantáneo	instantáneo	instantáneo	instantáneo

5 Actividad

5.1 Ejercicio 1

Implemente el algoritmo Quicksort de acuerdo a la implementación vista en clase [1].

Quicksort (A, p, r):
if $p < r$ **then**
  q = Partition(A, p, r)
  Quicksort$(A, p, q-1)$
  Quicksort$(A, q+1, r)$
end if

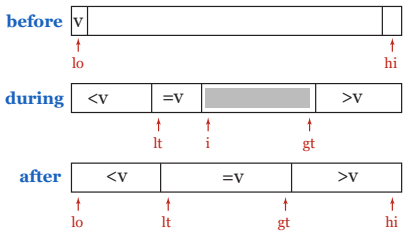
```
Partition(A, p, r):
x = A[r]
i = p − 1
for j = p to r − 1 do
  if A[j] ≤ x then
    i = i + 1
    exchange A[i] with A[j]
  end if
end for
exchange A[i + 1] with A[r]
return i+1
```

5.2 Ejercicio 2

El código implementado en el Ejercicio 1, no posee un mecanismo para detener el algoritmo en caso que el Array ya se encuentre ordenado. Proponga un método que pueda detectar cuando un Array ya se encuentra ordenado. (Incluya el código completo del método propuesto y su ubicación dentro del Quicksort, debe contener el análisis del tiempo de ejecución de su propuesta y como afecta el tiempo del Quicksort)

5.3 Ejercicio 3

E.W.Dijkstra popularizo el *Dutch National Flag problem*, el cual es bastante simple y puede ser llevado al algoritmo Quicksort. En esencia, con esta solución se puede manejar el caso que se encuentre varios elementos similares al pivote a utilizar en la comparación. El siguiente gráfico demuestra en resumen el objetivo de este algoritmo:



3-way partitioning overview

Implemente este algoritmo denominado *Quicksort with 3-way partition*, compare con la implementación del Ejercicio 1, y determine cual versión da mejores resultados en Arrays con varios valores repetidos (considere que cada elemento puede tener más de 10 apariciones en el mismo Array a ordenar).

5.4 Ejercicio 4

Modifique el código del Ejercicio 1 para que permita utilizar en conjunto al algoritmo del Insertion Sort. El comportamiento general del algoritmo será utilizar Quicksort para ordenar grandes cantidades de elementos, cuando los subarray a ordenar se encuentre debajo de una cantidad M se procede a ordenar con el algoritmo de Insertion Sort. Determine cual es el valor óptimo de M que permita mejorar el proceso de ordenamiento (pruebe valores entre $0 \leq M \leq 30$). Realice un gráfico con el tiempo promedio en ordenar Arrays aleatorios de tamaños $N = 10^3$, $N = 10^4$ y $N = 10^6$

6 Entregables

Al finalizar el estudiante deberá:

- Elaborar un documento, en donde se registre la resolución de cada uno de los ejercicios planteados.
- En los ejercicios que involucran elaborar código, es necesario colocar el código en formato de texto (de colocar una imagen del código, se debe de incluir también el código en formato de texto)
- Agregar **solo los archivos fuente de los código desarrollados** en un archivo comprimido (LABXX.zip) (el nombre de cada archivo de código debe ser LABXX_EjercicioXX.cpp)

- Deberán de subir a la plataforma Classroom el documento elaborado en **formato PDF** (se recomienda el uso de *LaTeX*) y el archivo comprimido solo con los códigos elaborados.

7 Rúbrica de Evaluación

Rúbrica	**Cumple**	**Cumple con Observaciones**	**No cumple**
Informe: Desarrolla un informe, con un formato limpio y fácil de leer.	4	2	0
Ejercicios: Resuelve correctamente cada ejercicio	16	8	0
Errores ortográficos: Se descontará 0.5 puntos de encontrarse errores			

- **IMPORTANTE** En caso de copia o plagio o similares todos los alumnos implicados tendrán sanción en toda la evaluación del curso.

References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.