



Escuela de Ingeniería en Computadores

CE 4301 — Arquitectura de Computadores I

Taller Introductorio a Simics

**Taller 2**

Profesor: Luis Chavarria Zamora

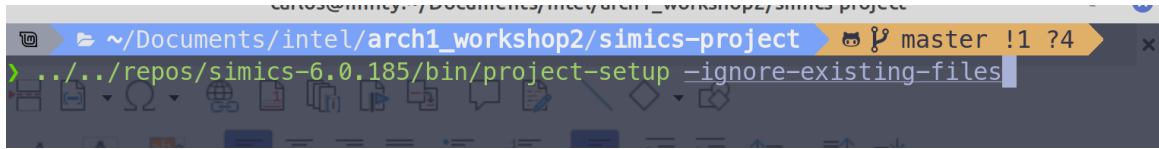
**Estudiantes:**

Carlos Andrés Mata Calderón

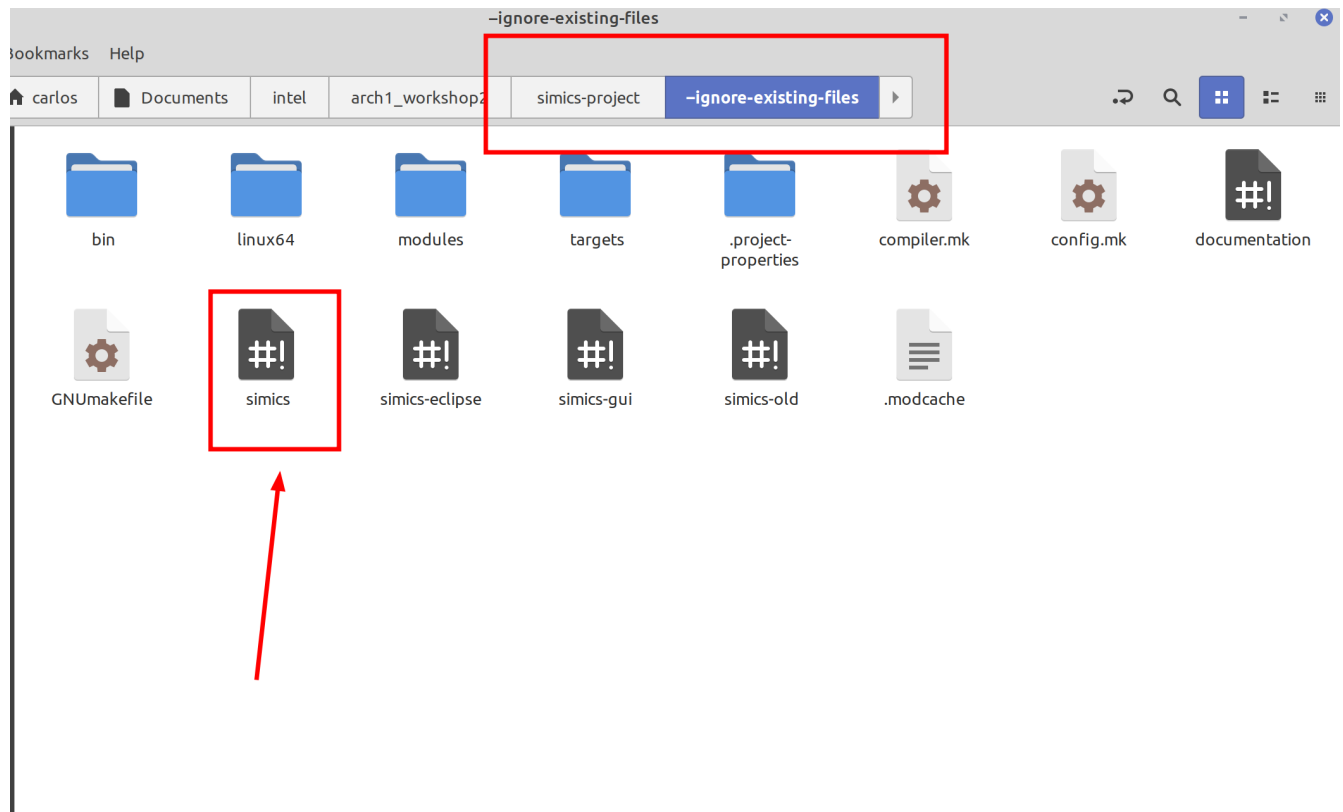
I Semestre 2024

# Ejercicio Practico

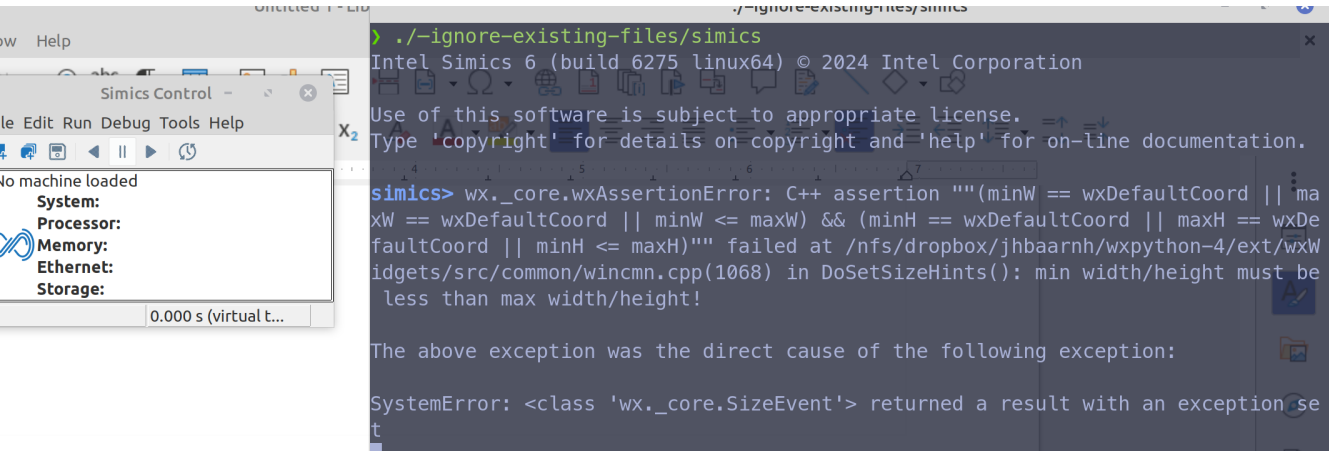
El simics fue instalado con el siguiente comando ``../../repos/simics-6.0.185/bin/project-setup --ignore-existing-files`` Que esta diciendno donde esta el simics y cual carpeta se guarda `--ignore-existing-files`



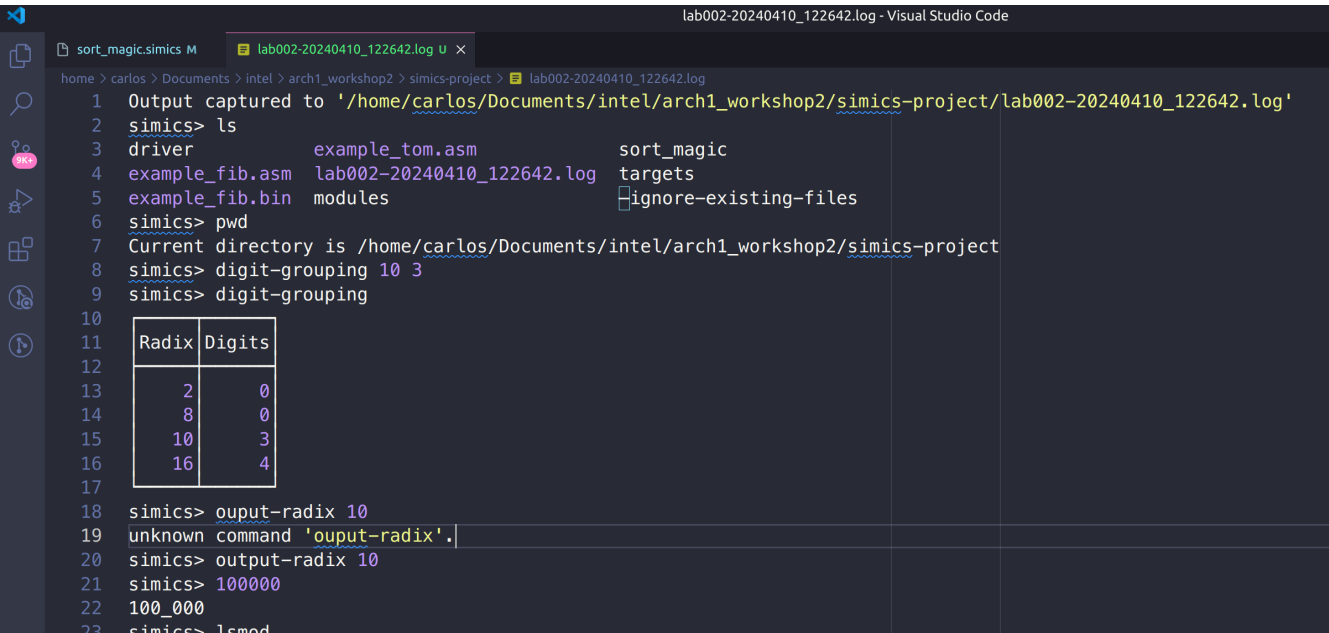
```
carlos@intel: ~/Documents/intel/arch1_workshop2/simics-project
> ../../repos/simics-6.0.185/bin/project-setup --ignore-existing-files
```



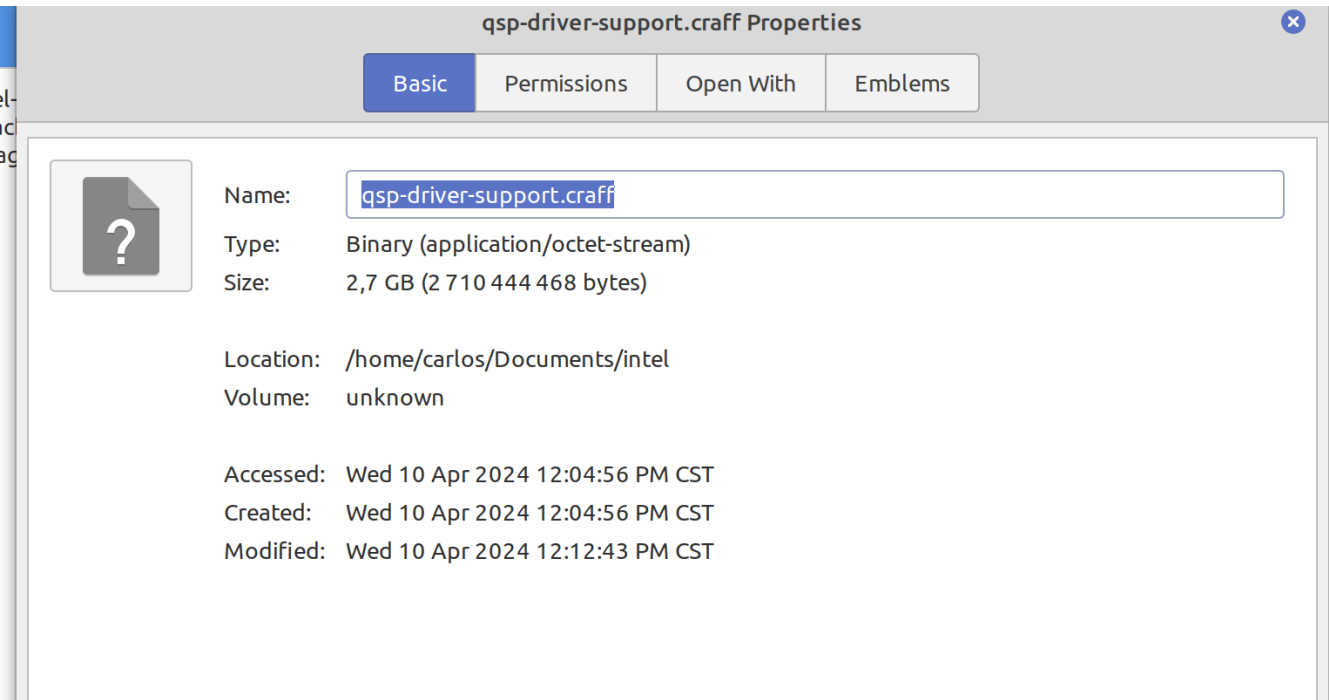
Se logra ejecutar simics se demuestra estando en la carpeta `simics-project`, con el comando se `./-ignore-existing-files/simics`.



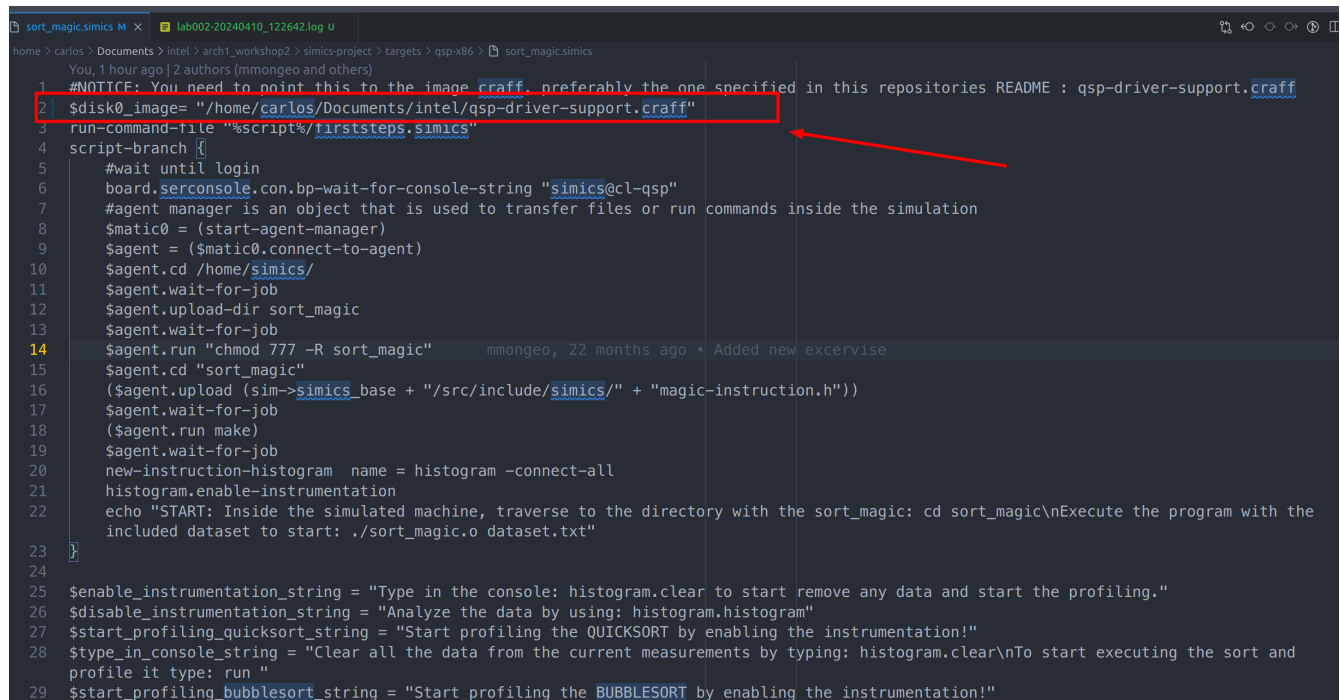
Y se demuestra estamos en esa carpeta usando ls, pwd y generando un log file de todos los comandos ejecutados dentro simics. Se adjunta una foto del log file generado



Ahora bien, se descargo el .craff en la siguiente ubicación ``/home/carlos/Documents/intel/qsp-driver-support.craff``

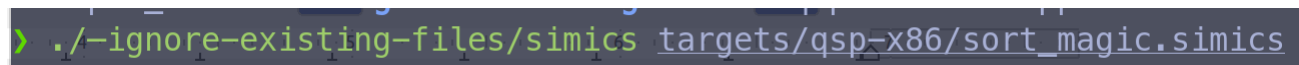


Se modifica el archivo `target/qsp-x86/sort-magic.simics` que esta ubicado dentro la carpeta `simics-project`.



```
1 #NOTICE: You need to point this to the image craff, preferably the one specified in this repositories README : qsp-driver-support.craff
2 $disk0_image= "/home/carlos/Documents/intel/qsp-driver-support.craff"
3 run-command-file "%script%/firststeps.simics"
4 script-branch {
5     #wait until login
6     board.serconsole.con.bp-wait-for-console-string "simics@cl-qsp"
7     #agent manager is an object that is used to transfer files or run commands inside the simulation
8     $matic0 = (start-agent-manager)
9     $agent = ($matic0.connect-to-agent)
10    $agent.cd /home/simics/
11    $agent.wait-for-job
12    $agent.upload-dir sort_magic
13    $agent.wait-for-job
14    $agent.run "chmod 777 -R sort_magic"
15    $agent.cd "sort_magic"
16    ($agent.upload (sim->simics_base + "/src/include/simics/" + "magic-instruction.h"))
17    $agent.wait-for-job
18    ($agent.run make)
19    $agent.wait-for-job
20    new-instruction-histogram name = histogram -connect-all
21    histogram.enable-instrumentation
22    echo "START: Inside the simulated machine, traverse to the directory with the sort_magic: cd sort_magic\nExecute the program with the
    included dataset to start: ./sort_magic.o dataset.txt"
23 }
24
25 $enable_instrumentation_string = "Type in the console: histogram.clear to start remove any data and start the profiling."
26 $disable_instrumentation_string = "Analyze the data by using: histogram.histogram"
27 $start_profiling_quicksort_string = "Start profiling the QUICKSORT by enabling the instrumentation!"
28 $type_in_console_string = "Clear all the data from the current measurements by typing: histogram.clear\nTo start executing the sort and
    profile it type: run "
29 $start_profiling_bubblesort_string = "Start profiling the BUBBLESORT by enabling the instrumentation!"
```

Se ejecuta el ultimo paso .9 que es ejecutar el `sort-magic.simics`



```
> ./-ignore-existing-files/simics targets/qsp-x86/sort_magic.simics
```

Y se obtiene el siguiente error

```
> ./-ignore-existing-files/simics targets/qsp-x86/sort_magic.simics
Intel Simics 6 (build 6275 linux64) © 2024 Intel Corporation
```

Use of this software is subject to appropriate license.  
Type 'copyright' for details on copyright and 'help' for on-line documentation.

```
Error: file '%script%/firststeps.simics' (/home/carlos/Documents/intel/arch1_workshop2/simics-project/targets/qsp-x86/firststeps.simics) not found in the Simics search path
```

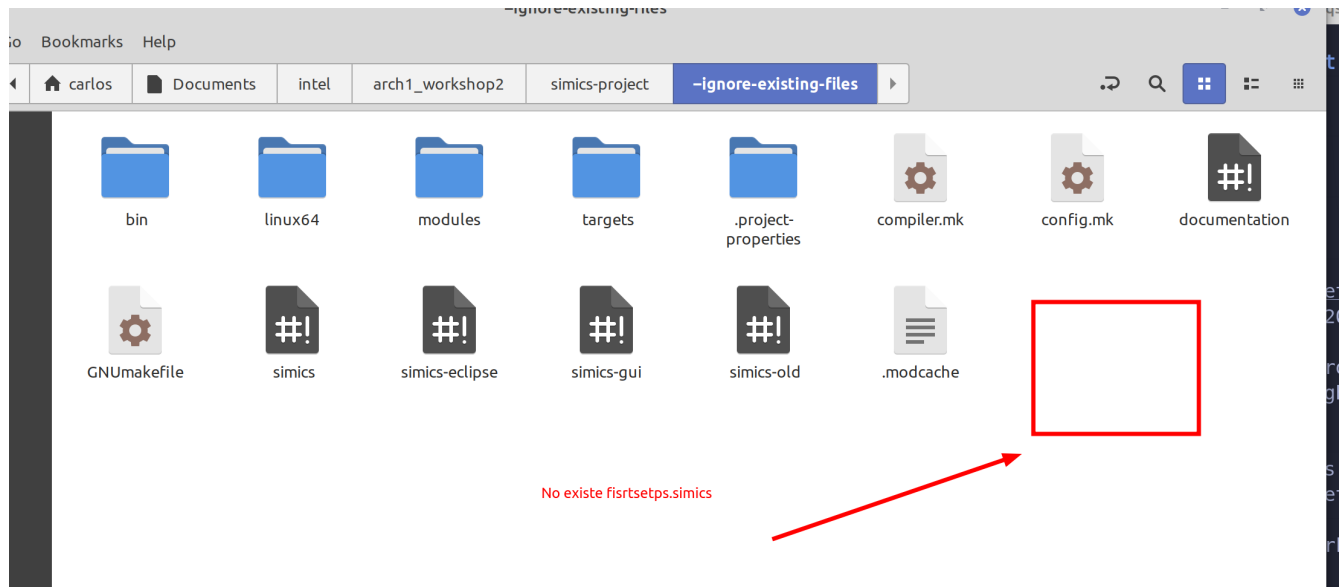
```
[/home/carlos/Documents/Intel/arch1_workshop2/simics-project/targets/qsp-x86/sort_magic.simics:3] error in 'run-command-file' command
Error - interrupting script.
```

```
simics> wx._core.wxAssertionError: C++ assertion ""(minW == wxDefaultCoord || maxW == wxDefaultCoord || minW <= maxW) && (minH == wxDefaultCoord || maxH == wxDefaultCoord || minH <= maxH)"" failed at /nfs/dropbox/jhbaarnh/wxpython-4/ext/wxWidgets/src/common/wincmn.cpp(1068) in DoSetSizeHints(): min width/height must be less than max width/height!
```

The above exception was the direct cause of the following exception:

```
SystemError: <class 'wx._core.SizeEvent'> returned a result with an exception set
```

El error básicamente dice que dentro de la carpeta donde se encuentra `simics` que en esta caso es `--ignore-existing-files` no se encuentra el archivo llamado `firststeps.simics`



**Se demuestra que efectivamente no existe ese archivo.** Note que este es un error del programa `target/qsp-x86/sort-magic.simics`, Por ello se concluye que el taller esta malo por esta razon no se puede ejecutar el archivo `sort-magic.simics`.

Se recomienda revisar y arreglar ese error, ya que en este caso es un error del programa del repositorio.

# Solución al Cuestionario

1. ¿Qué hace el código del `sort-magic.cpp` en las líneas `MAGIC(..)`, según lo leído en el paso 6?

El archivo `sort_magic.cpp`, las llamadas a `MAGIC(...)` con diferentes parámetros (como `QUICK_MAGIC`, `BUBBLE_MAGIC`, y `STOP_PROFILING`) se utilizan para controlar la instrumentación del código, específicamente para el perfilado de rendimiento durante la ejecución de las funciones de ordenamiento (`quicksort` y `bubblesort`).

La macro `MAGIC(...)` se define en el archivo "`magic-instruction.h`" y está diseñada para ejecutar instrucciones especiales que no alteran la lógica del programa pero permiten marcar específicamente dónde comienza y termina el perfilado de un bloque de código.

- `MAGIC(QUICK_MAGIC);` marca el inicio del perfilado para la ejecución del algoritmo de ordenamiento `quicksort`.
- `MAGIC(STOP_PROFILING);` se utiliza para detener el perfilado después de que `quicksort` ha completado su ejecución.
- Luego, antes de iniciar `bubblesort`, se llama nuevamente a `MAGIC(BUBBLE_MAGIC);` para marcar el inicio del perfilado para `bubblesort`.
- Finalmente, `MAGIC(STOP_PROFILING);` se invoca una vez más para detener el perfilado después de que `bubblesort` ha terminado.

2. ¿Cómo esto se relaciona con el archivo `targets/qsp-x86/sort-magic.simics`? Utilice el comando `help` desde la consola de Simics donde lo necesite, una pista es: `help bp.magic.wait-for`

La relación entre el código de `sort_magic.cpp` y el script `targets/qsp-x86/sort-magic.simics` se centra en el uso de instrucciones mágicas para controlar la instrumentación y el perfilado de ejecución en el simulador Simics. Las instrucciones mágicas, identificadas en `sort_magic.cpp` por las llamadas a `MAGIC(...)` con diferentes valores (como `QUICK_MAGIC`, `BUBBLE_MAGIC`, y `STOP_PROFILING`), se utilizan para marcar puntos específicos en la ejecución del programa que son de interés para el perfilado.

El archivo `sort-magic.simics` utiliza estas instrucciones mágicas como puntos de sincronización para controlar el flujo de ejecución del simulador y recolectar datos de perfilado en momentos específicos del programa `sort_magic.cpp`. Esto se logra mediante el uso de `bp.magic.wait-for`, una característica del administrador de puntos de ruptura en Simics, que permite pausar la ejecución del script hasta que ocurra una instrucción mágica específica.



La notación Big O de Quicksort en el caso promedio es  $O(n \log n)$ , mientras que en el peor caso es  $O(n^2)$ . Sin embargo, debido a su diseño y a la manera en que particiona el conjunto de datos, Quicksort es generalmente más rápido y eficiente en la práctica para conjuntos de datos grandes, asumiendo una buena elección del pivote. Por otro lado, BubbleSort tiene una complejidad temporal tanto en el promedio como en el peor caso de  $O(n^2)$ , lo que lo hace significativamente menos eficiente para conjuntos de datos grandes debido a su enfoque simple de comparaciones y swaps adyacentes.

En promedio, Quicksort debería ejecutarse más rápidamente que BubbleSort debido a su menor complejidad temporal en el caso promedio y su eficiencia general en la mayoría de los conjuntos de datos.

4. Guarde los histogramas de ambos algoritmos y adjúntelos en el entregable.

Esto es imposible por lo explica en la parte practica del taller.

5. ¿Sabiendo la complejidad de ambos algoritmos, cuál es el impacto en la cantidad de instrucciones ejecutadas en el rendimiento? Es decir, elabore en cuáles operaciones del procesador deberían de ser más costosas y por ende hacer que el programa tome más tiempo y donde hay la mayor diferencia entre ambos histogramas (un gráfico le ayudará a comparar mejor los valores).

Las operaciones del procesador que tienden a ser más costosas y, por ende, afectan el tiempo de ejecución del programa, incluyen accesos a memoria, especialmente si causan fallos de caché, y las instrucciones de comparación y salto condicional que son abundantes en los algoritmos de ordenamiento. En el caso de BubbleSort, el alto número de comparaciones y swaps (intercambios) de elementos adyacentes conlleva a numerosos accesos a memoria y recálculos de direcciones, lo cual es costoso en términos de tiempo de ejecución. Por otro lado, aunque Quicksort realiza también comparaciones y swaps, su naturaleza divide y vencerás reduce significativamente el número total de estas operaciones necesarias para ordenar el mismo número de elementos, resultando en un mejor aprovechamiento de la localidad de datos y, por lo tanto, menor impacto por fallos de caché y accesos a memoria.

6. ¿Cuál de las dos implementaciones de ordenamiento elegiría para un programa y por qué?

Elegiría la implementación de Quicksort para un programa debido a su eficiencia superior y menor complejidad computacional en comparación con BubbleSort, especialmente para conjuntos de datos grandes. Quicksort, en promedio, ofrece un rendimiento de  $O(n \log n)$ , lo que lo hace significativamente más rápido que BubbleSort, cuya eficiencia es  $O(n^2)$  tanto en el mejor como en el peor caso. Esta eficiencia se traduce en menos tiempo de procesamiento y un mejor uso de los recursos del sistema, lo que es crucial para aplicaciones que manejan grandes volúmenes de datos o requieren alta velocidad de procesamiento. Además, Quicksort es versátil y puede optimizarse para diferentes tipos de datos y condiciones iniciales, lo que lo hace una elección más adaptable y robusta para el ordenamiento en una amplia gama de aplicaciones.