# **15 DE ABRIL DE 2020**



# **PRÁCTICA 2:**

# **EXCLUSIÓN MUTUA**

## **SISTEMAS DISTRIBUIDOS 2019-2020**

DAMIÁN MALENO GONZÁLEZ

GEOVANNY RISCO CARRERA

# Decisiones de diseño

# Análisis de los gráficos

# Reparto de trabajo

Se utilizó un repositorio en Github para trabajar de manera conjunta. Se puede acceder mediante mediante el siguiente link <https://github.com/geovalexis/task1-SD.git>, aunque tal vez no tengas acceso porque es un repositorio privado. En caso de que sea necesario lo podemos hacer público o bien darte acceso como contribuidor.

En todo momento se trabajó de forma coordinada discutiendo y trabajando el diseño del programa asi como elaborando el código en Python utilizando *Visual Studio Code*. En general se puede decir que cada uno participó mayoritariamente en las siguientes funciones:

* Damian Maleno: función de inicialización y análisis de resultados.
* Geovanny Risco: funciones map y reduce, y decisiones de diseño.

# Preguntas

Respuestas a las siguientes preguntas sobre la exclusión mutua.

* Muchos algoritmos distribuidos como el propuesto es esta práctica requieren del uso de un proceso ***master***o **coordinador**. ¿En qué medida se pueden considerar realmente estos algoritmos como distribuidos? Justificar.

Cuando hablamos de algoritmos centralizados, normalmente existe un proceso fijo que actúa como *master* o coordinador. El hecho de que los procesos sean ejecutados en diferentes máquinas hace que el algoritmo pueda decirse distribuido. En los algoritmos distribuidos que no disponen de un proceso coordinador fijo, este es elegido mediante algoritmos de elección de líder entre los diferentes procesos que forman parte del algoritmo, entre estos podemos encontrar algoritmos de elección de líder aleatorios, en anillo, en malla… Por ello podemos afirmar que la presencia de un proceso *master* o coordinador en un algoritmo no hace que este sea más o menos distribuido.

* Ahora supón que la función ***master*** falla. ¿Esto siempre conlleva la violabilidad de la exactitud del algoritmo? ¿Si no, bajo qué circunstancias ocurre? ¿Existe alguna forma de evitar el problema y hacer el sistema tolerante a fallos del coordinador? Justificar.
* En el algoritmo propuesto, los permisos de escritura se otorgan en el orden en que son solicitados, de manera que no hay *slaves* que esperan infinitamente (*no starvation*)*.* ¿Si la función *master* eligiese las funciones *slave* de manera aleatoria, podrían las funciones *slave* sufrir *starvation*?

# Referencias

* Funcionamiento e implementación de pywren: <https://github.com/pywren/pywren-ibm-cloud>
* Programación en Python: <https://runestone.academy/runestone/books/published/pythonds/index.html>
* Librería de Python para las operaciones matemáticas: <https://docs.scipy.org/doc/numpy/index.html>
* Diapositivas, fórum y apuntes de clase.

# ANEXO I: Gráficos