Diego Velásquez Varela 202010004101– Steven Oviedo Aristizábal 202010028101

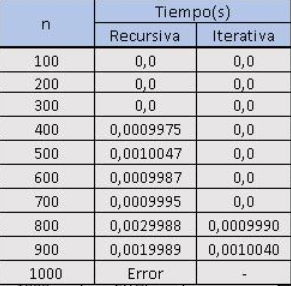
Reto #1 Análisis

La recursividad es la forma en la cual especificamos un proceso basado en su propia definición. En ciencias de la computación, podemos definirla como una forma de atajar y solventar problemas, resolver un problema mediante este método significa que la solución general, dependerá a su vez de las soluciones de pequeñas instancias del mismo problema.

Por otra parte, la iteración significa repetir varias veces un proceso con la intención de alcanzar una meta deseada, objetivo, o resultado. Una estructura iterativa, permite ejecutar una o varias instrucciones un número determinado de veces, o de forma indefinida siempre y cuando cumpla una condición.

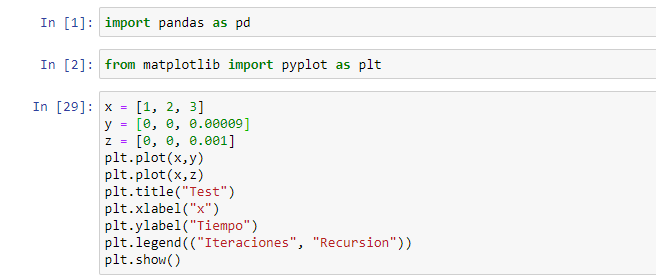
En el presente informe daremos solución a una serie de interrogantes, que determinarán cuando y bajo que condiciones se hace más util utilizar alguna de las estrategias mencionadas.

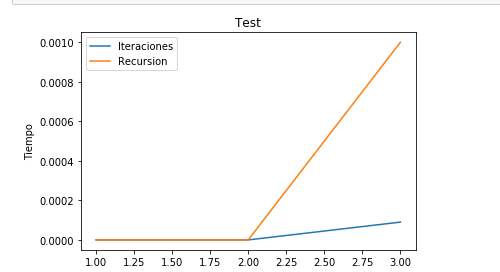
1. **Cuáles fueron los tiempos que obtuviste. Realice una tabla comparativa**



Observamos que, para N muy pequeños, el tiempo de ejecución es bastante rápido, al punto que es imperceptible para el IDE contar la cantidad de millonésimas de segundos que tardó el programa en dar la solución.

Conforme el número N va creciendo, podemos ver como el tiempo para la implementación iterativa se mantiene en **0,0,** pero el tiempo de la implementación recursiva empieza a tender al alza de una forma más rápida. El siguiente código implementado en JupiterNoteBook usando la librería matplotlib y pandas, nos da una ***representación visual*** de los tiempos que tarda la recursión vs los bucles o iteraciones





1. **Que problemas se presentaron y si se presentan por que se presentan**

En las primeras iteraciones, cuando se usaban n números muy bajos la maquina no era capaz de detectar las millonésimas de segundos. Por otra parte, cuando se utilizaban n muy altos (a partir de 1000) se excedía el limite de la pila creada con el método recursivo, saltando un error del tipo:

***RecursionError: maximum recursion depth exceeded in comparison.***

El cual señala que la profundidad de recursión máxima fue excedida (Se llamó a si mismo demasiadas veces) este problema puede solucionarse importando una librería del tipo:

***from functools import lru\_cache #least recently used cache, me añade la memorización en una línea***

***@lru\_cache(maxsize = 1000)***

Que conforma avanza la ejecución del problema, almacena valores tipo “Cache”, ahorrándole llamadas y tiempo al método recursivo.

*(Por efectos de tiempo esta implementación no fue realizada en el código, pero puede verse en nuestro taller de recursión con el método Fibonacci)*

1. **Que decisión tomarías al requerir implementar la función factorial. Cuál es la mejor opción entre la recursiva y la iterativa. ¿Por qué?**

Para este caso, la mejor decisión es utilizar la implementación iterativa, ya que además de ser más optima en cuanto al tiempo, no produce ningún tipo de error para N que son demasiado altos.

Sin embargo, el método recursivo puede refinarse para que no tenga este tipo de errores utilizando memorización, esto es, usar la librería lru\_cache que nos permite ir almacenando de forma paralela las llamadas recursivas (Así podemos aumentar de forma indefinida la profundidad de las llamadas que realicemos)

En conclusión, tanto la estrategia iterativa, como la recursiva son metodologías muy poderosas en la computación, para decidir cual es más conveniente, tendremos que realizar una lectura cuidadosa del tipo de problema que queremos solucionar 😊