P0 TIME

Ismael Crespo

February 2022

1 Introducción

Este reporte presenta los resultados de la medición de tiempo y memoria requerida para trabajar con matrices vectores de *n filas y n columnas* en R y Python. Los valores de n fueron seleccionados haciendo pruebas para determinar el máximo permitido por el equipo de computo utilizado.

2 Medición de tiempos y espacio

El equipo utilizado para la medición de rendimientos es una computadora portatil *Acer E5-573* con un Procesador *Intel Core i5 2.20GHz*,una memoria RAM instalada de 8.0 GB y un sistema operativo de 64 bits.

2.1 Medición en R.

Se crearon matrices de $2^n * 2^n$ con números aleataorios y se utilizaron las funciones sysetm.time y object.size Listing 1 para para medir el tiempo requerido para crear cada matriz y el espacio requerido en la memoria RAM. Los intervalos de valor n se encontraron experimentando uno a uno hasta encontrar un máximo permisible por el equipo de computo en en n=14. Los resultados se presentan en la tabla 1

n	2^n	Espacio en RAM (Bytes)	Tiempo (Seg)
2	4	344	0
3	8	728	0
4	16	2264	0
5	32	8408	0
6	64	32984	0
7	128	131288	0
8	256	524504	0
9	512	2097368	0.01
10	1024	8388824	0.04
11	2048	33554648	0.17
12	4096	134217944	0.79
13	8192	536871128	2.82
14	16384	2147483864	17.91

Table 1: Tiempo y espacio requerido para cada matriz

Listing 1: Codigo en R para medir rendimeintos

```
for(n in 2:14) {k=2^n; cat(k,system.time(matrix(runif(k*k),
nrow=k))[3], '\n')}
library(pryr)
for(n in 2:14) {k=2^n; cat(k,object_size(matrix(runif(k*k),
nrow=k)), '\n')}
```

2.2 Medición en Python.

Para medir el rendimiento en Python sin necesidad de usar una libreria adicional se utilizó un Listing 2, que guarda la hora antes de comenzar a ordenar número aleatorios en vectores de 2^n, para restarsela a la hora final y obtener el tiempo que tarda en hacer este procedimeintos. Los resultados se presentan en la tabla 2.

Listing 2: Codigo en Python para medir rendimeintos

```
from random import randint
desde = 1
hasta = 1000
menor = 15
mayor = 21
from time import time # traer libreria
for k in range(menor, mayor + 1):
    n = 2 ** k
    lista = [ randint(desde, hasta) for i in range(n) ]
    antes = time() # ver la hora
    lista.sort()
    despues = time() # volver ver la hora
    diferencia = despues - antes # segundos
    print('ordenar', n, 'elementos toma', diferencia, 'segundos')
```

n	2^n Tiempo (Seg)	
15	32768	0.0
16	65536	0.011695
17	131072	0.0325872
18	262144	0.0680267
19	524288	0.14938402
20	1048576	0.22246694
21	2097152	0.41567270

Table 2: Tiempo requerido para ordendar cada vector en Python