P0 TIME

Ismael Crespo

9 de febrero de 2022

1. Introducción

Este reporte presenta los resultados de la medición de tiempo y memoria requerida para trabajar con matrices y 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 cómputo 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 2.20 GHz*,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 `system.time` y `object.size` Listing **??** 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 el cuadro 1 y figura1

Cuadro 1: Tiempo y espacio requerido para cada matriz

n	2^n	Espacio en RAM (Bytes)	Tiempo (Seg)
2	4	344	—
3	8	728	—
4	16	2264	—
5	32	8408	—
6	64	32984	—
7	128	131288	—
8	256	524504	—
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

2.2. Medición en Python.

Para medir el rendimiento en Python sin necesidad de usar una libreria adicional se utilizó el Listing **??**,y en la grafica 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 estos procedimeinto, el espacio ocupado por cada vector se obtuvo utilizando la función *getsizeof*. Los resultados se presentan en la tabla 2.

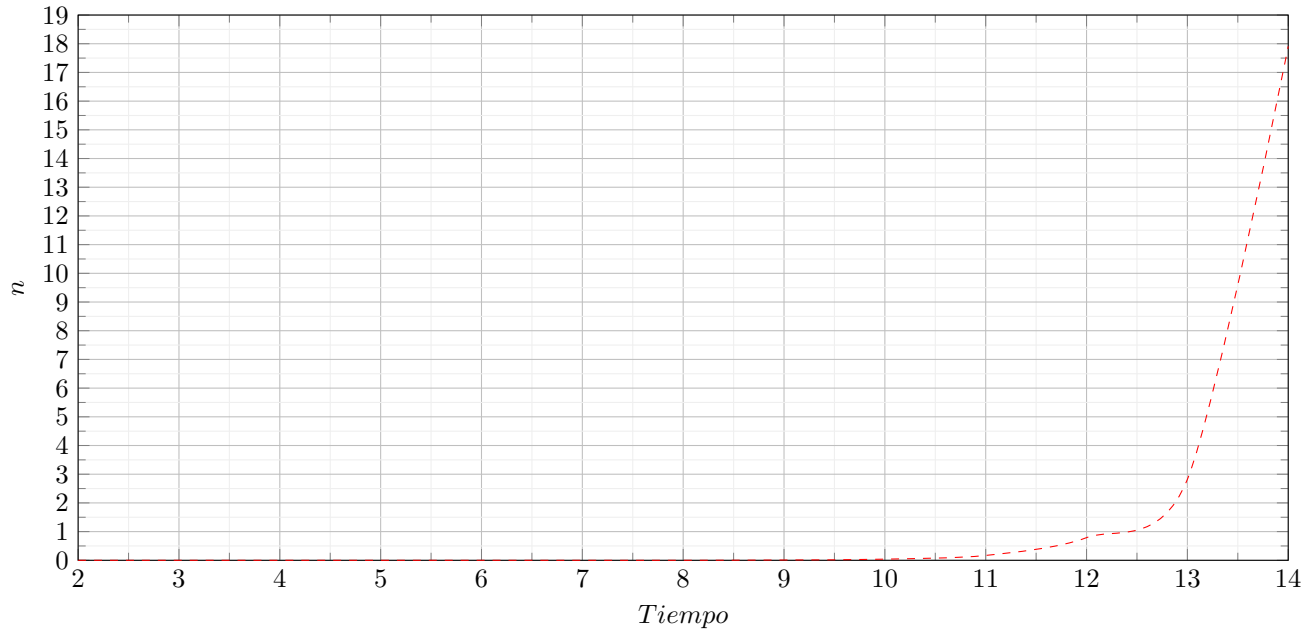
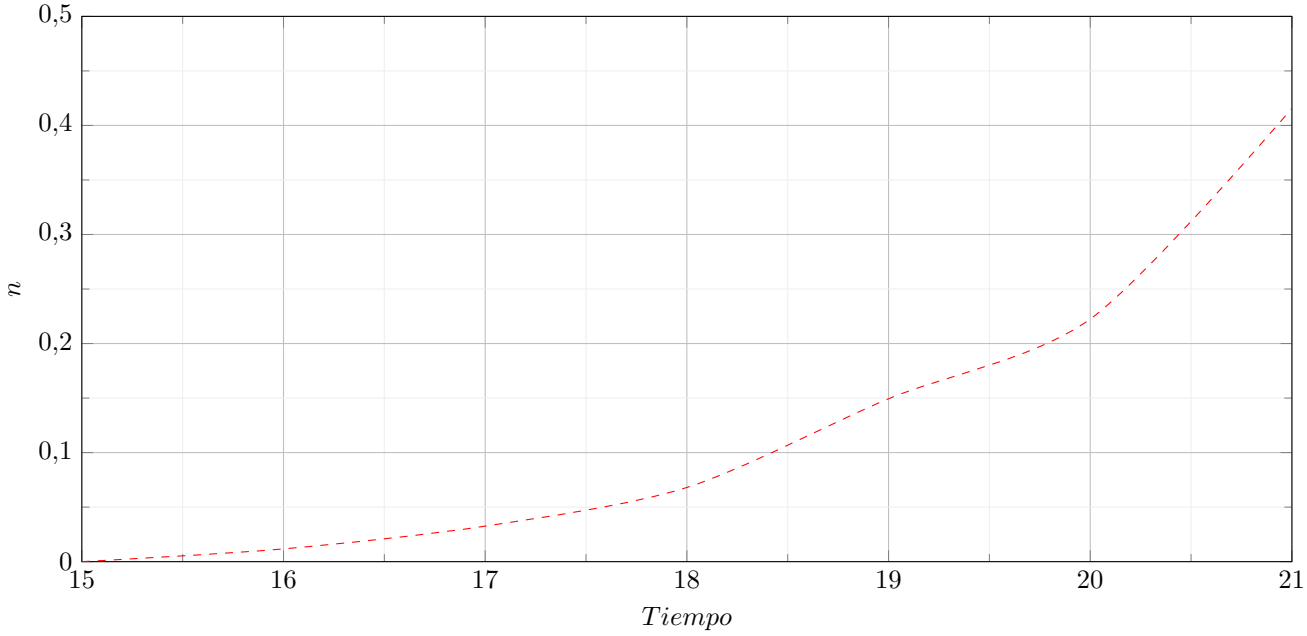


Figura 1: Tiempo requerido para ordenar una matriz de $2^n * 2^n$ elementos aleatorios en R

```
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')}
```

Listing 1: Codigo en R

Figura 2: Tiempo requerido para ordenar un vector de 2^n números aleatorios en Python



```python
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')

import sys
from sys import getsizeof
import numpy as np
from random import randint
menor=15
mayor=21
for k in range(menor, mayor + 1):
    n=2**k
    lista = [ randint(1, 1000) for i in range(n) ]
    espacio=getsizeof(lista)
    print('un vector de',n,'elementos ocupa',espacio,'bytes de memoria')
```

Listing 2: Codigo en Python

n	2^n	Tiempo [Seg]	Espacio [Bytes]
15	32768	—	277336
16	65536	0.011695	562488
17	131072	0.0325872	1140568
18	262144	0.0680267	2312472
19	524288	0.14938402	4688312
20	1048576	0.22246694	8448728
21	2097152	0.41567270	17128280

Cuadro 2: Tiempo requerido para ordenar cada vector en Python

3. Conclusiones

El equipo de computo utilizado presenta un numero maximo permitible para realizar trabajos matriciales y dependen de cada herramienta utilizada para progrmar así como de las tareas realizadas en paralelo. Es necesario conocer los estados optimos para sacar el máximo potencial al equipo.