Tarea 2 EDA - Informe

Cristóbal Nilo Garrido - cristobal.nilo@sansano.usm.cl

Agosto 2020

1. Introducción

En el curso ELO-320 nos hemos familiarizado con algoritmos de ordenamiento y su complejidad. Como tarea hemos codificado tanto una implementación de Insertion Sort como una implementación de Heap Sort. En el presente informe se presentarán los resultados comparativos de la ejecución de ambos algoritmos, para diferentes tamaños de entrada y en distinto hardware.

1.1. Hardware utilizado.

Web Server 1

- Sistema Operativo: Ubuntu 18.04
- CPU: 1 Núcleo
- Memoria RAM: 1 GB
- Dirección IP (host): 34.220.245.41

Web Server 2

- Sistema Operativo: Ubuntu 18.04
- CPU: 4 Núcleos
- Memoria RAM: 16 GB
- Dirección IP (host): 18.237.176.205

PC Personal

- Sistema Operativo: Arch Linux 5.6.13
- CPU: 4 Núcleos
- Memoria RAM: 8 GB

2. Resultados

A continuación se presentan los gráficos comparativos para cada hardware y algoritmo.

2.1. Gráficos

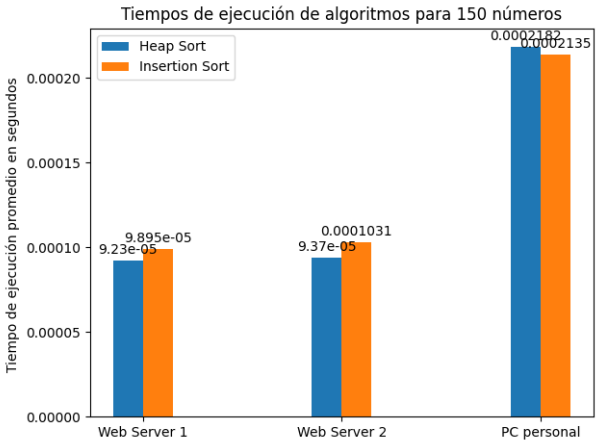

Figura 1: Gráfico comparativo entre algoritmos para 150 números en cada PC.

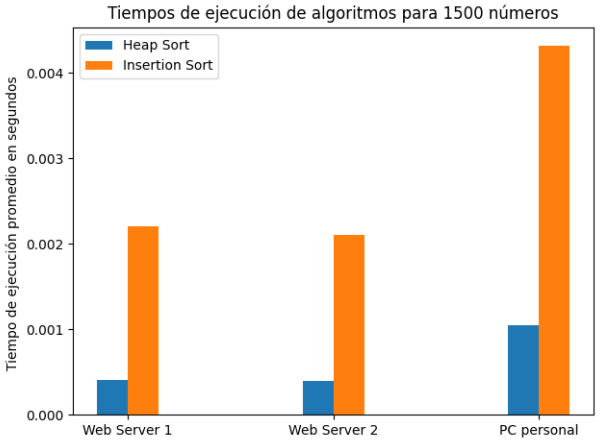

Figura 2: Gráfico comparativo entre algoritmos para 1500 números en cada PC.

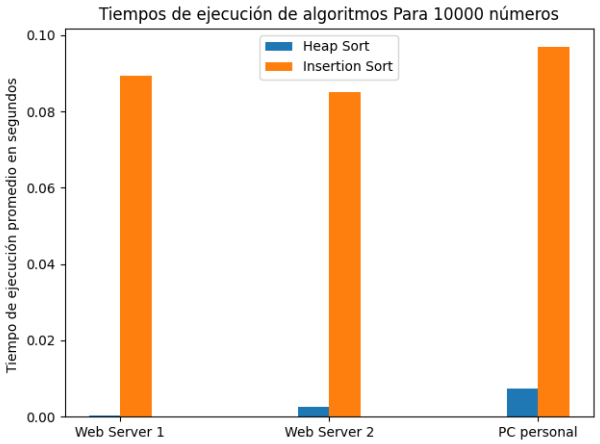

Figura 3: Gráfico comparativo entre algoritmos para 10000 números en cada PC.

2.2. Porcentaje de uso de CPU

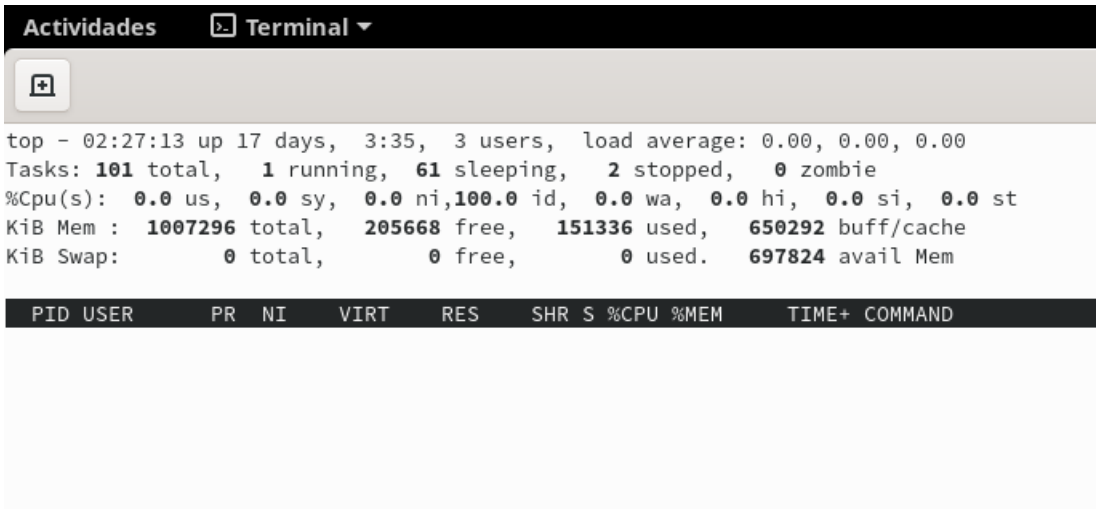

Figura 4: Uso de CPU: 0 % de Servidor 1

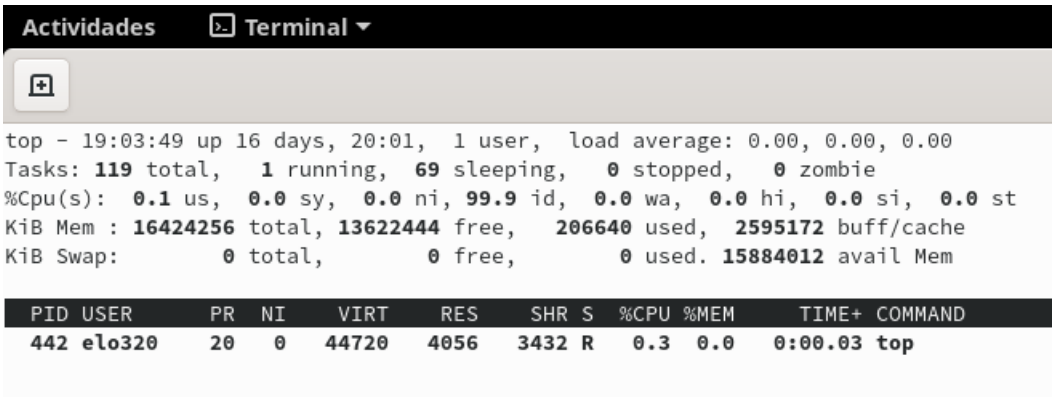

Figura 5: Uso de CPU: 0.1 % de Servidor 2

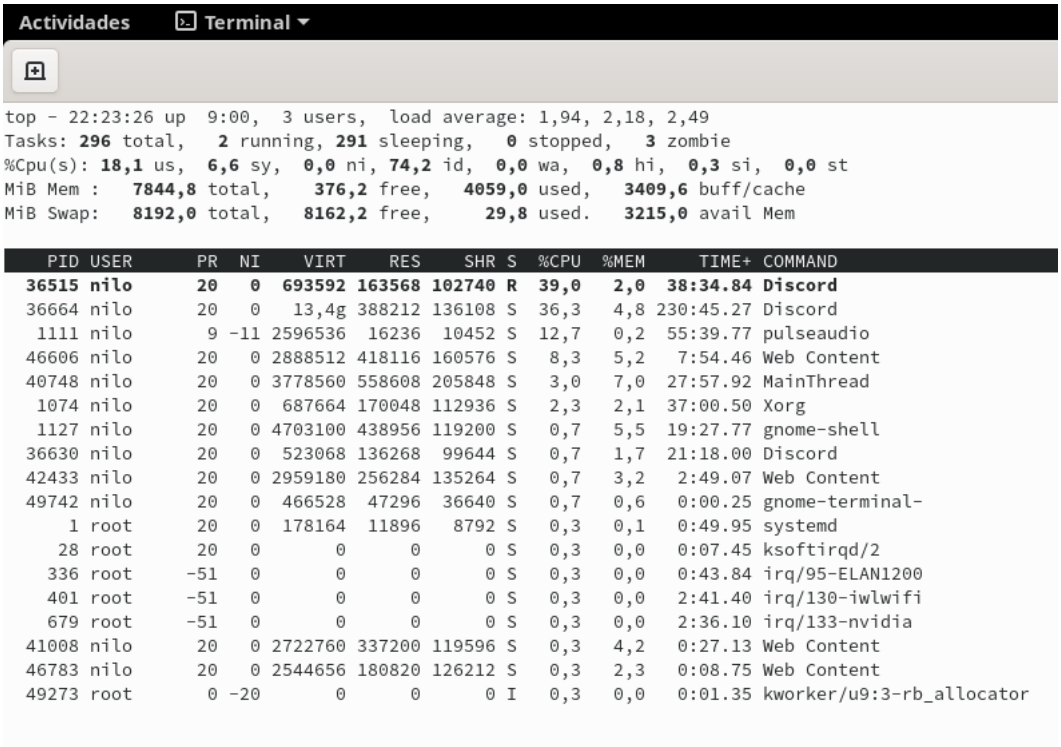

Figura 6: Uso de CPU: 19.1 % de PC Personal

2.3. Análisis de gráficos

Es notable la tendencia en los algoritmos. En todos los PC donde el algoritmo fue ejecutado, tanto Heap Sort como Insertion Sort poseen tiempos de ejecución muy similares para 150 números. Para 1500 números se nota una diferencia de el doble o más en los tiempos de ejecución siendo el menor tiempo para Heap Sort. finalmente esta tendencia se agudiza para los 10000 números, aumentando la diferencia en tiempo de ejecución siendo Insertion Sort mucho mayor. En términos de hardware para usos de CPU similares (Web Server 1 y Web Server 2) los tiempos de ejecución son bastante similares aún notando las diferencias en hardware presentes, de esto se puede desprender que para tareas elementales como son ls algoritmos de ordenamiento, es realmente difícil encontrar diferencias ya que incluso en hardware que pueden ser considerados limitados o básicos como el del servidor 1, son más que suficientes para correr estos algoritmos de manera eficiente y rápida. Diveros expertos dicen que en el estado del arte de la programación, hay una confianza ciega en la ley de moore, y la optimización de algoritmos se ha dejado de lado precisamente porque la ejecución dada la tecnologia actual es muy rápida. Ahora bien, cuando vamos a mi computador personal donde el uso de CPU fue de un 20 %, se ven tiempos considerablemente más elevados, esto debido a que como la CPU está en uso, el proceso también se ve ralentizado.

2.4. Análisis de complejidad

Si consideramos lo visto a través del curso, en el código entregado y la ejecución del algoritmo, en el enfoque de sift down es posible observar la recursividad en la llamada de sift down, al momento de construir el heap. Insertar un nodo a través de sift down es más eficiente que con sift up debido a que a medida que insertamos nodos con sift down, disminuye el tiempo debido a que estamos más cerca de las hojas, mientras que en caso contrario, al usar sift up el tiempo aumenta a cada nodo. En consecuencia de estas recursividades se pueden desprender ciertas relaciones de recurrencia al momento de construir el heap.

2.4.1. Análisis de complejidad teórica.

Teóricamente para heap sort se tienen las siguientes complejidades:

- Peor caso: $\mathcal{O}(n \log n)$
- Caso promedio: $\mathcal{O}(n \log n)$
- Mejor caso: $\mathcal{O}(n \log n)$

Así, es relevante indicar que la complejidad será la misma para cualquier arreglo entregado. Para poder comparar se mostrará la complejidad de Insertion Sort:

- Peor caso: $\mathcal{O}(n^2)$
- Caso promedio: $\mathcal{O}(n^2)$

- Mejor caso: $\mathcal{O}(n)$

Entendiendo que el mejor caso es que el arreglo esté ordenado, podemos usar los inputs para calcular complejidades.

- Comparamos 150^2 y $150 * \log 150$ obteniendo que la ejecución de insertion para un input de 150 elementos es "sólo"20 veces superior a la de heapsort.
- Comparamos 1500^2 y $1500 * \log 1500$ obteniendo que la ejecución de insertion para un input de 1500 elementos es 142 veces superior a la de heapsort.
- Finalmente para el caso de 10000 elementos de entrada 10000^2 es 752 veces superior a $10000 * \log 10000$

De estas comparaciones se desprende que a mayor tamaño de entrada, las diferencias de tiempos de ejecución crecerá rápidamente, lo que es consecuente con la notación asintótica usada en el análisis de complejidad. Se puede concluir que mientras mayor sea el input estas diferencias aumentarán aún más porque el crecimiento de Insertion Sort es claramente superior. Además se concluye que a pesar de que los tiempos de ejecución teóricos sean 20 veces mayores para Insertion Sort en el input de 150 elementos, no significa una diferencia significativa en los ordenes de magnitud que usamos en la medición.

3. Conclusiones.

Es posible de manera práctica y a través de codificación sencilla, entender los contenidos vistos en el curso, que es consecuente con la teoría vista. No deja de ser llamativo la idea de poder ver materializada las diferencias en los tiempos de ejecución.

4. Documentación extra (Tabulación de los datos.

4.0.1. Tablas para input de 150 datos

150 Números	Heap Sort	Insertion Sort
1	0,000100	0.000106
2	0,000087	0.000098
3	0,000093	0.000097
4	0,000091	0.000098
5	0,000092	0.000097
6	0,000108	0.000099
7	0,000095	0.000099
8	0,000100	0.000097
9	0,000088	0.000097
10	0,000092	0.000102
11	0,000091	0.000095
12	0,000091	0.000108
13	0,000096	0.000090
14	0,000091	0.000102
15	0,000090	0.000099
16	0,000091	0.000098
17	0,000088	0.000098
18	0,000082	0.000089
19	0,000089	0.000114
20	0,000091	0.000096
PROMEDIO	0,0000923	0,00009895

Figura 7: Tiempos de ejecución para ambos algoritmos en el Servidor Web 1

150 Números	Heap Sort	Insertion Sort
1	0,000100	0.000100
2	0,000095	0.000100
3	0,000096	0.000101
4	0,000092	0.000099
5	0,000090	0.000102
6	0,000092	0.000112
7	0,000091	0.000100
8	0,000094	0.000099
9	0,000092	0.000101
10	0,000096	0.000100
11	0,000095	0.000102
12	0,000097	0.000138
13	0,000095	0.000103
14	0,000093	0.000098
15	0,000092	0.000100
16	0,000091	0.000101
17	0,000092	0.000099
18	0,000098	0.000101
19	0,000092	0.000101
20	0,000091	0.000105
PROMEDIO	0,0000937	0.0001031

Figura 8: Tiempos de ejecución para ambos algoritmos en el Servidor Web 2

150 Números	Heap Sort	Insertion Sort
1	0,000268	0,000089
2	0,000085	0,000203
3	0,000262	0,000256
4	0,000086	0,000259
5	0,000112	0,000283
6	0,000264	0,000258
7	0,000265	0,000143
8	0,000269	0,000256
9	0,000270	0,000259
10	0,000121	0,000287
11	0,000267	0,000281
12	0,000271	0,000108
13	0,000263	0,000289
14	0,000268	0,000081
15	0,000077	0,000285
16	0,000123	0,000080
17	0,000284	0,000284
18	0,000274	0,000082
19	0,000271	0,000292
20	0,000264	0,000194
PROMEDIO	0,0002182	0,0002135

Figura 9: Tiempos de ejecución para ambos algoritmos en el PC personal

4.0.2. Tablas para input de 1500 datos

1500 Números	Heap Sort	Insertion Sort
1	0,000424	0,002183
2	0,000395	0,002212
3	0,000400	0,002159
4	0,000392	0,002201
5	0,000396	0,002251
6	0,000397	0,002305
7	0,000399	0,00225
8	0,000404	0,002203
9	0,000425	0,00229
10	0,000396	0,002164
11	0,000411	0,002171
12	0,000395	0,002218
13	0,000400	0,002195
14	0,000417	0,002204
15	0,000449	0,002157
16	0,000399	0,002119
17	0,000418	0,002272
18	0,000398	0,002191
19	0,000395	0,002166
20	0,000394	0,002175
PROMEDIO	0,0004052	0,0022043

Figura 10: Tiempos de ejecución para ambos algoritmos en el Servidor Web 1

1500 Números	Heap Sort	Insertion Sort
1	0,000453	0,002116
2	0,000398	0,002125
3	0,000399	0,002101
4	0,000394	0,002087
5	0,000392	0,002181
6	0,000400	0,002073
7	0,000395	0,002181
8	0,000398	0,002091
9	0,000395	0,002089
10	0,000398	0,002083
11	0,000394	0,002088
12	0,000397	0,002084
13	0,000397	0,002112
14	0,000394	0,002109
15	0,000395	0,002124
16	0,000397	0,002098
17	0,000393	0,002082
18	0,000393	0,002081
19	0,000391	0,002107
20	0,000409	0,002146
PROMEDIO	0,0003991	0,0021079

Figura 11: Tiempos de ejecución para ambos algoritmos en el Servidor Web 2

1500 Números	Heap Sort	Insertion Sort
1	0,001196	0,005976
2	0,001196	0,004891
3	0,001186	0,004142
4	0,001204	0,005976
5	0,000357	0,005979
6	0,000373	0,001677
7	0,001210	0,005857
8	0,001188	0,005976
9	0,001178	0,005378
10	0,001194	0,004138
11	0,001184	0,005973
12	0,001191	0,00538
13	0,001191	0,001713
14	0,001192	0,005973
15	0,001199	0,00602
16	0,000836	0,004126
17	0,001181	0,001738
18	0,001181	0,001814
19	0,000347	0,001767
20	0,001197	0,001813
PROMEDIO	0,0010491	0,0043154

Figura 12: Tiempos de ejecución para ambos algoritmos en el PC personal

4.0.3. Tablas para input de 10000 datos

10000 Números	Heap Sort	Insertion Sort
1	0,002671	0,088971
2	0,002623	0,090528
3	0,002656	0,089038
4	0,002657	0,089200
5	0,002651	0,088785
6	0,002670	0,089427
7	0,002668	0,088999
8	0,002753	0,089408
9	0,002643	0,088625
10	0,002682	0,088996
11	0,002662	0,090168
12	0,002644	0,089972
13	0,002730	0,090741
14	0,002689	0,089078
15	0,002663	0,088802
16	0,002645	0,088568
17	0,002665	0,089480
18	0,002671	0,089657
19	0,002707	0,089505
20	0,002670	0,091079
PROMEDIO	0,002671	0,08945135

Figura 13: Tiempos de ejecución para ambos algoritmos en el Servidor Web 1

10000 Números	Heap Sort	Insertion Sort
1	0,002654	0,084817
2	0,002647	0,084773
3	0,002650	0,085030
4	0,002663	0,084760
5	0,002599	0,084721
6	0,002688	0,085439
7	0,002603	0,085120
8	0,002678	0,084786
9	0,002616	0,085184
10	0,002590	0,085834
11	0,002648	0,085100
12	0,002600	0,085127
13	0,002644	0,084883
14	0,002605	0,084949
15	0,002635	0,084763
16	0,002773	0,084786
17	0,002636	0,084986
18	0,002633	0,085088
19	0,002608	0,085097
20	0,002623	0,084836
PROMEDIO	0,00263965	0,08500395

Figura 14: Tiempos de ejecución para ambos algoritmos en el Servidor Web 2

10000 Números	Heap Sort	Insertion Sort
1	0,007957	0,099862
2	0,007947	0,095064
3	0,007960	0,092385
4	0,007951	0,095593
5	0,007988	0,112619
6	0,002322	0,08597
7	0,008129	0,091857
8	0,007988	0,096392
9	0,007970	0,099215
10	0,007952	0,095751
11	0,007964	0,098284
12	0,007445	0,096494
13	0,007959	0,096526
14	0,007975	0,097419
15	0,007983	0,097965
16	0,007962	0,096341
17	0,008001	0,098685
18	0,002394	0,097465
19	0,007953	0,098585
20	0,007956	0,095666
PROMEDIO	0,0073878	0,0969069

Figura 15: Tiempos de ejecución para ambos algoritmos en el PC personal