Análisis Reto 2

Juan Camilo Gonzalez, 201911030 , jc.gonzalezc20
Luis Francisco Escobar, 202020323, lf.escobarg1

Encontramos que a pesar de ser un poco más complejas de implementar las tablas hash son sustancialmente más rápidas que las listas, como ARRAY_LIST y LINKED_LIST, además de permitir organizar los datos de mejor manera haciendo que escribir los códigos sea más corto.

Algo negativo destacable es el uso de la memoria, ya que con las tablas de hash se notó un mayor uso, lo que puede implicar que para computadoras con recursos limitados sea lento de ejecutar.

Esto se ve en las siguientes tablas:

	Máquina 1	**Máquina 2**
Procesadores	AMD Ryzen 5 3600 3.60 GHz	Intel(core) i7-7500U 3.50 GHz
Memoria RAM (GB)	16.0 GB	4.0 GB
Sistema Operativo	Windows 10 Pro 64-bits	Windows 10 Pro 64-bits

Maquina 1

Tamaño de la muestra (ARRAYLIST)	**Insertion Sort [ms]**	**Selection Sort [ms]**	**Shell Sort [ms]**	**Quick Sort [ms]**	**Merge Sort [ms]**
1000	**500.0**	**640.62**	**31.25**	15.62	31.25
2000	**2031.25**	**2625.0**	**62.5**	46.87	46.87
4000	**8375.0**	**10515.62**	**140.62**	109.37	93.75
8000	**33015.61**	**42234.37**	**375.0**	234.37	218.75
16000	**138453.12**	**173781.25**	**812.5**	437.5	437.5
32000	**572375.0**	**695180.4**	**1953.12**	984.37	968.75
64000	**2858437.5**	**3193543.8**	**4640.62**	2171.87	2031.25
128000	**11433960.3**	**12784175.2**	**11046.87**	4687.5	4500.0
256000	**45735678.2**	**51096880.1**	**28500.0**	11046.87	9468.75

Tamaño de la muestra (LINKED_LIST)	Insertion Sort [ms]	Selection Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
1000	**30562.5**	**27078.12**	**1546.87**	1015.62	156.25
2000	**255078.12**	**224328.13**	**7328.12**	5640.62	625.0
4000	**2129484.37**	**1861923.479**	**34687.5**	25765.62	2593.75
8000	**17674720.27**	**15453964.88**	**176093.75**	120203.12	10078.125
16000	**146700178.2**	**128267908.5**	**880468.75**	478125.0	39468.75
32000	**1217611479**	**1064623640**	**4402343.8**	2204171.87	166781.25
64000	**10106175280**	**8836376214**	**22011719**	+15min	709562.5
128000	**83881254821**	**73341922579**	**110058594**	+ 15min	+15min
256000	**6.96214E+11**	**6.08738E+11**	**550292969**	+15min	+15min

Carga de Catálogo PROBING

Factor de Carga (PROBING)	Consumo de Datos [kB]	Tiempo de Ejecución [ms]
0.30	575966.239	21653.181
0.50	575966.239	21426.026
0.80	575966.239	21305.097

Carga de Catálogo CHAINING

Factor de Carga (CHAINING)	Consumo de Datos [kB]	Tiempo de Ejecución [ms]
2.00	575974.036	21600.977
4.00	575974.036	21488.472
6.00	575974.036	21514.682

Maquina 2

Tamaño de la muestra (ARRAYLIST)	Insertion Sort [ms]	Selection Sort [ms]	Shell Sort [ms]	Quick Sort [ms]	Merge Sort [ms]
1000	937.5	1390.62	46.875	15.625	46.875

2000	4437.5	7328.12	109.37	46.875	62.5
4000	15812.5	17109.37	218.75	187.5	156.25
8000	65156.25	79406.25	687.5	593.75	468.75
16000	290609.37	317421.87	1312.5	743.375	984.375
32000	1162437.5	1269265.62	3171.87	1578.125	1671.875
64000	4649750.37	5077062.5	7203.12	4390.625	3640.625
128000	18599001.5	20308250.38	14406.2	8015.625	9484.375
256000	74396006.5	81233001.5	28812.5	22312.5	15843.75

Tamaño de la muestra (LINKED_LIST)	**Insertion Sort [ms]**	**Selection Sort [ms]**	**Shell Sort [ms]**	**Quick Sort [ms]**	**Merge Sort [ms]**
1000	87953.12	45128.37	3764.5	3640.625	453.125
2000	700359.37	361027.5	1822.5	18203.125	1812.5
4000	3501791.87	2888220.37	9,412	91015.625	7250.125
8000	17508959.38	23105763.13	47060.87	455078.125	29000.5
16000	87544796.88	184846105.5	235,304,37	2275390.625	116002.625
32000	437723984.4	1478768845	1176521.87	+15min	464010.5
64000	2188619922	11830150759	5882609.37	+15min	+15min
128000	8754479688	94641206076	29413046.88	+15min	+15min
256000	35017918750	7.5713E+11	147065234.5	+15min	+15min

Carga de Catálogo CHAINING

Factor de Carga (CHAINING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución [ms]**
2.00	570069.371	38772.271
4.00	570069.371	49669.785
6.00	570069.371	50267.950

Carga de Catálogo PROBING

Factor de Carga (PROBING)	**Consumo de Datos [kB]**	**Tiempo de Ejecución [ms]**
0.30	570069.605	60672.120

0.50	570069.449	42707.427
0.80	570069.371	50006.709

Gracias a estas tablas nosotros concluimos, teniendo en cuenta tiempos de ejecución, uso de memoria y implementación en le código, que lo más conveniente es usar las tablas de hash con el manejo de colisiones CHAINING o ARRAY_LIST y algoritmos para ordenar el merge sort. Por esto utilizamos lo anterior en nuestro Reto 2.