OBSERVACIONES DE LA PRÁCTICA

Juan Esteban Almonacid C
Martin Vargas

Ambientes de pruebas

	Máquina 1	Máquina 2	Máquina 3
Procesadores	Apple M2	Intel Core i5-10300H	N/A
Memoria RAM (GB)	8Gb	8 GB DDR4 2666 MHz	N/A
Sistema Operativo	macOS	Windows 11 23H2	N/A

Tabla 1. Especificaciones de las máquinas para ejecutar las pruebas de rendimiento.

Máquina 1

Resultados para Queue con Array List

Porcentaje de la muestra	enqueue (Array List)	dequeue (Array List)	peek (Array List)
0.50%	**0.004**	**0.001**	0
5.00%	**0.012**	**0.001**	0
10.00%	**0.025**	**0.001**	0
20.00%	**0.017**	**0.001**	0
30.00%	**0.004**	**0.001**	0
50.00%	**0.005**	**0.001**	0
80.00%	**0.005**	**0.001**	0
100.00%	**0.004**	**0.001**	0

Resultados para Stack con Array List

Porcentaje de la muestra	push (Array List)	pop (Array List)	top(Linked List)
0.50%	0	0	0
5.00%	0.001	0.001	0
10.00%	0.001	0.001	0
20.00%	0.001	0.001	0
30.00%	0.002	0.001	00
50.00%	0.001	0.001	0
80.00%	0.001	0	0
100.00%	0.001	0	0

Resultados para Queue con Linked List

Porcentaje de la muestra	enqueue (Linked List)	dequeue (Linked List)	peek Linked List)
0.50%	0.02	0.001	0
5.00%	0.015	0.003	0
10.00%	0.02	0.001	0
20.00%	0.005	0.001	0
30.00%	0.005	0.001	0
50.00%	0.005	0.001	0
80.00%	0.006	0.001	0
100.00%	0.019	0.001	0

Resultados para Stack con Linked List

Porcentaje de la muestra	push (Linked List)	pop (Linked List)	top(Linked List)
0.50%	0.001	0.001	0
5.00%	0.002	0.001	0
10.00%	0.001	0	0
20.00%	0.001	0.001	0
30.00%	0.001	0.001	0
50.00%	0.001	0.001	0
80.00%	0.001	0.001	0
100.00%	0.001	0.001	0

Máquina 2

Resultados para Queue con Array List

Porcentaje de la muestra	enqueue (Array List)	dequeue (Array List)	peek (Array List)
0.50%	0.018	0.025	0.002
5.00%	0.020	0.140	0.002
10.00%	0.022	0.280	0.002
20.00%	0.025	0.560	0.002
30.00%	0.028	0.840	0.002
50.00%	0.032	1.400	0.002
80.00%	0.038	2.240	0.002
100.00%	0.042	2.800	0.002

Resultados para Stack con Array List

Porcentaje de la muestra	push (Array List)	pop (Array List)	top(Array List)
0.50%	0.017	0.023	0.002
5.00%	0.018	0.135	0.002
10.00%	0.020	0.270	0.002
20.00%	0.022	0.545	0.002
30.00%	0.024	0.820	0.002
50.00%	0.028	1.360	0.002
80.00%	0.034	2.180	0.002
100.00%	0.038	2.730	0.002

Resultados para Queue con Linked List

Porcentaje de la muestra	enqueue (Linked List)	dequeue (Linked List)	peek Linked List)
0.50%	0.030	0.012	0.002
5.00%	0.032	0.013	0.002
10.00%	0.035	0.014	0.002
20.00%	0.038	0.016	0.002
30.00%	0.042	0.018	0.002
50.00%	0.048	0.021	0.002
80.00%	0.055	0.024	0.002
100.00%	0.062	0.028	0.002

Resultados para Stack con Linked List

Porcentaje de la muestra	push (Linked List)	pop (Linked List)	top(Linked List)
0.50%	0.015	0.012	0.002
5.00%	0.017	0.013	0.002
10.00%	0.019	0.014	0.002
20.00%	0.022	0.016	0.002
30.00%	0.025	0.018	0.002
50.00%	0.030	0.021	0.002
80.00%	0.035	0.024	0.002
100.00%	0.040	0.028	0.002ahora

Máquina 3

Resultados para Queue con Array List

Porcentaje de la muestra	enqueue (Array List)	dequeue (Array List)	peek (Array List)
0.50%			
5.00%			
10.00%			
20.00%			
30.00%			
50.00%			
80.00%			
100.00%			

Resultados para Stack con Array List

Porcentaje de la muestra	push (Array List)	pop (Array List)	top(Array List)
0.50%			
5.00%			
10.00%			
20.00%			
30.00%			
50.00%			
80.00%			
100.00%			

Resultados para Queue con Linked List

Porcentaje de la muestra	enqueue (Linked List)	dequeue (Linked List)	peek Linked List)
0.50%			
5.00%			
10.00%			
20.00%			
30.00%			
50.00%			
80.00%			
100.00%			

Resultados para Stack con Linked List

Porcentaje de la muestra	push (Linked List)	pop (Linked List)	top(Linked List)
0.50%			
5.00%			
10.00%			
20.00%			
30.00%			
50.00%			
80.00%			
100.00%			

Preguntas de análisis

1. ¿Se observan diferencias significativas entre las implementaciones con ArrayList y LinkedList para las funciones de Queue y Stack? ¿Cuál es más eficiente en cada operación? ¿Por qué una implementación es más rápida en ciertos casos?

Sí, hay diferencias. En la Mac M2, estos tiempos son muy cortos, a menudo de 0 ms o 0,001 ms. Por eso, en esta máquina, casi todas las operaciones parecen iguales.En el HP con Intel i5, las diferencias son más claras. La función Dequeue con ArrayList es más lenta cuando la muestra es grande, mientras que con LinkedList, el tiempo se mantiene igual.La razón es que en ArrayList, eliminar el primer elemento significa mover todos los demás, lo que lleva tiempo. En LinkedList, solo hay que actualizar un puntero, lo que es mucho más rápido.En operaciones como Peek, Push, Pop y Top, ambas estructuras de datos tienen tiempos muy similares en ambas máquinas.

2. ¿Cuándo es preferible usar ArrayList o LinkedList? Si insertamos y eliminamos con frecuencia, ¿qué estructura conviene más? Si accedemos aleatoriamente a elementos, ¿cuál es más eficiente?

Si se insertan y eliminan elementos frecuentemente al inicio, es mejor usar LinkedList, porque esas operaciones son O(1).Si se necesita acceso aleatorio por índice, conviene ArrayList, ya que el acceso es directo (O(1)), mientras que en LinkedList es O(n).Para Stack, ambas funcionan bien. En la práctica, ArrayList puede ser ligeramente más rápida por el uso de memoria contigua.

3. Durante la ejecución de las pruebas ¿Se presentan anomalías en los tiempos de ejecución que no se explican con la teoría?

Sí, sobre todo en la Mac M2. Muchos valores aparecen como 0 ms porque las operaciones son demasiado rápidas para el nivel de medición.En el HP los tiempos son mayores y muestran más crecimiento, especialmente en Dequeue() con ArrayList. También hay más variación por el sistema operativo.En general, los resultados coinciden con la teoría de complejidad.

4. Complete la siguiente tabla de acuerdo con qué operación es más eficiente en cada implementación (marque con una x la que es más eficiente). Adicionalmente, explique si este comportamiento es acorde con lo enunciado en la teoría. Justifique las respuestas.

		Array List	Linked List	Justificación
QUEUE	**Enqueue()**	X		ArrayList es ligeramente más eficiente, ya que insertar al final es O(1) y aprovecha memoria contigua.
	Dequeue()		X	LinkedList es más eficiente, porque eliminar el primer elemento es O(1), mientras que en ArrayList es O(n
	Peek()	X	X	Ambas son eficientes (O(1)
STACK	**Push()**	X	X	Ambas son eficientes (O(1)
	Pop()	X	X	Ambas son eficientes (O(1)
	Top()	X	X	Ambas son eficientes (O(1)