OBSERVACIONES DE LA PRÁCTICA

Valentina España Cuellar 202414079

Juan Sebastian Cortes Cortes 202411692

Tomas Alarcón Martinez Troncoso 202420126

**Ambientes de pruebas**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Máquina 1 | Máquina 2 | Máquina 3 |
| Procesadores | AMD Ryzen 5 4500U | Intel R Core Ultra 7 | Intel Core i7-1255U |
| Memoria RAM (GB) | 8 GB | 32GB | 32GB |
| Sistema Operativo | Windows 11 | Windows 11 | Windows 11 |

*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% | 0.09 | 0.162 | 0.007 |
| 5.00% | 0.322 | 0.389 | 0.006 |
| 10.00% | 0.507 | 0.627 | 0.004 |
| 20.00% | 1.383 | 2.094 | 0.006 |
| 30.00% | 2.072 | 2.75 | 0.008 |
| 50.00% | 3.716 | 9.284 | 0.008 |
| 80.00% | 7.683 | 14.537 | 0.008 |
| 100.00% | 5.788 | 19.528 | 0.009 |

**Resultados para Stack con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Linked List) |
| 0.50% | 0.032 | 0.749 | 0.004 |
| 5.00% | 0.192 | 0.218 | 0.002 |
| 10.00% | 0.39 | 0.457 | 0.004 |
| 20.00% | 1.023 | 0.892 | 0.003 |
| 30.00% | 1.870 | 2.25 | 0.009 |
| 50.00% | 3.33 | 3.345 | 0.009 |
| 80.00% | 5.678 | 5.619 | 0.008 |
| 100.00% | 7.084 | 5.74 | 0.021 |

**Resultados para Queue con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| 0.50% | 1.393 | 1.917 | 0.026 |
| 5.00% | 0.473 | 0.345 | 0.004 |
| 10.00% | 0.795 | 0.633 | 0.005 |
| 20.00% | 1.832 | 5.967 | 0.005 |
| 30.00% | 2.589 | 6.307 | 0.006 |
| 50.00% | 3.948 | 4.154 | 0.009 |
| 80.00% | 15.244 | 5.868 | 0.006 |
| 100.00% | 7.454 | 12.547 | 0.007 |

**Resultados para Stack con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| 0.50% | 0.199 | 0.038 | 2.595 |
| 5.00% | 0.398 | 26.562 | 0.063 |
| 10.00% | 0.629 | 95.128 | 0.103 |
| 20.00% | 1.713 | 371.788 | 0.225 |
| 30.00% | 3.178 | 893.723 | 0.55 |
| 50.00% | 1582.839 | 3420.067 | 0.583 |
| 80.00% | 7.265 | 9401.708 | 1.242 |
| 100.00% | 9.947 | 17624.424 | 0.863 |

**Máquina 2**

**Resultados para Queue con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| 0.50% | **0.159 [ms]** | **0.050 [ms]** | **0.007 [ms]** |
| 5.00% | **0.394 [ms]** | **0.324 [ms]** | **0.006 [ms]** |
| 10.00% | **0.809 [ms]** | **0.732 [ms]** | **0.005 [ms]** |
| 20.00% | **1.544 [ms]** | **1.296 [ms]** | **0.003 [ms]** |
| 30.00% | **2.156 [ms]** | **2.308 [ms]** | **0.003 [ms]** |
| 50.00% | **7.558 [ms]** | **6.024 [ms]** | **0.004 [ms]** |
| 80.00% | **52.130 [ms]** | **3.914 [ms]** | **0.002 [ms]** |
| 100.00% | **15.795 [ms]** | **0.003 [ms]** | **12.349 [ms]** |

**Resultados para Stack con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Array List) |
| 0.50% | **0.032 [ms]** | **0.033 [ms]** | **0.006 [ms]** |
| 5.00% | **0.197 [ms]** | **0.220 [ms]** | **0.006 [ms]** |
| 10.00% | **0.724 [ms]** | **0.739 [ms]** | **0.009 [ms]** |
| 20.00% | **0.759 [ms]** | **0.831 [ms]** | **0.002 [ms]** |
| 30.00% | **2.287 [ms]** | **2.336 [ms]** | **0.004 [ms]** |
| 50.00% | **3.526 [ms]** | **4.380 [ms]** | **0.005 [ms]** |
| 80.00% | **2.289 [ms]** | **3.201 [ms]** | **0.002 [ms]** |
| 100.00% | **4.463 [ms]** | **7.614 [ms]** | **0.004 [ms]** |

**Resultados para Queue con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| 0.50% | **0.091 [ms]** | **0.061 [ms]** | **0.005 [ms]** |
| 5.00% | **0.237 [ms]** | **0.223 [ms]** | **0.002 [ms]** |
| 10.00% | **1.177 [ms]** | **1.133 [ms]** | **0.004 [ms]** |
| 20.00% | **1.484 [ms]** | **1.289 [ms]** | **0.002 [ms]** |
| 30.00% | **3.667 [ms]** | **3.509 [ms]** | **0.003 [ms]** |
| 50.00% | **3.275 [ms]** | **2.357 [ms]** | **0.002 [ms]** |
| 80.00% | **62.022 [ms]** | **3.975 [ms]** | **0.002 [ms]** |
| 100.00% | **13.884 [ms]** | **12.676 [ms]** | **0.004 [ms]** |

**Resultados para Stack con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| 0.50% | **0.045 [ms]** | **0.045 [ms]** | **0.005 [ms]** |
| 5.00% | **0.224 [ms]** | **0.271 [ms]** | **0.002 [ms]** |
| 10.00% | **0.711 [ms]** | **0.973 [ms]** | **0.003 [ms]** |
| 20.00% | **0.765 [ms]** | **0.809 [ms]** | **0.003 [ms]** |
| 30.00% | **2.237 [ms]** | **1.675 [ms]** | **0.005 [ms]** |
| 50.00% | **1.693 [ms]** | **1.536 [ms]** | **0.004 [ms]** |
| 80.00% | **6.383 [ms]** | **5.229 [ms]** | **0.005 [ms]** |
| 100.00% | **8.664 [ms]** | **7.378 [ms]** | **0.006 [ms]** |

**Máquina 3**

**Resultados para Queue con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| 0.50% | **0.03** | **0.019** | **0.001** |
| 5.00% | **0.459** | **0.432** | **0.003** |
| 10.00% | **1.06** | **1.376** | **0.002** |
| 20.00% | **2.06** | **3.425** | **0.003** |
| 30.00% | **3.482** | **3.699** | **0.002** |
| 50.00% | **6.481** | **7.008** | **0.009** |
| 80.00% | **4.971** | **4.846** | **0.001** |
| 100.00% | **57.333** | **6.323** | **0.001** |

**Resultados para Stack con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Array List) |
| 0.50% | **0.019** | **0.449** | **0.002** |
| 5.00% | **0.115** | **0.131** | **0.001** |
| 10.00% | **0.234** | **0.274** | **0.002** |
| 20.00% | **0.614** | **0.535** | **0.002** |
| 30.00% | **1.122** | **1.350** | **0.005** |
| 50.00% | **1.998** | **2.007** | **0.005** |
| 80.00% | **3.407** | **3.371** | **0.005** |
| 100.00% | **4.250** | **3.444** | **0.013** |

**Resultados para Queue con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| 0.50% | **0.836** | **1.15** | **0.016** |
| 5.00% | **0.284** | **0.207** | **0.002** |
| 10.00% | **0.477** | **0.38** | **0.003** |
| 20.00% | **1.099** | **3.58** | **0.003** |
| 30.00% | **1.553** | **3.784** | **0.004** |
| 50.00% | **2.369** | **2.492** | **0.005** |
| 80.00% | **9.146** | **3.521** | **0.004** |
| 100.00% | **4.472** | **7.528** | **0.004** |

**Resultados para Stack con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| 0.50% | **0.13** | **0.14** | **0.002** |
| 5.00% | **0.348** | **0.325** | **0.003** |
| 10.00% | **0.715** | **0.75** | **0.002** |
| 20.00% | **0.862** | **0.825** | **0.003** |
| 30.00% | **1.436** | **1.837** | **0.003** |
| 50.00% | **2.865** | **2.705** | **0.006** |
| 80.00% | **3.002** | **2.884** | **0.001** |
| 100.00% | **2.502** | **2.528** | **0.002** |

**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, existen diferencias significativas entre las implementaciones con ArrayList y LinkedList para las funciones de Queue y Stack, dependiendo de la eficiencia en cada operación. LinkedList es más eficiente en Queue porque la eliminación al inicio e insertar al final es O(1), en cambio, para ArrayList la eliminación al inicio es O(n). ArrayList sería más eficiente en Stack en caso de que el SingleList no tuviera puntero al final. Por lo tanto, LinkedList es más adecuada para Queue, pero para el Stack se podría usar ambas, aunque ArrayList tiene mejor localidad espacial en Stack.

1. ¿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?

Todo depende de la funcionalidad que se le vaya a dar a las funciones, si se requiere insertar y eliminar datos con frecuencia, sería preferible usar linkedlist si se eliminan datos desde el inicio y se añaden datos al final, pero sería más eficiente usar arraylist si se busca acceder datos aleatoriamente o si se busca eliminar y añadir elementos en diferente orden.

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

Si, en algunos casos sin conexión alguna, se presentan tiempos de respuesta demasiado elevados comparándolos con el resto de tiempos y la tendencia de los mismos, que no puede ser explicada por medio de la complejidad algorítmica, sino que es independiente de cada prueba. Podría ser alguna afectación externa del dispositivo en el cual se realizaron las pruebas.

1. 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 | En un ArrayList, insertar al final (enqueue) es O(1) amortizado, pero si necesita redimensionarse puede ser O(n). En un LinkedList, insertar al final es O(1) constante si mantenemos un puntero al final. Para colas, LinkedList es generalmente preferible. |
| **Dequeue()** |  | X | En ArrayList, eliminar del principio (dequeue) requiere desplazar todos los elementos una posición, lo que es O(n). En LinkedList, eliminar del inicio es O(1) porque solo implica reasignar el puntero head. |
| **Peek()** | X |  | Ambos pueden implementar peek en O(1), pero ArrayList tiene mejor localidad espacial para acceso directo al primer elemento, lo que puede resultar en mejor rendimiento práctico debido a la caché del procesador. |
| STACK | **Push()** | X |  | Para stacks, push es insertar al final, y ArrayList tiene mejor rendimiento O(1) amortizado. Aunque LinkedList también puede ser O(1), ArrayList tiene mejor localidad espacial y menos sobrecarga por nodo. |
| **Pop()** | X |  | Eliminar del final en ArrayList es O(1). En LinkedList sería O(n) si solo tenemos puntero al inicio, o O(1) si mantenemos un puntero al final, pero seguiría teniendo peor rendimiento práctico que ArrayList. |
| **Top()** | X |  | Acceder al último elemento es O(1) en ArrayList. En LinkedList sería O(n) si solo tenemos acceso al inicio, o O(1) con doble puntero, pero ArrayList tiene mejor localidad espacial. |