OBSERVACIONES DE LA PRÁCTICA

Juliana Sofía Rodríguez Morales - 202421552

Juan Andrés Lozada Barragán - 202510410

Maria Clara Quijano Acevedo- 202420069

# **Ambientes de pruebas**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Máquina 1 | Máquina 2 | Máquina 3 |
| Procesadores | 11th Gen Intel(R) 2.80 GHz | 1.6 GHz Dual-Core Intel Core i5 | 11th Gen Intel(R) 3.00 Hz |
| Memoria RAM (GB) | 32.0 GB | 8 GB | 8 GB |
| Sistema Operativo | 64 bits (x64) | MacOS | 64 bits (x64) |

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

# **Máquina 1 – Juliana Rodríguez**

## **Resultados para Queue con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| 0.50% | **0.030** | **0.032** | **0.004** |
| 5.00% | **0.279** | **0.258** | **0.006** |
| 10.00% | **0.310** | **0.534** | **0.00** |
| 20.00% | **0.612** | **5.038** | **0.002** |
| 30.00% | **0.907** | **9.582** | **0.002** |
| 50.00% | **1.517** | **26.338** | **0.002** |
| 80.00% | **2.455** | **61.536** | **0.002** |
| 100.00% | **3.049** | **93.571** | **0.002** |

## **Resultados para Stack con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Linked List) |
| 0.50% | **0.017** | **0.039** | **0.004** |
| 5.00% | **0.182** | **0.314** | **0.003** |
| 10.00% | **0.290** | **0.612** | **0.001** |
| 20.00% | **0.351** | **0.748** | **0.001** |
| 30.00% | **0.526** | **1.104** | **0.001** |
| 50.00% | **1.032** | **1.964** | **0.002** |
| 80.00% | **1.771** | **3.250** | **0.001** |
| 100.00% | **2.082** | **3.966** | **0.001** |

## **Resultados para Queue con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| 0.50% | **0.104** | **0.040** | **0.005** |
| 5.00% | **5.085** | **0.206** | **0.002** |
| 10.00% | **20.735** | **0.488** | **0.003** |
| 20.00% | **71.399** | **0.934** | **0.002** |
| 30.00% | **169.401** | **1.647** | **0.002** |
| 50.00% | **469.804** | **2.749** | **0.003** |
| 80.00% | **1224.864** | **4.511** | **0.003** |
| 100.00% | **1914.111** | **5.692** | **0.002** |

## **Resultados para Stack con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| 0.50% | **0.084** | **0.146** | **0.003** |
| 5.00% | **3.758** | **6.059** | **0.001** |
| 10.00% | **16.775** | **23.417** | **0.003** |
| 20.00% | **73.37** | **101.207** | **0.002** |
| 30.00% | **169.029** | **215.294** | **0.002** |
| 50.00% | **459.447** | **625.612** | **0.002** |
| 80.00% | **1238.896** | **1643.945** | **0.002** |
| 100.00% | **1908.368** | **2517.238** | **0.003** |

# **Máquina 2 – Maria Clara**

## **Resultados para Queue con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| 0.50% | **0.549** | **0.155** | **0.008** |
| 5.00% | **21.701** | **0.918** | **0.091** |
| 10.00% | **52.559** | **1.196** | **0.004** |
| 20.00% | **175.137** | **2.434** | **0.004** |
| 30.00% | **359.651** | **4.16** | **0.005** |
| 50.00% | **953.367** | **6.93** | **0.005** |
| 80.00% | **2656.308** | **18.363** | **0.005** |
| 100.00% | **4475.097** | **23.3** | **0.005** |

## **Resultados para Stack con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Array List) |
| 0.50% | **0.229** | **0.14** | **0.004** |
| 5.00% | **12.718** | **0.863** | **0.042** |
| 10.00% | **35.951** | **1.388** | **0.003** |
| 20.00% | **137.945** | **2.778** | **0.004** |
| 30.00% | **328.228** | **4.941** | **0.004** |
| 50.00% | **964.167** | **9.196** | **0.004** |
| 80.00% | **2334.975** | **11.004** | **0.004** |
| 100.00% | **3752.249** | **14.654** | **0.003** |

## **Resultados para Queue con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| 0.50% | **0.675** | **0.137** | **0.006** |
| 5.00% | **26.192** | **1.452** | **0.075** |
| 10.00% | **58.163** | **1.313** | **0.003** |
| 20.00% | **164.164** | **2.652** | **0.005** |
| 30.00% | **343.891** | **4.044** | **0.004** |
| 50.00% | **1132.398** | **8.336** | **0.004** |
| 80.00% | **2751.853** | **10.98** | **0.003** |
| 100.00% | **4821.664** | **17.939** | **0.005** |

## **Resultados para Stack con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| 0.50% | **0.243** | **0.324** | **0.004** |
| 5.00% | **20.334** | **62.921** | **0.026** |
| 10.00% | **35.627** | **33.227** | **0.003** |
| 20.00% | **139.897** | **130.422** | **0.004** |
| 30.00% | **318.918** | **315.593** | **0.003** |
| 50.00% | **2008.992** | **820.364** | **0.003** |
| 80.00% | **2659.76** | **2481.39** | **0.003** |
| 100.00% | **4085.763** | **3868.256** | **0.003** |

# **Máquina 3 – Juan Andrés**

## **Resultados para Queue con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| 0.50% | 0.487 | 0.049 | 0.008 |
| 5.00% | 14.402 | 0.786 | 0.022 |
| 10.00% | 37.896 | 1.157 | 0.005 |
| 20.00% | 125.201 | 4.622 | 0.003 |
| 30.00% | 258.949 | 9.850 | 0.007 |
| 50.00% | 867.276 | 82.803 | 0.006 |
| 80.00% | 2126.793 | 76.896 | 0.003 |
| 100.00% | 3219.216 | 119.596 | 0.003 |

## **Resultados para Stack con Array List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Array List) |
| 0.50% | 0.113 | 0.042 | 0.003 |
| 5.00% | 7.531 | 1.763 | 0.049 |
| 10.00% | 31.306 | 0.753 | 0.003 |
| 20.00% | 120.591 | 1.567 | 0.003 |
| 30.00% | 258.811 | 1.585 | 0.007 |
| 50.00% | 845.500 | 2.918 | 0.009 |
| 80.00% | 2083.351 | 6.649 | 0.003 |
| 100.00% | 3182.815 | 0.002 | 0.002 |

## **Resultados para Queue con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| 0.50% | 0.151 | 0.053 | 0.005 |
| 5.00% | 7.794 | 0.375 | 0.004 |
| 10.00% | 39.519 | 0.774 | 0.007 |
| 20.00% | 467.802 | 4.936 | 0.006 |
| 30.00% | 309.381 | 2.354 | 0.003 |
| 50.00% | 834.251 | 3.900 | 0.015 |
| 80.00% | 2238.014 | 9.855 | 0.003 |
| 100.00% | 3666.856 | 0.004 | 9.271 |

## **Resultados para Stack con Linked List**

|  |  |  |  |
| --- | --- | --- | --- |
| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| 0.50% | 0.108 | 0.199 | 0.003 |
| 5.00% | 12.180 | 13.688 | 0.005 |
| 10.00% | 32.381 | 40.583 | 0.016 |
| 20.00% | 390.359 | 542.884 | 0.007 |
| 30.00% | 318.370 | 2.354 | 0.003 |
| 50.00% | 842.212 | 1138.496 | 0.042 |
| 80.00% | 2147.691 | 2863.973 | 0.002 |
| 100.00% | 3943.408 | 4579.526 | 0.003 |

# **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?

En el caso de la máquina 1, la diferencia entre ArrayList y LinkedList fue grande, siendo mejor el ArrayList para Colas y Pilas. La razón de porque una implementación es más rápida en ciertos casos es debido a las propiedades que tiene. Para la máquina 2, también es mejor ArrayList en la mayoría de casos menos en top(), ya que esta devuelve el último elemento en la lista, la cual se puede acceder con ArrayList. En el caso del Array al ser una lista continua se accede fácilmente a las posiciones mientras que un SingleLinked solo se conoce el siguiente, primero y último.

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?

Si se inserta o elimina con frecuencia lo mejor es usar un LinkedList ya que su complejidad algorítmica será de O(1) en la mayoría de los casos, ya que sería cambiar punteros y guardar variables temporales, no obstante, su complejidad es de O(n) cuando se pide eliminar/agregar en medio de la lista ya que se debe hacer un recorrido. Mientras que en un Array cuando se elimina y agrega en la mayoría de los casos los elementos dentro siempre deben moverse para completar espacios o abrir espacios, por lo que su complejidad es O(n) en todos los casos (incluso en el agregar de últimas porque se considera el peor caso que el array esté lleno y se deba dimensionar de nuevo el array y copiar elementos).

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

En el caso de la máquina 1 la lista SingleLinked se demoró mucho más que cuando se usó Array para insertar y eliminar un elemento en la Pila/Cola. Lo anterior, según la teoría, debería ser al revés.

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 | X | En ArrayList, agregar al final es amortizado O(1). En LinkedList, insertar al final también es O(1) si hay referencia al tail. Ambas son eficientes. |
| **Dequeue()** |  | X | En ArrayList, remover el primer elemento requiere mover todos los demás (O(n)). En LinkedList, eliminar el primer nodo es O(1). |
| **Peek()** | X | X | En ambos es O(1): en ArrayList accediendo al índice 0 y en LinkedList accediendo al head. |
| STACK | **Push()** | X |  | Ambos tienen complejidad O(1), pero el array list tomó menos tiempo para mostrar el resultado. |
| **Pop()** | X |  | Como mencionado en la teoría, es más eficiente utilizar un Array List ya se puede aceder a las posiciones con mayor facilidad. También es O(1). |
| **Top()** |  | X | Comparando los tiempos, se puede observar que esta función toma menor tiempo si se utiliza un linked list, esto se debe a que se sabe la posición final de la list. |