OBSERVACIONES DE LA PRÁCTICA

Estudiante 1 Cod 202321247

Estudiante 2 Cod 202421543

Estudiante 3 Cod XXXX

# **Ambientes de pruebas**

|  | Máquina 1 | Máquina 2 | Máquina 3 |
| --- | --- | --- | --- |
| Procesadores | 1,1 GHz Intel Core i5 de cuatro núcleos | 3.40 GHz Intel i1-14700K de 20 nucleos | Apple M3 |
| Memoria RAM (GB) | 8 GB 3733 MHz LPDDR4X | 32 GB 6000 MT/s DDR5 | 8 GB |
| Sistema Operativo | MacOS | Windows | MacOS |

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

# **Máquina 1**

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

| Tamaño de la muestra (ARRAY\_LIST) | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| --- | --- | --- | --- |
| 50,00 | 0.151 | 0.556 | 0.006 |
| 500,00 | 1.414 | 25.342 | 0.030 |
| 1000,00 | 2.464 | 84.145 | 0.005 |
| 2000,00 | 4.568 | 233.389 | 0.006 |
| 3000,00 | 9.574 | 481.467 | 0.006 |
| 5000,00 | 12.284 | 1.314.842 | 0.005 |
| 8000,00 | 16.661 | 3.242.715 | 0.007 |
| 10000,00 | 21.079 | 5.586.194 | 0.005 |

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

| Tamaño de la muestra (ARRAY\_LIST) | push (Array List) | pop (Array List) | top(Array List) |
| --- | --- | --- | --- |
| 50,00 | 0.133 | 0.276 | 0.006 |
| 500,00 | 1.069 | 2.452 | 0.004 |
| 1000,00 | 2.359 | 4.062 | 0.005 |
| 2000,00 | 4.666 | 7.546 | 0.004 |
| 3000,00 | 5.249 | 11.790 | 0.004 |
| 5000,00 | 9.382 | 17.030 | 0.005 |
| 8000,00 | 14.774 | 19.382 | 0.004 |
| 10000,00 | 14.679 | 20.768 | 0.004 |

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

| Tamaño de la muestra (LINKED\_LIST) | enqueue (Linked List) | dequeue (Linked List) | peek (Linked List) |
| --- | --- | --- | --- |
| 50,00 | 0.155 | 0.212 | 0.012 |
| 500,00 | 1.059 | 1.709 | 0.006 |
| 1000,00 | 2.092 | 3.689 | 0.005 |
| 2000,00 | 4.480 | 7.379 | 0.005 |
| 3000,00 | 7.016 | 11.336 | 0.006 |
| 5000,00 | 9.656 | 15.847 | 0.005 |
| 8000,00 | 16.035 | 24.417 | 0.005 |
| 10000,00 | 18.936 | 29.745 | 0.004 |

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

| Tamaño de la muestra (LINKED\_LIST) | push (Linked List) | pop (Linked List) | top(Linked List) |
| --- | --- | --- | --- |
| 50,00 | 0.122 | 0.633 | 0.014 |
| 500,00 | 0.878 | 26.651 | 0.067 |
| 1000,00 | 1.266 | 61.629 | 0.079 |
| 2000,00 | 2.012 | 237.078 | 0.121 |
| 3000,00 | 2.630 | 599.618 | 0.322 |
| 5000,00 | 5.615 | 1.575.898 | 0.336 |
| 8000,00 | 8.247 | 3.940.290 | 0.533 |
| 10000,00 | 9.915 | 6.190.237 | 0.621 |

# **Máquina 2**

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

| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| --- | --- | --- | --- |
| 0.50% | **0.042** | **0.045** | **0.004** |
| 5.00% | **0.201** | **0.337** | **0.002** |
| 10.00% | **0.440** | **0.713** | **0.002** |
| 20.00% | **0.779** | **1.354** | **0.002** |
| 30.00% | **1.141** | **2.014** | **0.002** |
| 50.00% | **1.930** | **3.597** | **0.003** |
| 80.00% | **3.362** | **4.286** | **0.003** |
| 100.00% | **22.310** | **4.439** | **0.004** |

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

| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Array List) |
| --- | --- | --- | --- |
| 0.50% | **0.026** | **0.054** | **0.002** |
| 5.00% | **0.150** | **0.331** | **0.001** |
| 10.00% | **0.417** | **0.808** | **0.002** |
| 20.00% | **0.759** | **1.411** | **0.001** |
| 30.00% | **1.163** | **2.406** | **0.001** |
| 50.00% | **1.991** | **3.261** | **0.002** |
| 80.00% | **1.934** | **3.827** | **0.001** |
| 100.00% | **2.700** | **5.071** | **0.001** |

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

| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| --- | --- | --- | --- |
| 0.50% | **0.030** | **0.029** | **0.003** |
| 5.00% | **0.213** | **0.345** | **0.002** |
| 10.00% | **0.424** | **0.676** | **0.002** |
| 20.00% | **0.495** | **0.772** | **0.002** |
| 30.00% | **1.267** | **1.895** | **0.003** |
| 50.00% | **2.000** | **3.053** | **0.003** |
| 80.00% | **2.046** | **3.520** | **0.002** |
| 100.00% | **22.158** | **4.385** | **0.003** |

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

| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| --- | --- | --- | --- |
| 0.50% | **0.016** | **0.033** | **0.002** |
| 5.00% | **0.216** | **0.424** | **0.001** |
| 10.00% | **0.400** | **0.834** | **0.002** |
| 20.00% | **0.465** | **0.954** | **0.001** |
| 30.00% | **0.911** | **1.506** | **0.001** |
| 50.00% | **1.751** | **2.798** | **0.001** |
| 80.00% | **2.049** | **3.087** | **0.001** |
| 100.00% | **2.457** | **4.848** | **0.001** |

# **Máquina 3**

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

| Porcentaje de la muestra | enqueue (Array List) | dequeue (Array List) | peek (Array List) |
| --- | --- | --- | --- |
| 0.50% | 0.049 | 0.099 | 0.005 |
| 5.00% | 0.210 | 0.195 | 0.012 |
| 10.00% | 1.050 | 0.980 | 0.006 |
| 20.00% | 0.060 | 0.50 | 0.005 |
| 30.00% | 0.090 | 0.075 | 0.007 |
| 50.00% | 0.150 | 0.125 | 0.011 |
| 80.00% | 0.240 | 0.200 | 0.005 |
| 100.00% | 0.300 | 0.250 | 0.005 |

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

| Porcentaje de la muestra | push (Array List) | pop (Array List) | top(Array List) |
| --- | --- | --- | --- |
| 0.50% | 0.043 | 0.050 | 0.005 |
| 5.00% | 0.291 | 0.260 | 0.005 |
| 10.00% | 0.789 | 0.450 | 0.006 |
| 20.00% | 1.163 | 0.900 | 0.005 |
| 30.00% | 1.200 | 4.550 | 0.006 |
| 50.00% | 2.500 | 4.800 | 0.007 |
| 80.00% | 3.500 | 4.848 | 0.009 |
| 100.00% | 4.000 | 5.000 | 0.009 |

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

| Porcentaje de la muestra | enqueue (Linked List) | dequeue (Linked List) | peek Linked List) |
| --- | --- | --- | --- |
| 0.50% | 0.055 | 0.077 | 0.006 |
| 5.00% | 0.313 | 0.591 | 0.004 |
| 10.00% | 0.933 | 1.553 | 0.005 |
| 20.00% | 1.458 | 2.442 | 0.005 |
| 30.00% | 6.800 | 9.400 | 0.005 |
| 50.00% | 9.500 | 12.765 | 0.006 |
| 80.00% | 11.050 | 15.876 | 0.006 |
| 100.00% | 17.400 | 25.543 | 0.005 |

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

| Porcentaje de la muestra | push (Linked List) | pop (Linked List) | top(Linked List) |
| --- | --- | --- | --- |
| 0.50% | 0.151 | 0.532 | 0.011 |
| 5.00% | 0.765 | 20.239 | 0.043 |
| 10.00% | 0.800 | 55.683 | 0.050 |
| 20.00% | 1.000 | 100.900 | 0.080 |
| 30.00% | 2.700 | 230.094 | 0.190 |
| 50.00% | 4.516 | 999.039 | 0.344 |
| 80.00% | 5.243 | 1.235.349 | 0.598 |
| 100.00% | 10.519 | 5.349.302 | 0.712 |

# **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?  
   Las diferencias significativas se muestran al final, donde siempre el enqueue se vuelve más ineficiente que el push por un margen muy grande. Además alrededor de el porcentaje de 30% o 50% el dequeue y pop empiezan a igualarse y mostrar muy poca diferencia en sus tiempos. Push se muestra ser más eficiente que enqueue, pop y dequeue llevan tiempos muy parecidos y finalmente top es más eficiente que peek pero solo por alrededor de 0.001 o 0.002 ms. Es mas rapida stack al final qué queue por el tipo de listas que son, ya qué push solo requiere O(1) mientras que enqueue es O(n).
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?  
   Es preferible usar ArrayList cuando se necesita un acceso rápido a los elementos mediante índices, ya que proporciona una complejidad O(1) en accesos aleatorios, mientras que LinkedList tiene una complejidad O(n) debido a que necesita recorrer la lista desde el inicio o el final. Sin embargo, si la operación principal es insertar o eliminar elementos con frecuencia, LinkedList es más eficiente, ya que estas operaciones tienen una complejidad O(1) cuando se dispone del nodo correcto, mientras que en ArrayList pueden ser O(n) debido al costo de desplazamiento de todos los elementos.
3. Durante la ejecución de las pruebas ¿Se presentan anomalías en los tiempos de ejecución que no se explican con la teoría?  
   Si se presentan, mi teoría sobre esto es que mientras que el ambiente en el que se hacen los tests no cambia, el computador no siempre está en la misma temperatura o nunca se usa el mismo porcentaje de ram continuamente, lo mismo que con la cpu. Esto afecta los resultados un cierto tamaño.
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 | En una cola, la inserción ocurre típicamente al final. En LinkedList, esta operación es O(1) si se usa un puntero al final, mientras que en ArrayList puede ser O(n) en el peor caso si hay que redimensionar el array. |
| **Dequeue()** |  | x | La eliminación del primer elemento en LinkedList es O(1), mientras que en ArrayList es O(n), ya que requiere desplazar los elementos restantes. |
| **Peek()** | x | x | Ambas estructuras permiten acceder al primer elemento en O(1). |
| STACK | **Push()** | x | x | Ambas estructuras permiten agregar elementos al final en O(1). Sin embargo, ArrayList puede requerir redimensionamiento en algunos casos. |
| **Pop()** | x | x | Ambas permiten eliminar el último elemento en O(1). En ArrayList, se reduce el tamaño lógico, y en LinkedList, se ajusta el puntero del último nodo. |
| **Top()** | x | x | Ambas permiten acceder al último elemento en O(1). |