Leah Benitez

Professor Fakhouri

CMP 338

October 27, 2020

Homework 4 Analysis

Queue Test Cases:

|  |
| --- |
| Running test = Enqueue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  9530.551 3982.529 2941.654 7701.592 3849.035 2396.732 2238.481 1696.148 1612.766 1819.397 3776.8885  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  ArrayBasedQueue  Running test = Enqueue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  1638.6328125 2293.9921875 2949.3515625 3604.7109375 4299.171875 4954.578125 5609.9765625 6265.3828125 6920.765625 7576.171875 4611.2734375  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  ArrayBasedQueue |
| Running test = Dequeue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  3327.694 2973.324 2957.965 2745.101 3765.174 2612.854 2714.743 7992.986 7313.732 1999.008 3840.2581000000005  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  ArrayBasedQueue  Running test = Dequeue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  8559.3671875 9214.7734375 9870.1796875 10525.578125 11180.984375 11836.3828125 12491.7890625 13147.1875 13802.59375 14457.9921875 11508.6828125  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  ArrayBasedQueue |
| Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  22116.66 21653.368 11086.992 22128.503 18628.165 24451.22 16420.663 12275.137 15825.402 9040.119 17362.6229  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  ArrayBasedQueue  Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  15113.3984375 15768.8046875 16384.0 1183.0390625 1870.9375 2597.9140625 3285.8125 3973.71875 4700.65625 5388.5625 7026.684375  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  ArrayBasedQueue |
| Running test = Enqueue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  3900.275 1441.895 1437.099 2202.521 1340.069 601.6 606.504 599.24 600.986 619.926 1335.0115  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  LinkedQueue  Running test = Enqueue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  6525.671875 7498.890625 8147.6953125 9120.890625 10094.109375 10742.9140625 11716.1328125 12689.375 13338.203125 14311.421875 10418.53046875  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  LinkedQueue |
| Running test = Dequeue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  3037.219 1664.147 1513.856 3369.636 1843.283 5210.028 9363.12 3733.657 1311.931 837.622 3188.4499000000005  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  LinkedQueue  Running test = Dequeue  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  15284.65625 15933.4609375 16879.109375 1892.921875 2541.734375 3514.953125 4488.171875 5136.984375 6110.203125 7083.421875 7886.56171875  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  LinkedQueue |
| Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  10415.357 8968.207 7626.77 7661.701 9974.062 13209.041 11479.919 8860.816 10499.685 12882.805 10157.836299999999  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  LinkedQueue  Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  7732.234375 8705.453125 9678.671875 10690.9453125 11339.734375 12312.953125 13286.171875 13974.0625 14947.28125 15920.5 11858.80078125  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  LinkedQueue |

Stack Test Cases:

|  |
| --- |
| Running test = Push  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  31662.424 24389.015 11619.732 17590.567 12204.254 10963.474 9631.594 15777.243 21963.36 7575.997 16337.766  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  ArrayBasedStack  Running test = Push  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  3651.59375 7556.1328125 11457.796875 31801.453125 19225.4296875 8377.3125 28727.15625 49077.0 3821.90625 23074.8203125 18677.06015625  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  ArrayBasedStack |
| Running test = Pop  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  9636.42 11888.875 11339.76 9918.523 11236.758 23272.839 25420.441 13879.795 9978.585 8967.008 13553.900400000002  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  ArrayBasedStack  Running test = Pop  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  43526.8671875 63774.703125 19652.2109375 38625.40625 59694.2734375 80771.234375 99746.8125 15551.140625 36647.234375 55708.1484375 51369.803125  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  ArrayBasedStack |
| Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  18202.457 15303.055 16031.992 15353.0 15181.636 14566.465 14986.764 27477.528 32169.046 30820.99 20009.293299999998  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  ArrayBasedStack  Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  76793.6953125 97894.28125 14328.203125 34507.3828125 54685.5078125 74862.421875 95003.875 115178.171875 135351.296875 155523.1328125 85412.796875  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  ArrayBasedStack |
| Running test = Push  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  3328.849 1386.698 2103.44 1360.43 1036.718 1061.638 1044.724 1049.365 1047.168 1252.842 1467.1872  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  LinkedStack  Running test = Push  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  156614.7578125 159950.4453125 159950.4453125 159950.4453125 159950.4453125 163286.1328125 163286.1328125 163286.1328125 163286.1328125 166621.8203125 161618.2890625  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  LinkedStack |
| Running test = Pop  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  4109.131 1383.317 1298.214 1337.651 2150.596 2111.628 3413.073 1580.48 1385.147 1585.11 2035.4347000000002  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  LinkedStack  Running test = Pop  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  166621.8203125 166621.8203125 166621.8203125 168790.59375 168790.59375 168790.59375 3944.5703125 3944.5703125 3944.5703125 3944.5703125 102201.55234375  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  LinkedStack |
| Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  16159.957 9374.197 12813.255 9035.196 14139.076 12952.744 11910.395 11165.551 10848.464 9781.239 11818.007400000002  MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds MicroSeconds  LinkedStack  Running test = Iterate  Run 1 Run 2 Run 3 Run 4 Run 5 Run 6 Run 7 Run 8 Run 9 Run 10 Average  7277.0234375 7277.0234375 7277.0234375 10609.453125 10609.453125 10609.453125 10609.453125 13941.8828125 13941.8828125 13941.8828125 10609.453125  KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes KiloBytes  LinkedStack |

1. Create graphs from the test times you measured.

Queue Test Cases:

Stack Test Cases:

Averages:

1. **Discuss the difference in test times and memory usage between the array based queue and the linked queue for all the test cases. Did the results match your expectations? Why or why not? Be as specific as possible.**

Except for maybe the deque test case (the results jumped all over the place in that graph), the linked queue was faster. The linked queue also took up more memory in the enqueue and iterate test cases.

I expected the linked queue to be faster in all of the cases, so the results did match what I thought would happen. I figured the strength in arrays is being able to access indices easily (especially middle ones). But with queues, you’re only really working with the front and back indices, so that strength is not really highlighted the same. In regards to the linked queues, I think the nodes make it easier to access and manipulate the front and back of the queue, so that is why I thought it would be faster.

In terms of memory, I did not really go into it with any expectations as I am unfamiliar with how memory works. But since I thought the linked queue would be faster, I am surprised it takes up more memory. In my head, faster speed = less memory, but I am sure that is oversimplified thinking.

1. **Are the test times ever similar for both the array based queue and the linked queue? Why or why not? Be as specific as possible.**

The test times were pretty similar for the array-based queue and the linked queue for the deque test case. I believe this is because the concepts are pretty similar. You are not adding anything new, you are just changing the front reference.

1. **In which test cases is the array based queue faster than the linked queue? Please explain the reason. Be as specific as possible.**

The array-based queue was not really faster than the linked queue. I believe the reason is the same as the one I mentioned earlier. I believe implementing a queue as a linked structure is advantageous because nodes make it easier to access and manipulate the front and back of the queue.

1. **Discuss the difference in test times and memory usage between the array based stack and the linked stack for all the test cases. Did the results match your expectations? Why or why not? Be as specific as possible.**

More clearly than with the queues, the array-based stack took longer than the linked stack

for all test cases. The linked stack took up more memory than the array-based stack in the push and pop test cases but way less memory in the iterate test case.

Again, I expected the linked stack to be faster than the array-based stack. And it is for the same reason, too. Since you are only really working with the top of the stack, the use of nodes makes that easier. I was surprised by the memory results because it just suddenly dipped for the linked stack. What caused that?

1. **Are the test times ever similar for both the array based stack and the linked stack? Why or why not? Be as specific as possible.**

The test times were very different for the array-based stack and the linked stack in all of the test cases. For the push test case specifically, I think a big weakness for the array-based stack is the fact that there is a restriction to the capacity of the array and it may need to be resized when you push a new element. Linked stacks do not have that same restriction. But in general, I believe the linked stack is faster because of its use of nodes.

1. **In which test cases is the array based stack faster than the linked stack? Please explain the reason. Be as specific as possible.**

There are no cases in which the array-based stack is faster than the linked stack. I am sorry to be repetitive but I really do believe the use of nodes is what made the linked stack faster.