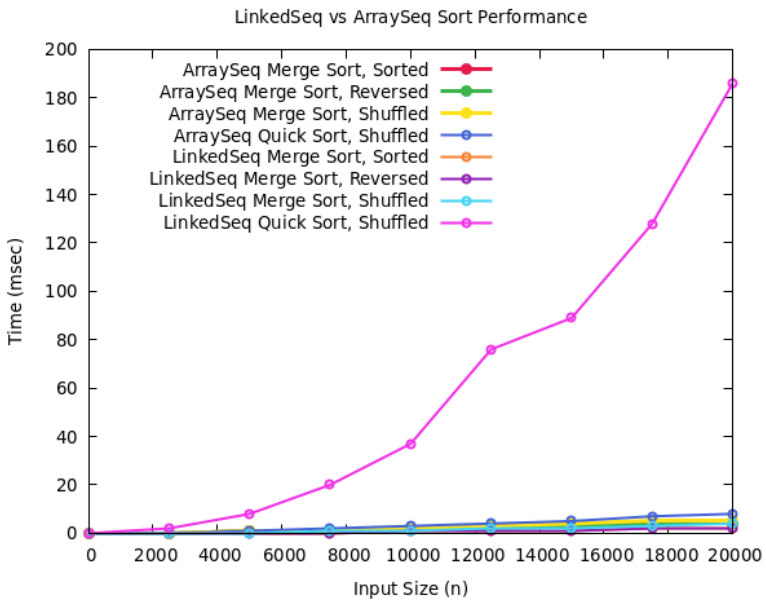
NAME: Ben Puryear

FILE: HW-4_WriteUp.pdf

DATE: Fall 2021

DESC: This pdf goes over the basics accomplished throughout all the HW-4 related files.

Graphs / Explanations



This graph shows the performance of merge_sort() and quick_sort() in the datatypes of array sequences as well as linked sequences. One thing that sticks out is just how much worse quick_sort() is for a shuffled linked sequence. This instance is over 100x worse than all of the other combinations. This makes sense because of how quick_sort() works, accessing indexes a lot, which means that it needs to traverse the linked sequence exponentially more than merge_sort().

Issues I faced:

The main issue I faced was figuring out quick sort for a linked sequence.

Explanation of Tests:

```
TEST(CustomTests, FourElementMergeArraySeq)
```

```
{
    ArraySeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.merge_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);
}
```

This test checks to see if a 4 element arrayseq can be sorted with merge_sort().

```
TEST(CustomTests, FourElementQuickeArraySeq)
{
    ArraySeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.quick_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);
}
```

This test checks to see if a 4 element arrayseq can be sorted with quick_sort().

```
TEST(CustomTests, FourElementMergeLinkedSeq)
{
    LinkedSeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.merge_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);
```

```
}
```

This test checks to see if a 4 element linkedseq can be sorted with merge_sort ().

```
TEST(CustomTests, FourElementQuickLinkedSeq)
{
    LinkedSeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.quick_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);
}
```

This test checks to see if a 4 element linkedseq can be sorted with quick_sort ().

```
TEST(CustomTests2, FourElementMergeArraySeq)
{
    ArraySeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.merge_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);

    seq.insert(0, 2);

    seq.merge_sort();
    ASSERT_EQ(0, seq[0]);
}
```

This is a test that checks to see if multiple merge_sort()’s can be preformed multiple times on an arrayseq.

```
TEST(CustomTests2, FourElementQuickeArraySeq)
{
    ArraySeq<int> seq;
```

```
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.quick_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);

    seq.insert(0, 2);

    seq.quick_sort();
    ASSERT_EQ(0, seq[0]);
}
```

This is a test that checks to see if multiple quick_sort()'s can be preformed multiple times on an arrayseq.

```
TEST(CustomTests2, FourElementMergeLinkedSeq)
{
    LinkedSeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.merge_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);

    seq.insert(0, 2);

    seq.merge_sort();
    ASSERT_EQ(0, seq[0]);
}
```

This is a test that checks to see if multiple merge_sort()'s can be preformed multiple times on an linkedseq.

```
TEST(CustomTests2, FourElementQuickLinkedSeq)
{
    LinkedSeq<int> seq;
    seq.insert(10, 0);
    seq.insert(20, 0);
```

```
    seq.insert(30, 0);
    seq.insert(90, 0);

    seq.quick_sort();
    ASSERT_EQ(10, seq[0]);
    ASSERT_EQ(20, seq[1]);
    ASSERT_EQ(30, seq[2]);
    ASSERT_EQ(90, seq[3]);

    seq.insert(0, 2);

    seq.quick_sort();
    ASSERT_EQ(0, seq[0]);
}
```

This is a test that checks to see if multiple quick_sort()’s can be preformed multiple times on an linkedseq.