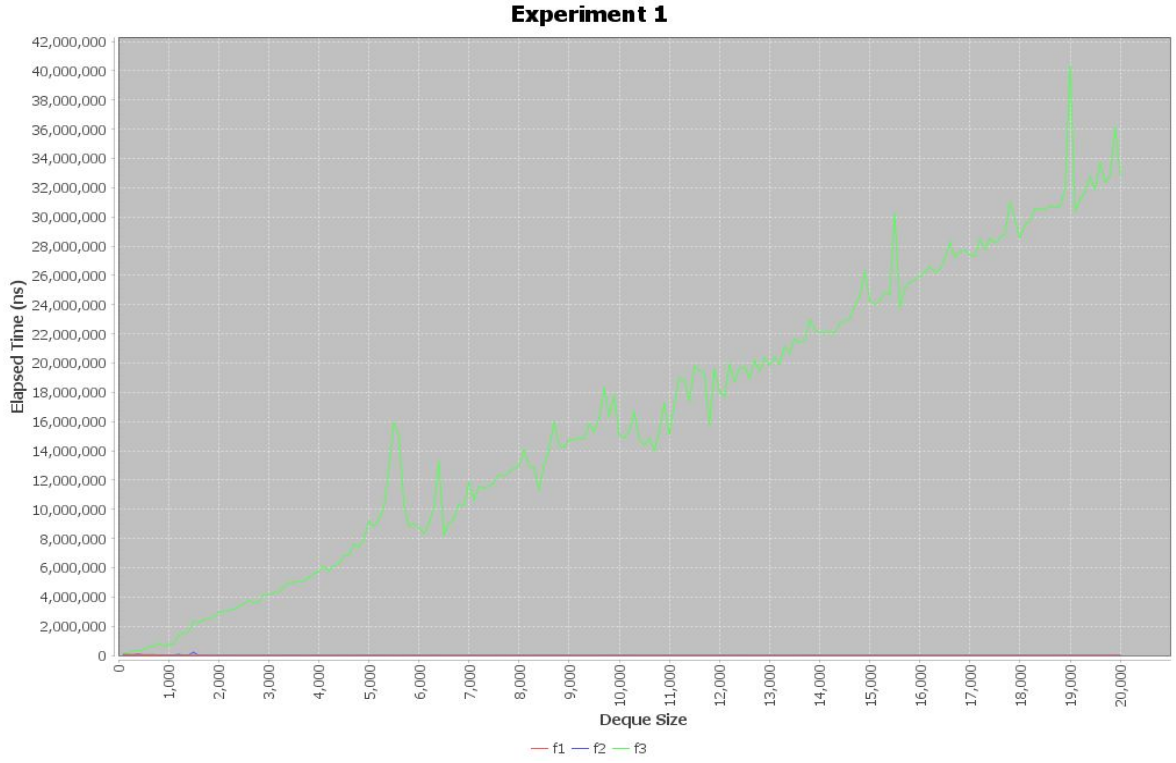
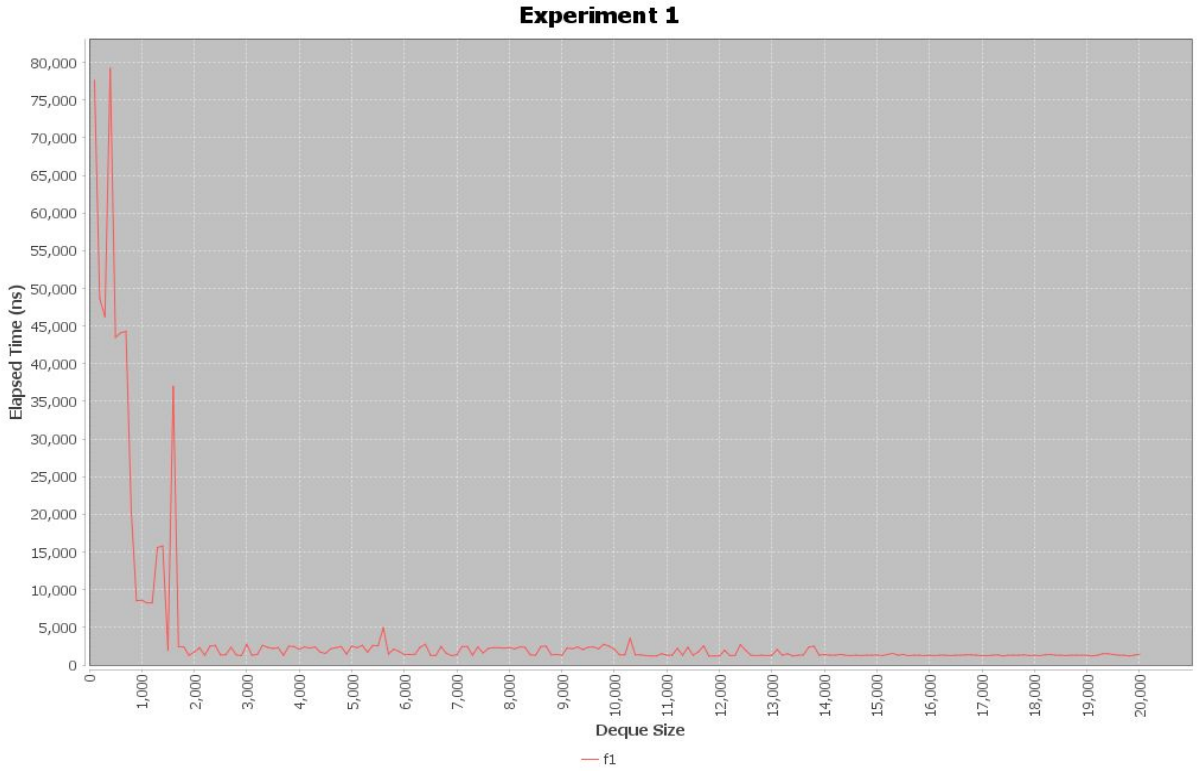
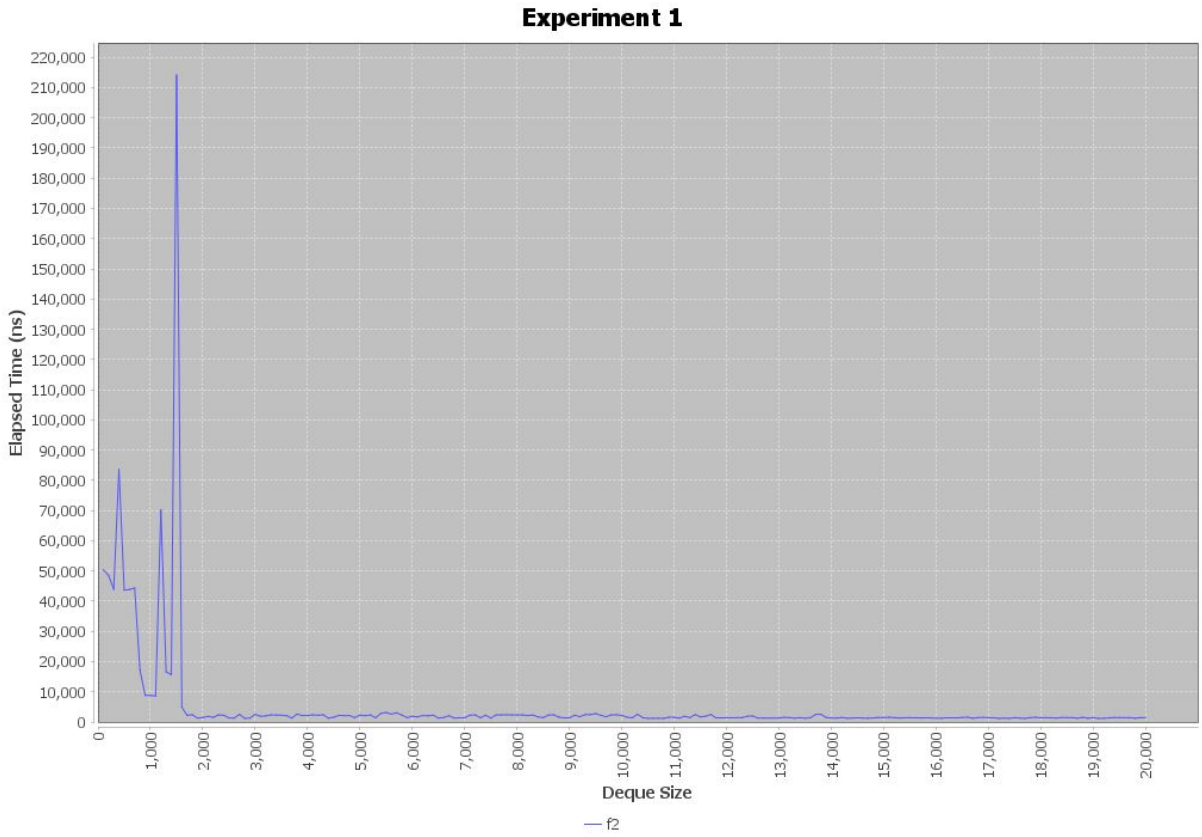
Project 1 Experiment

Experiment 1:

- **Prediction:**
  - f1: timeGetsAtIndex _0: constant
    // can use front to get directly, no need to enter for loop to search the rest.
  - f2: timeGetsAtIndex_size - 1: linear
    // since our code always searches from the front, it enters the while loop to go through N (size) times to reach this index.
  - f3: timeGetsAtIndex_size / 2: linear
    // // since our code always searches from the front, it enters the while loop to go through N (size) / 2 times to reach this index. It should take about half the time as it takes for f2.
- **Discussion:**
  - f1: This is the best case since the get method can access this index using "front" directly.
  - f2: This is the best case since the get method can quickly access to this index using "back" directly.
  - f3: This is the worst case since no matter using front or back, the get method has to go through the while loop that takes N(size)/2 times to reach the middle of the list to retrieve this value.
- **Result plot:**




Experiment 1
Elapsed Time (ns)
80,000
75,000
70,000
65,000
60,000
55,000
50,000
45,000
40,000
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
0
1,000
2,000
3,000
4,000
5,000
6,000
7,000
8,000
9,000
10,000
11,000
12,000
13,000
14,000
15,000
16,000
17,000
18,000
19,000
20,000
Deque Size
f1


Experiment 1
Elapsed Time (ns)
220,000
210,000
200,000
190,000
180,000
170,000
160,000
150,000
140,000
130,000
120,000
110,000
100,000
90,000
80,000
70,000
60,000
50,000
40,000
30,000
20,000
10,000
0
0
1,000
2,000
3,000
4,000
5,000
6,000
7,000
8,000
9,000
10,000
11,000
12,000
13,000
14,000
15,000
16,000
17,000
18,000
19,000
20,000
Deque Size
f2

Experiment 2:

- **Prediction:**
    - linkedDeque: linear
    - arrayDeque: linear (if size is increasing continuously or it is connected on the plot)
- **Discussion:**
    - It is because array takes up its memory for a certain size (so there might be some null elements, but they take up memory) when compiling that it results in spike-shapes in the plot. On the other hand, since linkedDeque only increases its element and takes up the memory during execution, its memory usage increases linearly.
- **Resulted Plot:**