1. **Your names, CSUF-supplied email address(es), and an indication that the submission is for project 2.**

Report for Project 2

Brianna Sharpe briannasharpe@csu.fullerton.edu

2. **Two scatter plots and other requirements**
    1. **Pseudocode and efficiency**

```
longest_nonincreasing_end_to_beginning(sequence A):
    n = size of sequence A -1 t.u
    initialize vector H -1 t.u
    for i = n-2 to 0: -n times
        for j = i+1 to n: -n times
            if A[i] >= A[j] && H[i] <= H[j]+1 -2 t.u
                H[i] = H[j]+1; -1 t.u
            endif
        endfor
    endfor

    calculate max -1 t.u
    initialize vector R -1 t.u
    initialize index and j -2 t.u
    for i to n -n times
        if H[i] == index -1 t.u
            R[j] = A[i] -1 t.u
            index-- -1 t.u
            j++ -1  t.u
        endif
    endfor

    return R -1 t.u
```

Efficiency: O(n^2)

Total steps: 14 + 3n steps

```
longest_nonincreasing_powerset(sequence A):
        n = size of sequence A -1 t.u
        initialize sequence best -1 t.u
        initialize vector stack -1 t.u
        k = 0 -1 t.u
        while(true) -2^n
                if (stack[k] < n) -1 t.u
                        stack[k+1] = stack[k] + 1 -1 t.u
                        ++k -1 t.u
                else
                        stack[k-1]++ -1 t.u
                        k-- -1 t.u
                endif

                if (k==0) -1 t.u
                        break -1 t.u
                endif

                initialize sequence candidate -1 t.u
                for i = 1 to k -k times
                        push A[stack[i]-1 to  candidate -1 t.u
                endfor

                if (is_nonincreasing(candidate) && candidate.size() >
                best.size()) -2 t.u
                        candidate = best -1 t.u
                endif
        endwhile

        return best -1 t.u
```

Efficiency: $O(2^n)$

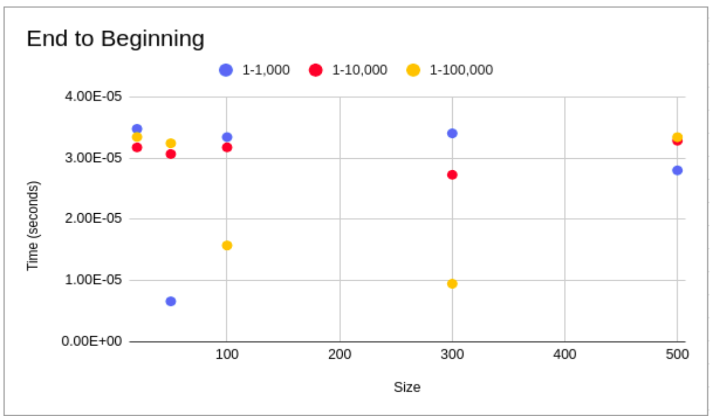
Total steps: $17 + 2^n + k$ steps

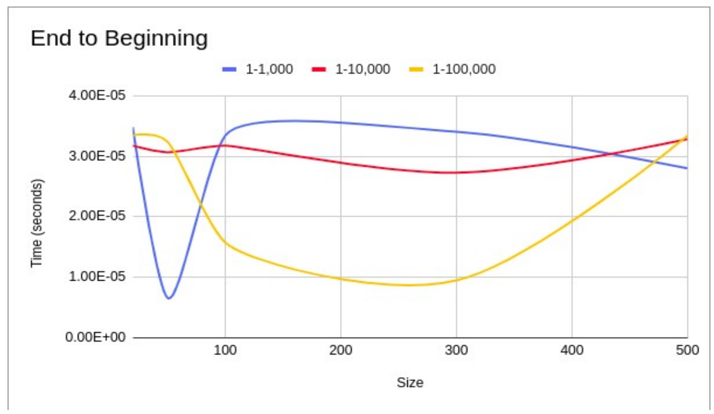
2. **Gather empirical timing data by running your implementation for various values of n.**

End to Beginning

	1-1,000	1-10,000	1-100,000
20	3.48E-05	3.17E-05	3.34E-05
50	6.61E-06	3.07E-05	3.24E-05
100	3.34E-05	3.17E-05	1.57E-05
300	3.40E-05	2.73E-05	9.48E-06
500	2.80E-05	3.28E-05	3.34E-05

3. **Draw a scatter plot and fit line for your timing data.**





4. **Conclude whether or not your empirically-observed time efficiency data is consistent, or inconsistent, with your mathematically-derived big-O efficiency class.**

   My empirically-observed time efficiency data is inconsistent with my mathematically-derived big-O efficiency class.

3. **Answers to the following questions, using complete sentences.**
   1. **Provide pseudocode for your two algorithms.**

```
longest_nonincreasing_end_to_beginning(sequence A):
        n = size of sequence A
        initialize vector H
        for i = n-2 to 0:
                for j = i+1 to n:
                        if A[i] >= A[j] && H[i] <= H[j]+1
                                H[i] = H[j]+1;
                        endif
                endfor
        endfor

        calculate max
        initialize vector R
        initialize index and j
        for i to n
                if H[i] == index
                        R[j] = A[i]
                        index--
                        j++
                endif
        endfor

        return R
```

```
longest_nonincreasing_powerset(sequence A):
        n = size of sequence A
        initialize sequence best
        initialize vector stack
        k = 0
        while(true)
                if (stack[k] < n)
                        stack[k+1] = stack[k] + 1
                        ++k
                else
                        stack[k-1]++
                        k--
                endif

                if (k==0)
                        break
                endif

                initialize sequence candidate
                for i = 1 to k
                        push A[stack[i]-1 to  candidate
                endfor

                if (is_nonincreasing(candidate) && candidate.size() >
                best.size())
                        candidate = best
                endif
        endwhile

        return best
```

2. **What is the efficiency class of each of your algorithms, according to your own mathematical analysis?**

   The efficiency class for the end_to_beginning algorithm is $O(n^2)$.

   The efficiency class for the exhaustive algorithm is $O(2^n)$.

3. **Is there a noticeable difference in the running speed of the algorithms? Which is faster, and by how much? Does this surprise you?**

   There is no noticeable difference in the running speed of the algorithms.

4. **Are the fit lines on your scatter plots consistent with these efficiency classes? Justify your answer.**

   I don't think the fit lines are consistent with the efficiency classes because the data points themselves do not show any specific kind of trend, no matter how small or large the container size is.

5. **Is this evidence consistent or inconsistent with the hypothesis stated on the first page? Justify your answer.**

   The evidence is inconsistent because there is no noticeable trend among the different container sizes and sequence seeds. The points seem to float up and down in no explainable way as there is an increase in the container size and sequence seeds.