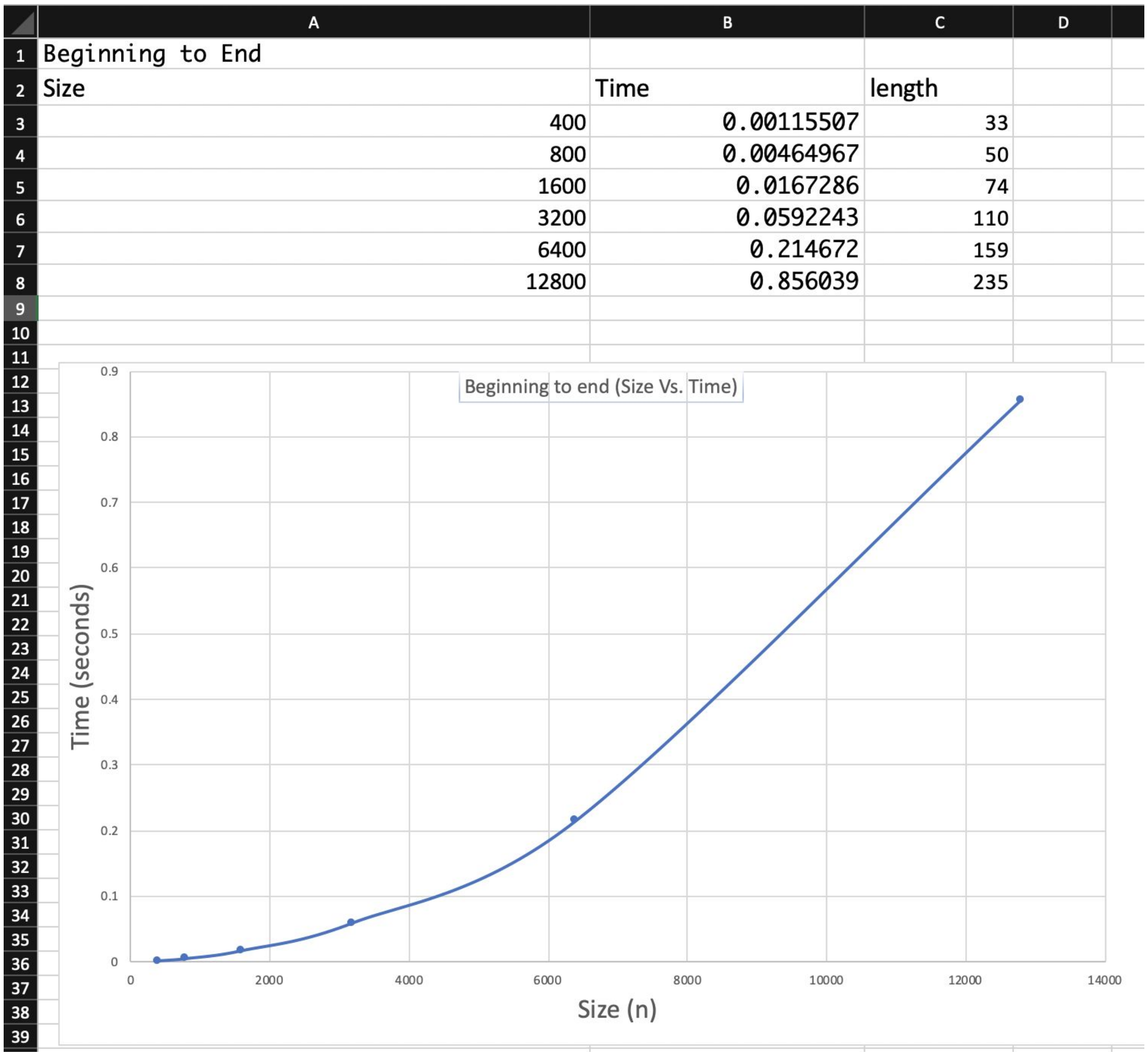
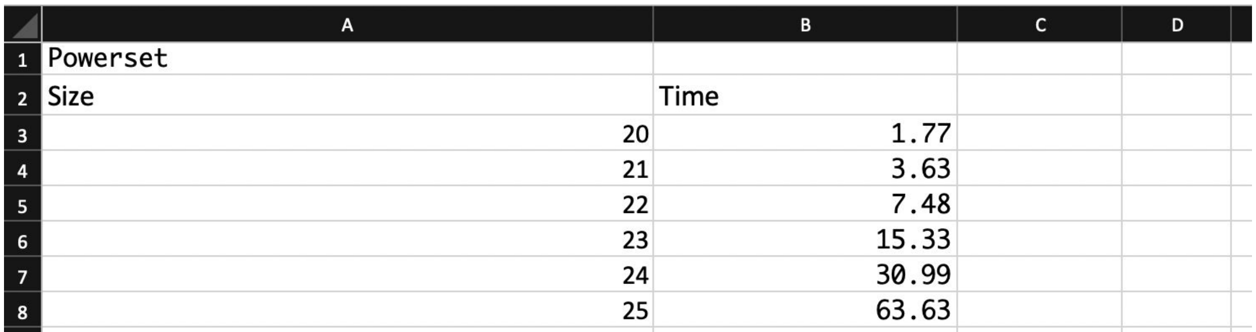
Project 2

Justin Drouin jdrouin@csu.fullerton.edu
Alexander Frederick afrederick2@csu.fullerton.edu

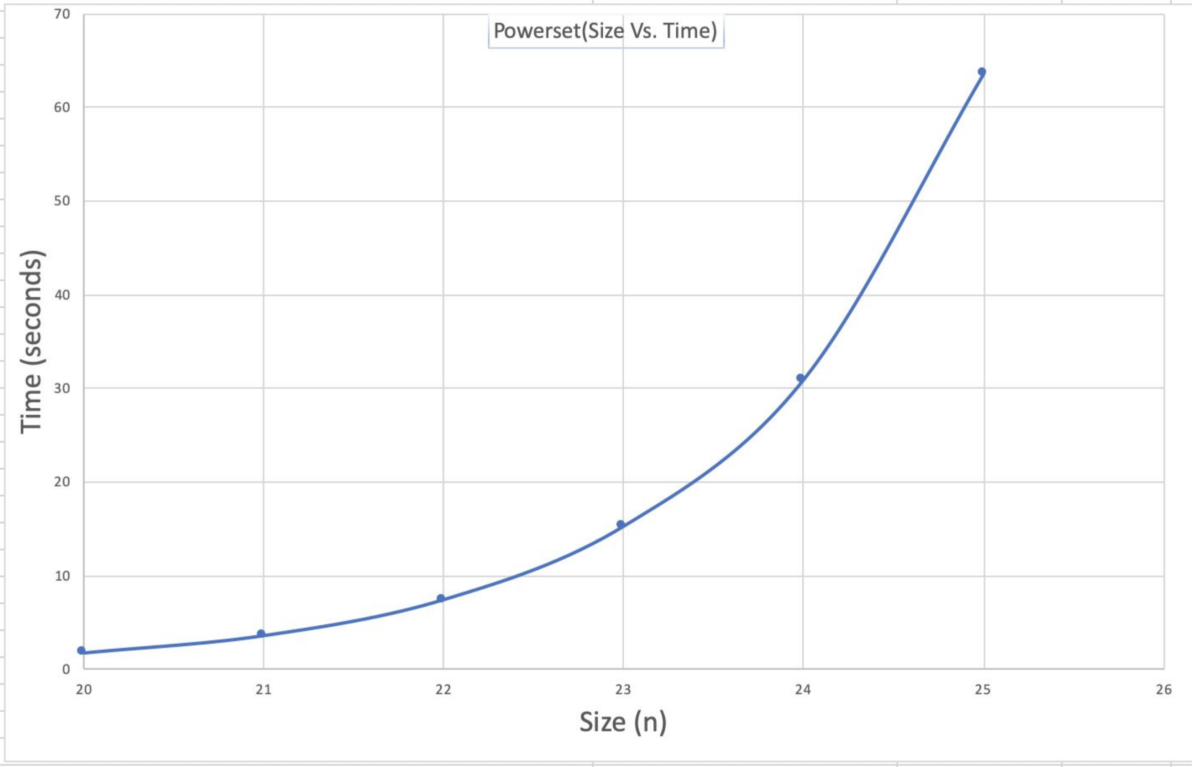
Scatter plots:



Beginning to End		
Size	Time	length
400	0.00115507	33
800	0.00464967	50
1600	0.0167286	74
3200	0.0592243	110
6400	0.214672	159
12800	0.856039	235



	A	B	C	D
1	Powerset			
2	Size	Time		
3	20	1.77		
4	21	3.63		
5	22	7.48		
6	23	15.33		
7	24	30.99		
8	25	63.63		



Pseudocode:

```
sequence longest_nonincreasing_end_to_beginning(const sequence& A) {
  const size_t n = A.size();                //1TU
  std::vector<size_t> H(n, 0);              //1TU

  for (signed int i = n-2;  i >= 0; i--) {   // n-2+1
    for (size_t j = i+1; j < n ; j++) {      // n
      if(A[j] <= A[i] && H[j] >= H[i]){     //7TU +max(4,0) = 11TU
```

```
        H[i] = 1 + H[j];                        //+4TU
      }
    }
  }
  auto max = *std::max_element(H.begin(), H.end()) + 1; //1TU+1TU+N?

  std::vector<int> R(max);                //1TU

  size_t index = max-1, j = 0;            //3TU

  for (size_t i = 0; i < n; ++i) {        //n-1+1
    if (H[i] == index) {                  //2+max(7,0)
      R[j] = A[i];                        //3TU
      index--;                            //2TU
      j++;                                //2TU
    }
  }
  return sequence(R.begin(), R.begin() + max); //1TU
}
```

2 TU

11(n-1)(n) = (11n-11)(n) = 2 + 11n^2 – 11n

2 + 11n^2 – 11n +2 + n + 1 + 3 = 2 + 11n^2 – 11n + n + 6 = 11n^2 – 11n + n + 8

11n^2 – 10n + 8

2+max(7,0) = 9TU = 9(n-1+1) = 9n

11n^2 – 10n + 8 + 9n + 1 = 11n^2 – 10n + 9n + 8 + 1 = 11n^2 – n + 9

Assuming O(n^2) Efficiency

lim (11n^2 – n + 9) / n^2 = lim(11) – lim(1/n) + lim(9/n^2)

= 11 – 0 + 0 = <u>**11**</u>

11 is non-negative and constant with respect to n^2. <u>Therefore 11n^2 – n + 9 ∈ O(n^2)</u>

```
 sequence longest_nonincreasing_powerset(const sequence& A) {
  const size_t n = A.size();              //2TU
  sequence best;                          //1TU
  std::vector<size_t> stack(n+1, 0);
  size_t k = 0;                           //2TU
  while (true) {                          //Due to k break value the exit condition
```

```
                                            //is 2^n as n increases k must do 2 steps

    if (stack[k] < n) {                     //1+max(7,0)TU = 8TU
      stack[k+1] = stack[k] + 1;            //4TU
      ++k;                                  //3TU
    } else {
      stack[k-1]++;                         //4TU
      k--;                                  //3TU
    }

    if (k == 0) {                           //1+max(1,0) = 2TU
      break;                                //1TU
    }

    sequence candidate;                     //1TU
    for (size_t i = 1; i <= k; ++i) {       //nTU
          candidate.push_back(A[stack[i]-1]);   //4TU
    }

    if(is_nonincreasing(candidate) && (candidate.size() > best.size())) {
                                            //4+max(1,0) = 5TU
          best = candidate;           //1TU
    }
```

$5+2^n(8+2+1+n+5) = 2^n(16+n) + 5 = 2^n*16 + 2^n*n + 5$

Assuming O(2^n*n) Efficiency

$\lim (2^n*16 + 2^n*n + 5) / n * 2^n = \lim(16/n) + \lim(1) + \lim(5/2^n*n)$

$= 0 + 1 + 0 = \underline{\mathbf{1}}$

1 is non-negative and constant with respect to 2^n*n. **Therefore $\underline{2^n*16 + 2^n*n + 5 \in O(2^n*n)}$**

a. Report document presentation = 3 points
b. Pseudocode = 3 points
c. Scatter plots = 3 points
d. Empirical analysis = 8 points (4 points for End-to-Beginning and 4 points for Exhaustive)
e. Question answers = 3 points

1. Two scatter plots meeting the requirements stated above.
2. Answers to the following questions, using complete sentences.
    a. Provide pseudocode for your two algorithms.

b. What is the efficiency class of each of your algorithms, according to your own mathematical analysis? (You are not required to include all your math work, just state the classes you derived and proved.)

The efficiency of the end to beginning algorithm is $O(n^2)$.
The efficiency of the powerset algorithm is $O(2^n * n)$.

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

The end to beginning is much faster than the power set algorithm. Since the difference between $O(2^n * n)$ grows much faster than $O(n^2)$ it is not really surprising that it runs faster, but what is surprising is how quickly the powerset algorithm becomes useless, from 20 to 25 values it comes to almost a stand still.

d. Are the fit lines on your scatter plots consistent with these efficiency classes? Justify your answer.
The powerset algorithm grows drastically as it increases, from a range of 20 million output possibilities at 20 input values, to 838 million output possibilities at 25 input.
Meanwhile the end to beginning algorithm grows quickly, but not at the rate of powerset. At 2000 input values there are 4 million possible output values, and at 12,800 input values there are 163 million possible output values.

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

Hypothesis 1: Exhaustive search algorithms are feasible to implement, and produce correct outputs.
They are feasible and they do produce correct output. Searching exhaustively ensures that the output values have considered every possible element of the input. Implementation is not bad as your end state is still the same as the worst case state of searching the data and not finding a verified candidate.

Hypothesis 2: Algorithms with exponential or factorial running times are extremely slow, probably too slow to be of practical use.
This hypothesis was proven consistent. As the input values pass 25 elements, the algorithm slows to a crawl as there are too many possibilities to test.