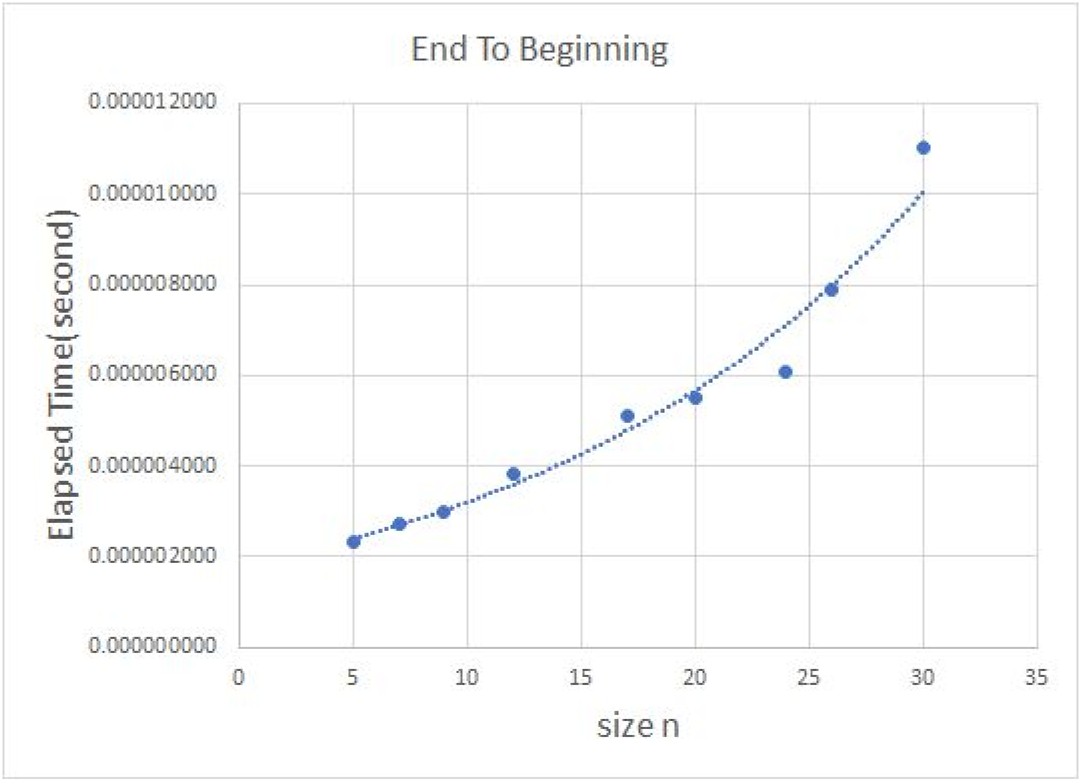
CPSC 335 - Project 2 - Report

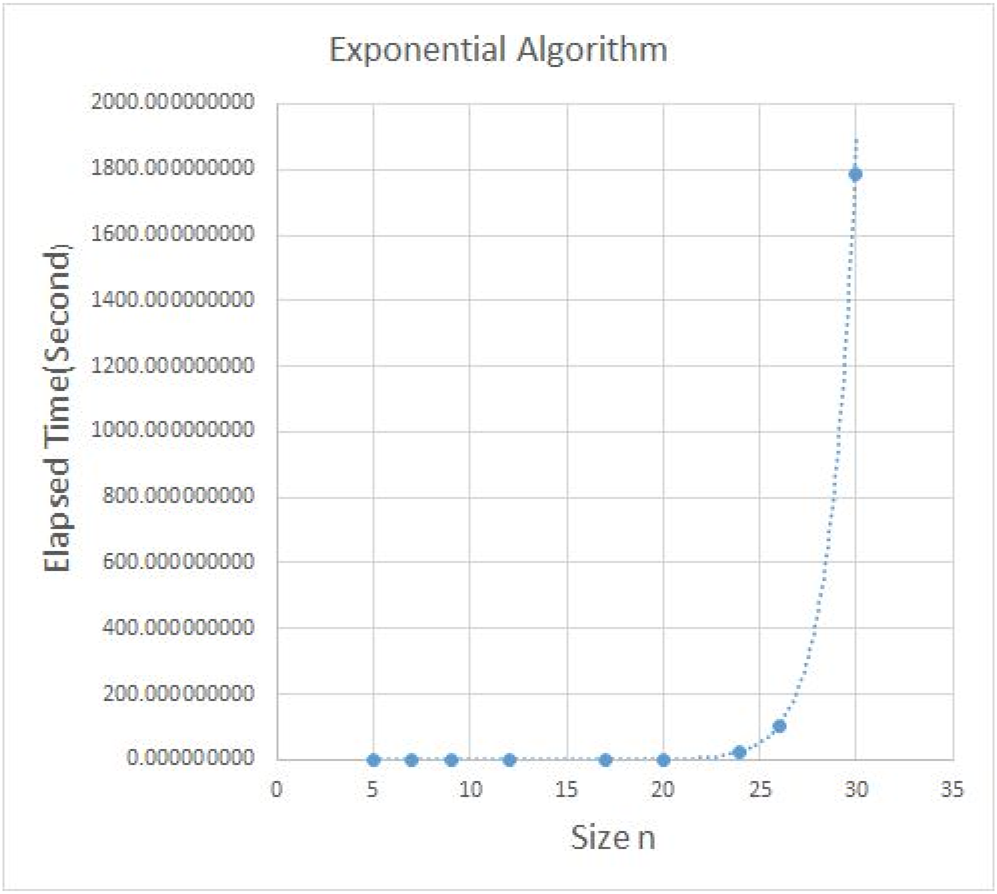
1. Your names, CSUF-supplied email address(es), and an indication that the submission is for project 2.

   Members
   Henry Ho vho23@csu.fullerton.edu
   Ji Xiaomei xiaomeiji@csu.fullerton.edu
   CPSC 335 - Project 2 - Report

2. Two scatter plots meeting the requirements .Gather empirical timing data by running your implementation for various values of n. As discussed in class, you need enough data points to establish the shape of the best-fit curve (at least 5 data points, maybe more), and you should use n sizes that are large enough to produce large time values (ideally multiple seconds or even minutes) that minimize instrumental error.

	Elapsed Time(second)	
Value of n	End to beginning	Powerset
5	2.343e-06	2.8505e-05
7	2.712e-06	0.000111653
9	3.008e-06	0.000532307
12	3.847e-06	0.0054774
17	5.129e-06	0.175092
20	5.53e-06	1.49388
24	6.078e-06	25.3245
26	7.912e-06	104.289
30	11.027e-06	1784.13


End To Beginning
0.000012000
0.000010000
0.000008000
0.000006000
0.000004000
0.000002000
0.000000000
Elapsed Time(second)
0
5
10
15
20
25
30
35
size n


Exponential Algorithm
2000.000000000
1800.000000000
1600.000000000
1400.000000000
1200.000000000
1000.000000000
800.000000000
600.000000000
400.000000000
200.000000000
0.000000000
Elapsed Time(Second)
0
5
10
15
20
25
30
35
Size n

3. 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.)

```
sequence longest_increasing_end_to_beginning(const sequence& A)
{

  const size_t n = A.size();
  std::vector<size_t> H(n, 0);

  for (signed int i = n-2;  i>= 0; i--) {
    for (size_t j = i+1; j < n ; j++) {
       if( (A[j]>A[i]) && (H[j])>=H[i]){
           H[i] = H[j] + 1;
     }
    }
  }

  auto max = *std::max_element(H.begin(), H.end()) + 1;
  std::vector<int> R(max);

    size_t index = max-1, j = 0;
    for (size_t i = 0; i < n; ++i) {
      if (H[i] == index) {
           R[j] = A[i];
           index--;
           j++;
      }

    }

  return sequence(R.begin(), R.begin() + max);
}
```

$T(n) = 2n^2 + 8n + 1 \in O(\text{Max}(2n^2, 8n + 1)) \in O(2n^2) \in O(n^2)$
(Dominated Term)

```
sequence longest_increasing_powerset(const sequence& A) {
  const size_t n = A.size();
  sequence best;
  std::vector<size_t> stack(n+1, 0);
  size_t k = 0;
  while (true) {

        if (stack[k] < n) {
              stack[k + 1] = stack[k] + 1;
              ++k;
        }
        else {
              stack[k - 1]++;
              k--;
        }

        if (k == 0) {
              break;
        }

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

        if ((is_increasing(candidate)) && (candidate.size() >
best.size())) {
              for (int i = 0; i < candidate.size(); i++) {
                   if (i >= best.size()) {
                         best.push_back(candidate[i]);
                   }
                   best[i] = candidate[i];
              }

        }
  }

  return best;
}
```

$T(n) = n.2^n + n + 3 \in O(\text{Max}(n.2^n, n + 3)) \in O(n.2^n)$

(Dominated Term)

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

   It is noticeable difference in the running speed of the algorithm, the end to beginning algorithm run a lot faster compare to the other algorithm. About how much faster, it is depends on the input size, the end to beginning time complexity growth quadratically, the other algorithm time complexity growth exponentially which is extremely slow with large input size, we can refer the data tables above for more details. It does surprise me, there only few lines of code and different algorithm can make a program run a lot of faster.

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

   The fit lines on our scatter plots are consistent with these efficiency class. Looking at the part of the graph with input size from n =26 to n =30, the end to beginning graph, the vertical distance between the dot at n = 6 and the dot at n = 30 is much smaller compare to the distance between the two dots for the other algorithm. That we can clearly see that it is consistent with the efficiency classes.

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

   We collected data with more than 5 different input size of n, and we drew a scatter plots to represent the running time of the 2 efficiency classes. We concluded that the exponential algorithm is slower that the end to beginning based on the graph and these data. The hypothesis saying that:

   1. Exhaustive search algorithms are feasible to implement, and produce correct outputs.
   2. Algorithms with exponential or factorial running times are extremely slow, probably too slow to be of practical use.

   → our evidence is consistent with the hypothesis.