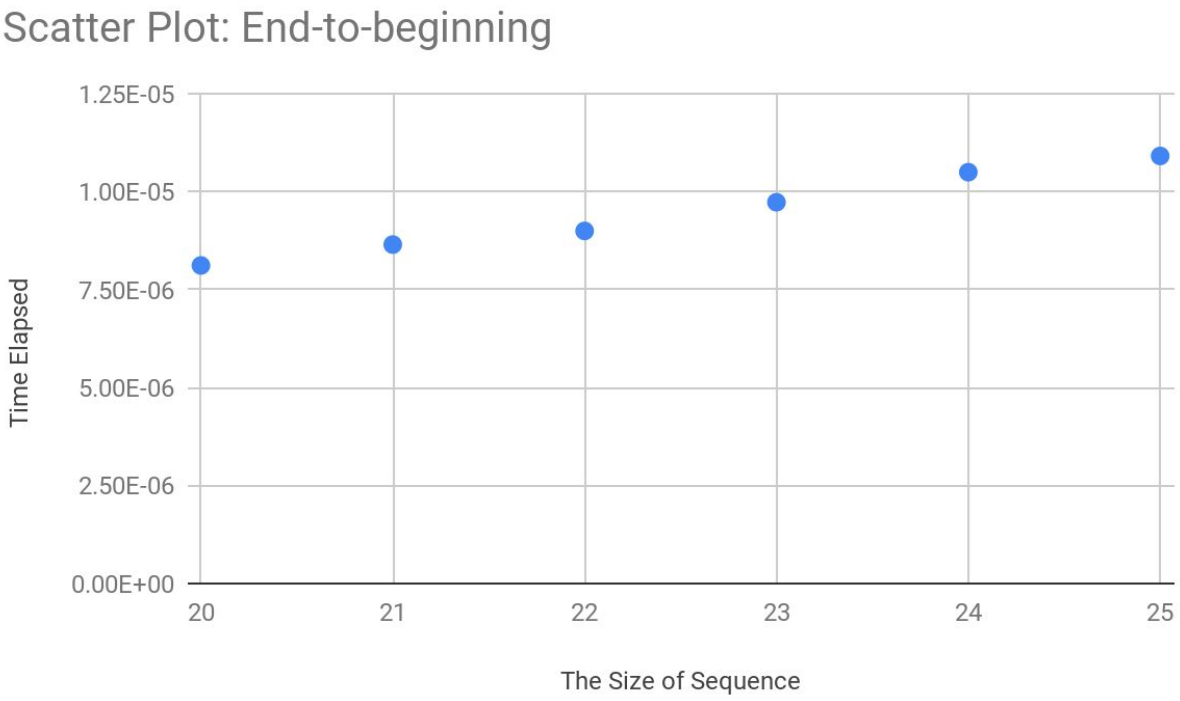
Author: Hyun Woo Kim, Raymond Laurente
Professor: Krutik Amin
Subject: CPSC335 Algorithm, Proj 2

In this assignment our group was tasked to implement longest decreasing subsequence algorithm starting from the end-to-beginning. We were also asked to compare two version of the subsequence: end-to-beginning and powerset method. We concluded that powerset method was meaningless and unfit to be useful in our algorithm. As you read through this data analysis we will further investigate the two algorithm performances.

In this data Analysis we will cover

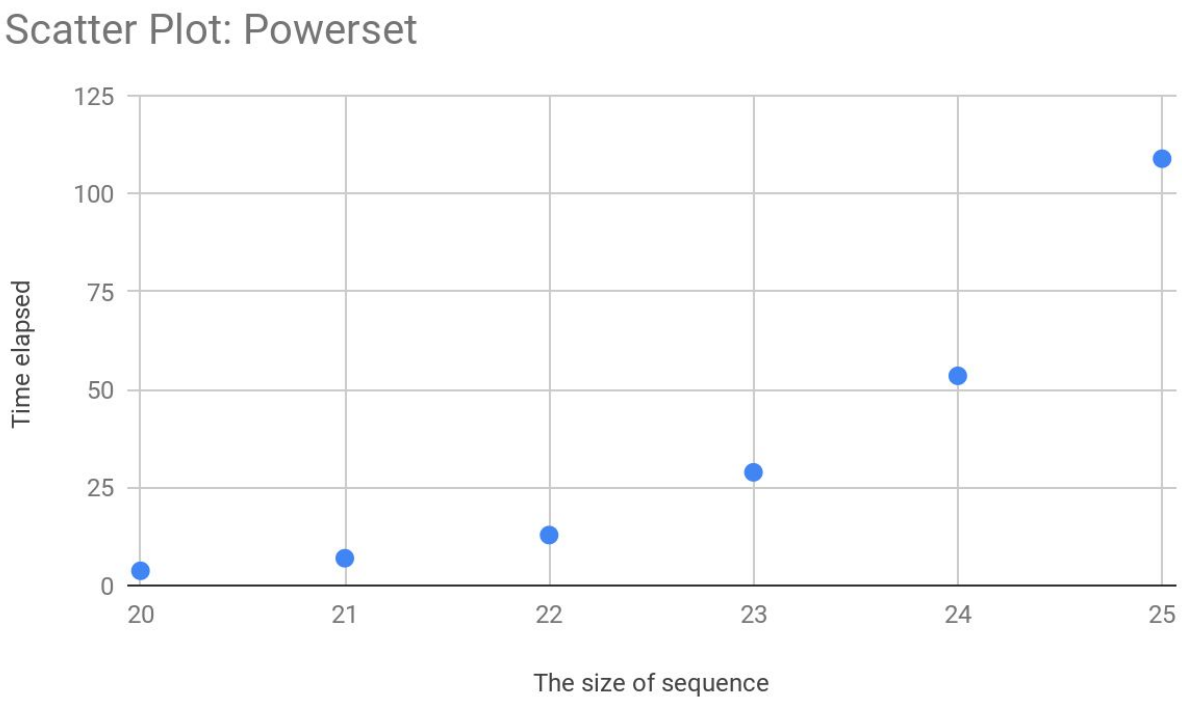
1. **End-to-beginning Algorithm**
    a. **Scatter plot**
2. **Powerset method**
    a. **Scatter plot**
3. **Empirically-observed time efficiency data**
    a. **End-to-beginning**
    b. **Powerset method**
4. **Pseudocode**
5. **Conclusion**

End-to-beginning Algorithm



As you can observe, depending on the size of the sequence the big O notation of the algorithm goes up gradually and consistently with the size of the sequence. You can also tell that the time elapsed time goes up gradually and concurrently with the size of the sequence. This is a good method and logical algorithm

Powerset Algorithm



This is the powerset method. As you can observe depending on the size of the sequence the time elapsed doubles as size increases. This is an example of bad algorithm. Due to the fact that time increases as size increment imagine if the size of sequence was 100 it would take too long to compute time elapsed. Therefore, powerset method is bad and is not fit to be a logical algorithm.

Empirically-observed time efficiency data

End-to-beginning

```
A:      for i=n-2 to i >= 0; i--
B:              For j = i+1 to j <n; j++
C:                      if(A[j] < A[i) && H[j] >= H[i])
                                H[i] = H[j] +1;
```

// We would first take care of the most inner source code
C: 3 + max(2,0) = 3 + 2 = 5;
B: [[(n-i+1)/1] + 1] * 5 = 5n- 5i + 10
A: ∑5n-5i+10

Therefore, O(n^2)

Powerset Method

```
        while(true)
A:              if(stack[k] < n){
                        Stack[k+1] = stack[k] +1
                        ++k
                }
B:              Else{
                  Stack[k-1]++
                  K--;
```

// step count
B: 2
A: 1 +max(3,0) = 4;
Total = 6;

6 +∑(depending on how many times we are running our code)
while(true) the way our code is written for true is when there is when decreasing subsequence is existing. Since our number is generated randomly to conclude our step count is arbitrary. Therefore we can conclude that our step count for while loop is powerset. 2 to the power of n.

Therefore, O(2^n)

Pseudocode

```
Def boo is_decreasing(sequence A)
        For i =1 to A.size do
                if(A[i-1] < A[i])
                        Return false
                Return true
Def sequence longest_decreasing_end_to_beginning(sequence A)
Def n = A.size()
Def vector H(n,0)
For i = n-2 to i = 0 do
        For j = i+1 to j < n do
                if(A[j] < A[i] && H[j] >= H[i]
                        H[i] = H[J] +1
Def max = max_element(H.begin(), H.end()+1)
Def  vector R(max)
Def index = max-1;
Def j =0;

For i to n do
        if(H[i] == index)
                R[j] = A[i]
                J++
                Index--
Return sequence(R.begin(), R.begin()+max)

Def sequence longest_decreasing_powerset(sequence A)
if(is_decreasing(candidate == true && candiate.size() > best.size()
        Best = candidate

Return best
```

Conclusion

Comparing our empirical analysis and mathematically-derived big-O efficiency was very difficult. Our group had short coming on calculating and formulating a correct step count for the powerset. We concluded that the efficiency of empirical analysis and mathematically-derived big O notation was consistent. Based from the empirical analysis we also concluded that the end-to-beginning method had much better performances and best worst case scenario algorithm. On the contrary, powerset method was not fit to be the algorithm because each added size of the sequence will double the time elapsed of the algorithm.