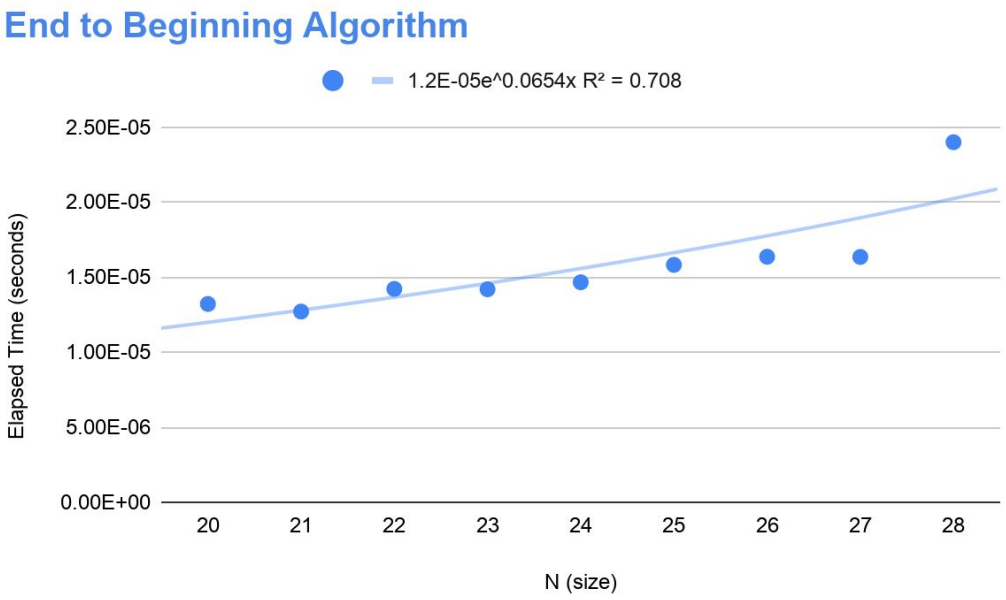
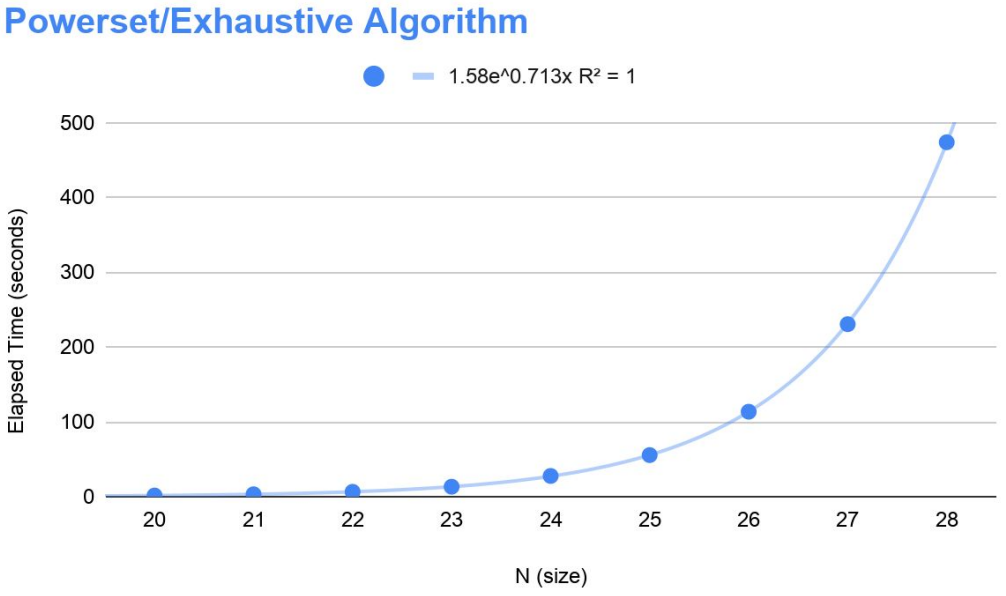
Project 2 Analysis

Group Members: Winson Gin winsongin@csu.fullerton.edu

Scatterplot:





Pseudocode:

longest_nonincreasing_end_to_beginning

```
const size_t n = A.size(); ---------- 1
std::vector<size_t> H(n, 0);
for i = n-2 to 0 do {
        for j = i+1 to n do {
                if(A.at(i) >= A.at(j) && H.at(i) <= H.at(j)){ ---------- 7
                        H.at(i) = H.at(j)+1; ---------- 4
                }
        }
}

auto max = *std::max_element(H.begin(), H.end()) + 1; ---------- 2
std::vector<int> R(max);

size_t index = max-1, j = 0; ---------- 3
for i = 0 to n-1do { ---------- ((n-1)-0/1) + 1 = n
        if(H[i] == index) { ---------- 1
                R.at(j) = A.at(i); ---------- 3
                index--; ---------- 2
                j++; ---------- 2
        }
}

return sequence(R.begin(), R.begin() + max); ---------- 2
}
```

Proof

$$S.C. = 1 + \left(\sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} 11\right) + 2 + 8 + n * \left(1 + \max(7,0)\right) + 2$$

$$= 8 + \left(\sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} 11\right) + n * 8$$

$$= 8 + 8n + \left(\sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} 11\right)$$

$$\sum_{j=i+1}^{n-1} 11 = 11\left((n-1)-(i+1)+1\right)$$

$$= 11(n-i-1)$$

$$= 8 + 8n + \left(\sum_{i=0}^{n-1} 11(n-1) - \sum_{i=0}^{n-1} 11i\right)$$

$$= 8 + 8n + \left(11(n-1)(n-1) - \frac{11(n-1)(n-1+1)}{2}\right)$$

$$= 8 + 8n + \left(11(n^2-2n+1) - \frac{11(n-1)(n)}{2}\right)$$

$$= 8 + 8n + \left(11n^2 - 22n + 11\right) - 11 \cdot \left(\frac{n^2-n}{2}\right)$$

$$= 8 + 8n + 11n^2 - 22n + 11 - 11 \cdot \left(\frac{n^2-n}{2}\right)$$

$$= \cancel{11n^2 + 2}$$

$$= 11n^2 - 14n + 19 - \frac{11n^2+n}{2} = \frac{22n^2 - 28n + 38}{2} - \frac{11n^2+n}{2} = \frac{11n^2 - 29n + 38}{2}$$

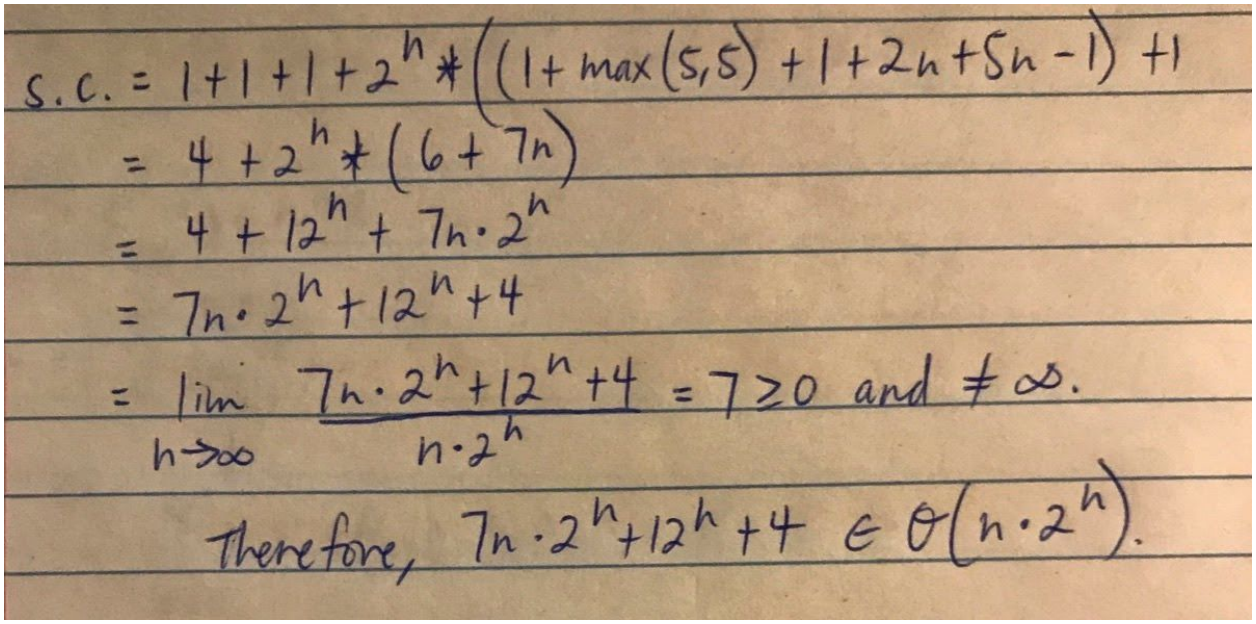
$$\lim_{n\to\infty} \frac{11n^2-29n+38}{2} = \frac{1}{2} \cdot \lim_{n\to\infty} \frac{11n^2-29n+38}{n^2} = \frac{11}{2} \geq 0 \text{ and } \neq \infty.$$

Therefore, $\frac{11n^2-29n+38}{2} \in O(n^2)$.

longest_nonincreasing_powerset

```
const size_t n = A.size(); ---------- 1
sequence best;
std::vector<size_t> stack(n+1, 0); ---------- 1
size_t k =0; ---------- 1
while(true) do { ---------- 2^n
        if(stack[k] < n) {  -------- 1
                stack[k+1] = stack[k] + 1; ---------- 3
                ++k; ---------- 2
        }
        else {
                stack[k-1]++; --------- 3
                k--; ---------- 2
        }

        if(k == 0) { ---------- 1
                break;
        }

        sequence candidate;

        For i = 1 to k do { -------- ((n-1)/1) + 1 = n
                candidate.push_back(A[stack[i]-1]); --------- 2
        }

        // s.c. of if-condition:  5n-2 + max(1, 0) = 5n - 1
        if(is_nonincreasing(candidate) && candidate.size() > best.size()) { ----------  5n-2
                best = candidate; --------- 1
        }
}
return best; ---------- 1
}
```

Proof



Answers to questions:

Running speed:

There is a noticeable difference between the running speed for the End to Beginning algorithm and the Powerset/Exhaustive Algorithm. When the sizes (N) are smaller, the difference between the two are not as noticeable. However, when the sizes (N) are larger, the difference is quite significant. The End to Beginning Algorithm stays in range of seconds, while the Powerset/Exhaustive algorithm starts to have a running speed of minutes.

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

The fit lines on the scatter plots are consistent with these efficiency classes because the difference in the order of growth is visible for each scatterplot. The fit line for the End to Beginning Algorithm shows a fit line that looks more linear and less of a steep curve like the Powerset/Exhaustive Algorithm. The Powerset/Exhaustive Algorithm has a fit line that you can clearly see curves upwards sooner than the End to Beginning Algorithm. This shows that the

Powerset/Exhaustive Algorithm is less efficient because when comparing the two algorithms using the same N (size), the Powerset/Exhaustive Algorithm is slower.

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

From running the subsequence_timing.cpp file, you are able to see that they both produce the same output. Exhaustive optimization algorithms are feasible to implement since it's not overly-complicated to write the implementation. However, because the Powerset/Exhaustive algorithm is exponential and runs much slower than the End to Beginning algorithm, it is clearly not the better option. Keep in mind that this is only with sizes ranging from 20 to 28. For practical use, sizes would be much larger than this and the run times for the Powerset/Exhaustive algorithm would be even slower.