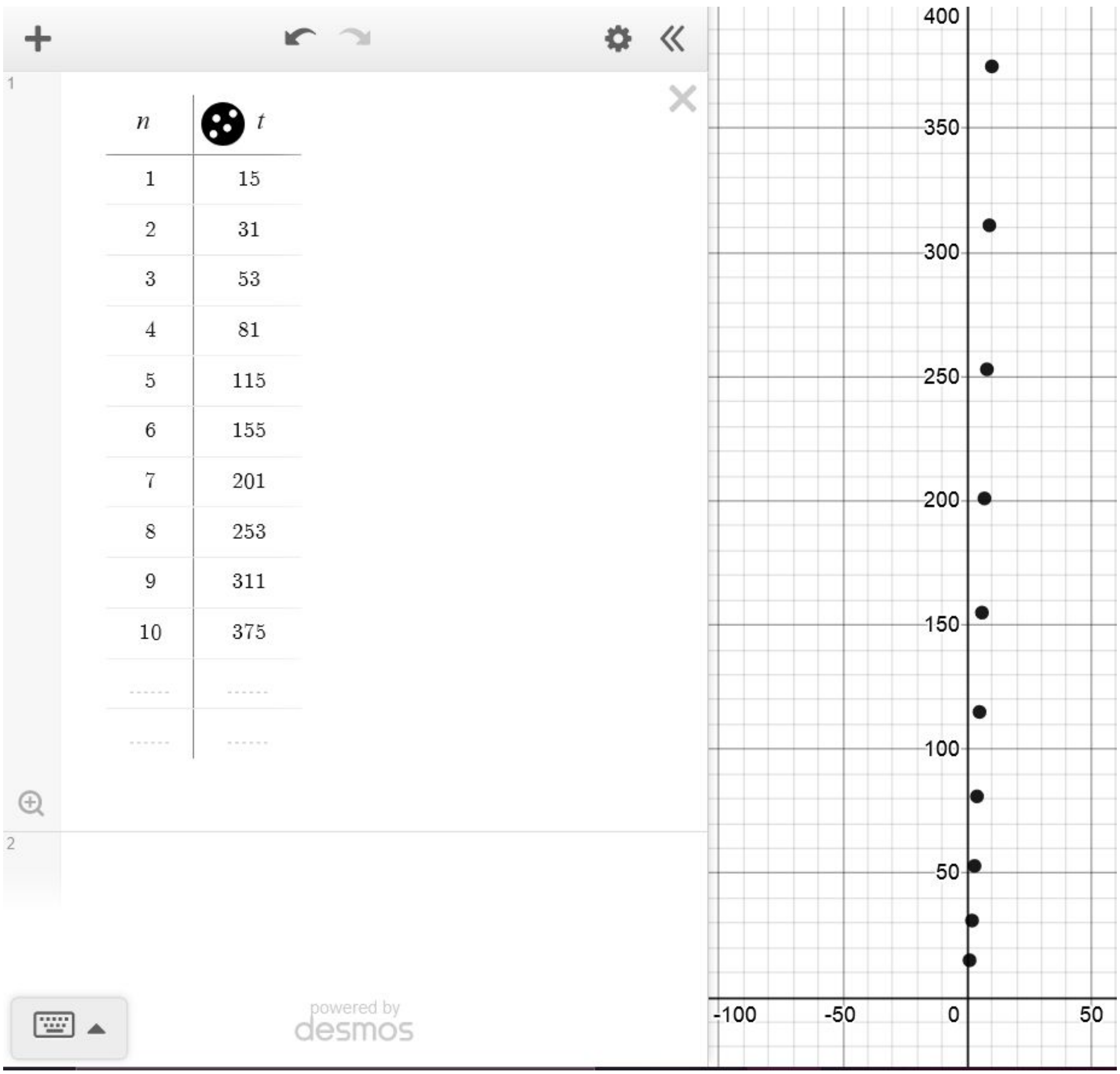
William Yung: yungwilliam@csu.fullerton.edu
Michael Lam: micheallam@csu.fullerton.edu
CPSC 335

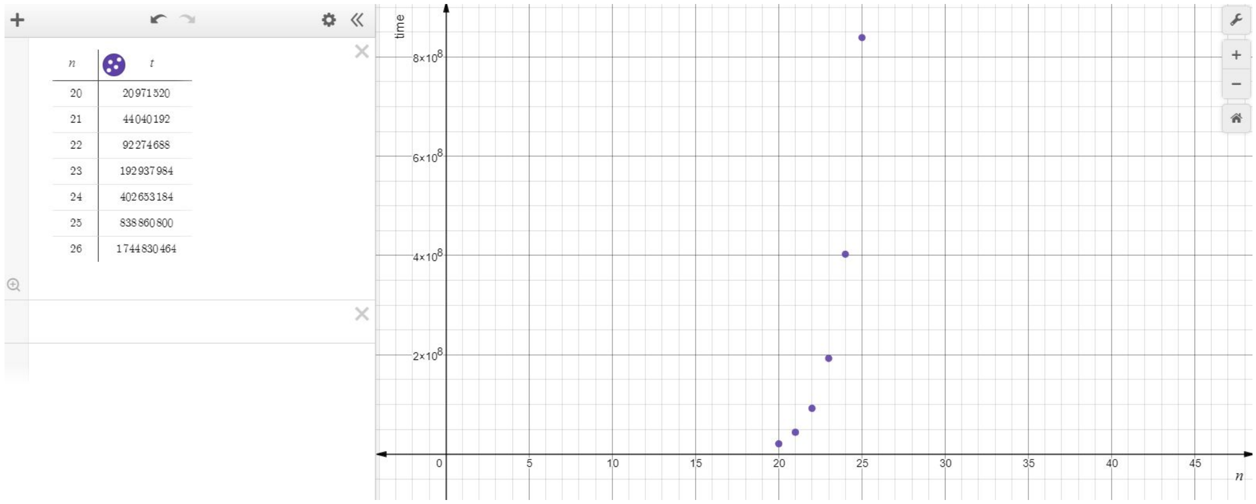
Project 2

Scatterplot:

End-to-Beginning:



Exhaustive:



n	t
20	20971520
21	44040192
22	92274688
23	192937984
24	402653184
25	838860800
26	1744830464

Pseudocode:

End-to-Beginning:

```
def end_to_beg(A):
        n = A.getSize()
        H[n] = 0
        for (i = n-2; i >= 0; i--)
                for (j = i+1; j < n; j++)
                        if (A[i] >= A[j] and H[i] <= H[j])
                                then H[i] = H[j] + 1;

        max = H.getLargestVal + 1
        R[max]
        index = max - 1
        l = 0;
        for(k = 0; k < n; k++)
                if (H[k] == index)
                        then R[l] = A[k]
                        l++
                        index--

        return R
```

Exhaustive:

```
def longest_powerset(A):
        n = A.size()
        seqeunce best
        stack[n+1] = 0
```

```
k = 0
while(1):
    if(stack[k] < n):
        stack[k+1] = stack[k] + 1
        K++
    else:
        stack[k-1] ++
        K--

    if(k == 0)
        Break; //out of the loop

    Sequence temp
    for(i = 1 to k):
        temp.push_back(A[stack[i]-1])

    if(best.empty or (nonincreasing(temp) and temp.size() > best.size()))
        best = temp

return best
```

Efficiency:

End-to-Beginning:

```
def end_to_beg(A):
        n = A.getSize() - 1 tu
        H[n] = 0 - 1 tu
        for (i = n-2; i >= 0; i--)
              for (j = i+1; j < n; j++)
                     if (A[i] >= A[j] and H[i] <= H[j]) - 3tu
                          then H[i] = H[j] + 1; - 3tu

        max = H.getLargestVal + 1 - 1tu
        R[max] - 1tu
        index = max - 1 - 2tu
        l = 0; - 1tu
        for (k = 0; k < n; k++) - n+1 tu
              if (H[k] == index) - 1tu
                    then R[l] = A[k] - 1tu
                    l++ - 1tu
                    index-- - 1tu

        return R - 1tu
```

$3n^2 + 3n - 6$ tu

$1 + 1 + 3n^2 + 3n - 6 + 1 + 1 + 2 + 1 + (n+1)(4)$

$3n^2 + 3n - 6 + 7 + 4n + 4$

$\boxed{SC = 3n^2 + 7n + 5 \text{ tu}}$

Exhaustive:

```
Exhaustive:

    def longest_powerset(A):
        n = A.size()          — 1 tu
        seqeunce best         — 1 tu
        stack[n+1] = 0        — 1 tu
        k = 0                 — 1 tu
        while(1):
            if(stack[k] < n):     — 1 tu
                stack[k+1] = stack[k] + 1     — 2 tu
                K++               — 1 tu
            else:
                stack[k-1] ++     — 1 tu
                K--               — 1 tu

            if(k == 0)            — 1 tu
                Break; //out of the loop   — 1 tu

            Sequence temp
            for(i = 1 to k):      — n tu.
                temp.push_back(A[stack[i]-1])   — 1 tu

            if(best.empty or (nonincreasing(temp) and temp.size() > best.size()))  — 3 tu
                best = temp           — 1 tu.

        return best               — 1 tu
```

(lines 4–7) 4 t.u

(lines 9–17) 8 t.u.

(lines 20–21) n tu.

(lines 23–24) 4 tu

$$S.C. = 1 + (2^n \cdot (n + 12)) + 4 + 1 = 5 + (n2^n + 24^n)$$
$$= n2^n + 24^n + 5$$
$$= \mathcal{O}(n + 2^n)$$

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 a noticeable difference in the running speed when the n is increased more than 20. End-to-Beginning ran faster than the power set.

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

- The lines on the scatter plots are consistent with the efficiency class due to the sudden spike in time that power set has when compared to the end-to-beginning algorithm.

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

- It is inconsistent with the hypothesis since powerset is a lot slower in this case. The more n was increased, the longer the program took to compile. Sometimes it also seemed like it froze.