Aarib and Aatib Abdullah

CWID: [redacted]

Project 2

The two scatter plots and their data is appended to the end of this document.

Answer to problem 3:

3. Answers to the following questions, using complete sentences.

a. Provide pseudocode for your **two** algorithms.

See yellow documents appended to this paper

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 End-to-Beginning algorithm has the efficiency class of O(n^2)

The Longest increasing subsequence algorithm has the efficiency class of 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?

No, it comes with no surprise at all. As predicted by the proven efficiency classes of End-toBeginning and Longest increasing subsequence algorithms and their best fit lines (equation) for scatterplots, longest increasing subsequence algorithm will take much longer as "n" increases than Endto-Beginning algorithm. One: O((2^n)*n) > O(n^2)   and Two: c's rise in time of execution is much quicker than End-to-Beginning algorithm rate of rise in time.

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

Yes, and as stated earlier. The trendlines compared in one chart shows that the End-toBeginning algorithm does not rise in time of execution as quick as longest increasing subsequence algorithm as "n" increases. The best fit line for End-to-Beginning algorithm is a polynomial of order 2 and longest increasing subsequence algorithm is exponential (per excel graph, check below).

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

Justify your answer.

The evidence are consistent with the hypothesis, since the rate of growth of best fit line for Endto-Beginning algorithm is a polynomial of order 2 and best fit line for longest increasing subsequence algorithm is exponential (per excel graph, check below). The behavior of each line matches with their individual algorithms.

```
End-to-Beginning(sequence/array A):
  n = sizeof(A) //1
  H[n] // 1
  for(i=0; i<n; i++) //n
    H[i]=0 //1

  For(i=n-2; !(i<0); i--): //n-2
    For(j=i+1; j<n; j++): // n - (i+1)+1
      if(A[i] < A[j] && H[i]≤H[j]): //3
        H[i] = H[j]+1 //2
  max = maxelementof(H)+1 //2
  R[max] //1
  r=0 //1
  Less = max-1 //2
  For(i=0; i<n && r!=max; i++): //n
    if(H[i]== Less) //1 + max(3,0) = 4
      R[r] = A[i] //1
      r++ //1
      Less-- //1
  return R //1
```

$$\Rightarrow \sum_{i=0}^{n-2}(n-i) \quad + \max(2,0) = 5$$

$$(n-2)n - \frac{(n-2)(n-1)}{2}$$

$$(n^2-2n) - \cancel{\tfrac{1}{2}}\left(\frac{n^2}{2}-\frac{3n}{2}+\frac{2}{2}\right)$$

$$\frac{1}{2}n^2 - \frac{1}{2}n - 1$$

$$4 \cdot n = 4n$$

$$T(n) = n + \left(\tfrac{1}{2}n^2 - \tfrac{1}{2}n - 1\right)\cdot 5 + 4n + 9$$

$$T(n) = n + \tfrac{5}{2}n^2 - \tfrac{5}{2}n - 5 + 4n + 9$$

$$T(n) = \tfrac{5}{2}n^2 + \tfrac{5}{2}n + 4$$

Where $f(n)=n^2$ Prove $T(n)=\Theta(n^2)$ by limit THM. AS $L>0$ AND $<\infty$.

$$\lim_{n\to\infty}\frac{T(n)}{f(n)} = \frac{\frac{d}{dn}\left(\frac{5}{2}n^2+\frac{5}{2}n+4\right)}{\frac{d}{dn}(n^2)} \Rightarrow \lim_{n\to\infty}\frac{5n+\frac{5}{2}}{2n} = \frac{\infty}{\infty}$$

$$\lim_{n\to\infty}\frac{\frac{d}{dn}(5n+\frac{5}{2})}{\frac{d}{dn}(2n)} = \frac{5}{2} > 0 \text{ AND } \frac{5}{2} < \infty$$

$$\therefore \frac{5}{2}n^2 + \frac{5}{2}n + 4 = \Theta(n^2)$$

$2^2=4$

| KB | | |
|---|---|---|
| 00 | ~~KB~~ | 0 |
| 10 | K | 1 |
| 01 | B | 1 |
| 11 | KB | 2 |

$\frac{2^n}{2}$

$2$

The idea of creating (generating) every possible subse ences from a sequence

| | | a b c | | 0 8 4 12 | | # ~~of~~ index searching through |
|---|---|---|---|---|---|---|
| 0 | ∅ = | 0 0 0 | 1 | 0 0 0 0 | = ∅ | ~~0~~ |
| 1 | a = | 1 0 0 | 2 | 1 0 0 0 | = 0 | ~~1~~ |
| 1 | b = | 0 1 0 | 3 | 0 1 0 0 | = 8 | ~~1~~ |
| 2 | ab = | 1 1 0 | 4 | 1 1 0 0 | = 0 8 | ~~2~~ |
| 1 | c = | 0 0 1 | 5 | 0 0 1 0 | = 4 | ~~1~~ |
| 2 | ac = | 1 0 1 | 6 | 1 0 1 0 | = 0 4 | ~~2~~ |
| 2 | bc = | 0 1 1 | 7 | 0 1 1 0 | = 8 4 | ~~2~~ |
| 3 | abc = | 1 1 1 | 8 | 1 1 1 0 | = 0 8 4 | ~~3~~ |
| | | | 9 | 0 0 0 1 | = 12 | ~~1~~ |
| | | | 10 | 1 0 0 1 | = 0 12 | ~~2~~ |
| | | | 11 | 0 1 0 1 | = 8 12 | ~~2~~ |
| | | | 12 | 1 1 0 1 | = 0 8 12 | 3 |
| | | | 13 | 0 0 1 1 | = 4 12 | ~~2~~ |
| | | | 14 | 1 0 1 1 | = 0 4 12 | ~~3~~ |
| | | | 15 | 0 1 1 1 | = 8 4 12 | ~~3~~ |
| | | | 16 | 1 1 1 1 | = 0 8 4 12 | ~~4~~ |

$8(4)$

$\frac{2^4}{2}(4)$ 16

$8(4)$

$\frac{n}{2}($

~~48~~(3) indices searched

~~8~~ n

$\frac{2^n}{2}(n)$

$n=3$ # sub seq = 8 $n=4$ # sub seq = 16 AND $n=2$ # sub seq = 4

So the patter he is $2^n$ of subsequences of a sequence.

AND searching through each index takes steps $T(n)=2^{n-1}\cdot n$

PSEUDOCODE FOR LONGEST INCREASING POWERSET

generates (subsequences)

this psuedo code is the

```
S = <subsets of a sequence>  // to generate per last page takes about 2^n time
best = Ø , is_best = TRUE  //2
For i in S:  // 2^n
    if( i.size() > best.size() ):  //1
        For(j=1; j< i.size(); j++):  //n
            if( i[j-1] >= i[j] ):  // 1 + max(2,0) = 3
                is_best = FALSE  //1
                break  //1

    if (is_best == FALSE):  //1+max(2,0)=3
        best = i  //1
        is_best = TRUE  //1
```

$T(n) = 2 + 2^n \cdot 1 \cdot n \cdot 3 + 3 = 3 \cdot 2^n \cdot n + 5$

To prove $T(n) = O(f(n))$ when $f(n) = 2^n \cdot n$

$$\lim_{n\to\infty} \frac{T'(n)}{f(n)} = \frac{\frac{d}{dn}(3 \cdot 2^n \cdot n + 5)}{\frac{d}{dn}(2^n \cdot n)} \to \lim_{n\to\infty} \frac{3 \, 2^n \ln(2) \cdot n + 3 \cdot 2^n}{2^n \cdot \ln(2) \cdot n + 2^n} = \frac{3\cancel{(2^n \ln(2) \cdot n + 2^n)}}{\cancel{(2^n \ln(2) n + 2^n)}}$$

$= 3$

$\therefore 3 \cdot 2^n \cdot n + 5 = O(2^n \cdot n)$

| | 1st trial | | 2nd trial | | 3rd trial | | AVG | | SD | |
|---|---|---|---|---|---|---|---|---|---|---|
| size-n | End-to-beginning | Increasing susequence | End-to-beginning | increasing susequence | End-to-beginning | increasing susequence | End-to-beginning | increasing susequence | End-to-beginning | increasing susequence |
| 2 | 2.40E-05 | 3.84E-05 | 2.40E-05 | 3.24E-05 | 3.35E-05 | 4.27E-05 | 2.72E-05 | 3.78E-05 | 5.50E-06 | 5.14E-06 |
| 3 | 2.26E-05 | 5.99E-05 | 3.63E-05 | 6.24E-05 | 3.39E-05 | 6.42E-05 | 3.09E-05 | 6.22E-05 | 7.33E-06 | 2.13E-06 |
| 4 | 2.57E-05 | 1.42E-04 | 2.75E-05 | 1.35E-04 | 3.14E-05 | 2.95E-04 | 2.82E-05 | 1.91E-04 | 2.89E-06 | 9.01E-05 |
| 5 | 2.96E-05 | 2.92E-04 | 3.31E-05 | 4.08E-04 | 3.14E-05 | 4.38E-04 | 3.14E-05 | 3.79E-04 | 1.76E-06 | 7.72E-05 |
| 6 | 3.46E-05 | 6.48E-04 | 3.46E-05 | 6.78E-04 | 5.61E-05 | 1.18E-03 | 4.17E-05 | 8.36E-04 | 1.24E-05 | 3.00E-04 |
| 7 | 4.37E-05 | 1.90E-03 | 3.67E-05 | 1.43E-03 | 3.70E-05 | 1.56E-03 | 3.91E-05 | 1.63E-03 | 3.97E-06 | 2.47E-04 |
| 8 | 6.24E-05 | 4.04E-03 | 4.23E-05 | 3.32E-03 | 7.41E-05 | 5.60E-03 | 5.96E-05 | 4.32E-03 | 1.61E-05 | 1.17E-03 |
| 9 | 5.57E-05 | 7.19E-03 | 4.73E-05 | 7.10E-03 | 7.86E-05 | 8.28E-03 | 6.05E-05 | 7.52E-03 | 1.62E-05 | 6.59E-04 |
| 10 | 5.04E-05 | 1.58E-02 | 5.25E-05 | 1.61E-02 | 7.26E-05 | 2.12E-02 | 5.85E-05 | 1.77E-02 | 1.23E-05 | 3.06E-03 |
| 13 | 7.44E-05 | 1.70E-01 | 9.63E-05 | 1.75E-01 | 7.48E-05 | 1.71E-01 | 8.18E-05 | 1.72E-01 | 1.25E-05 | 2.72E-03 |
| 15 | 1.08E-04 | 7.66E-01 | 8.36E-05 | 7.22E-01 | 9.94E-05 | 7.25E-01 | 9.70E-05 | 7.37E-01 | 1.24E-05 | 2.45E-02 |
| 18 | 1.16E-04 | 5.84E+00 | 1.13E-04 | 5.81E+00 | 1.22E-04 | 5.74E+00 | 1.17E-04 | 5.80E+00 | 4.46E-06 | 4.81E-02 |
| 20 | 1.38E-04 | 2.47E+01 | 1.37E-04 | 2.44E+01 | 1.42E-04 | 2.82E+01 | 1.39E-04 | 2.58E+01 | 2.94E-06 | 2.15E+00 |
| 21 | 0.000149509 | 52.6101 | | | | | | | | |