CIS-277 Data Structures and Algorithms

Review: Analysis of Algorithms

1. One way to determine the execution speed of an algorithm is to implement it in a programming language, execute the program, and directly measure the actual time it takes to complete. Briefly explain the possible problem(s) with this approach.

   ↳ speed of cpu we are using
   ↳ # of programs we are running

2. For each of the following code segments:
   - identify the statements that are "mice".
   - identify the Big-0 run-time (show how you arrived at the answer)

   a.
   ```
   sum_sqr = 0;                                  - O(1) - mice
   for( ct = 0; ct < size; ++ ct )
           sum_sqr += data[ct] * data[ct];       O(n) - B-O
   mean_sqr = sum_sqr / size;                    O(1) - mice
   ```

   b.
   ```
   sum = 0                                       - O(1)
   for( ct = 1; ct <= size; ++ct )
           sum += ct;                            O(n)
   product = 1;                                  O(1)
   for( ct = 1; ct <= size; ++ct )
           product *= ct;                        O(n)
   difference = product – sum;                   O(1)
   ```
   O(n) + O(n) = 2 O(n) = O(n)

   c.
   ```
   max = data[0, 0];                             O(1)
   for( ctr = 0; ctr < n; ++ctr )                O(n)
           for( ctc = 0; ctc < n; ++ctc          O(n)
                   if ( data[ctr] [ctc] > max )
                           max = data[ctr] [ctc]; O(1)
   ```
   O(n) x O(n) = O(n²)

   d.
   ```
   max = data[0, 0];                             O(1)
   for( ctr =0; ctr < 5; ++ctr )                 O(1) - this loop runs only 5 times no matter
           for( ctc = 0; ctc < 5; ++ctc )        O(1)   the value of ctr
                   if ( data[ctr] [ctc] > max )
                           max = data[ctr] [ctc]; O(1)
   ```
   O(1)

   e.
   ```
   sum = 0;                                      O(1)
   for( ctr = 0; ctr < 3; ++ctr )                O(1)
           for( ctc = 0; ctc < n; ++ctc )        O(n)
                   sum += data[ctr] [ctc];       O(1)
   ```
   O(n)

   f.
   ```
   total =0;                                     O(1)
   for( ctr = 0; ctr < n; ++ctr )                O(n)
           for( ctc = ctr + 1; ctc < n; ++ctc )  O(n)
                   total += data[ctr] [ctc];     O(1)
   ```
   O(n²)

g.
```
found = 0;
in.open( "a:client.dta:, ios::binary );
in.read((char*)&client, sizeof(client));
while( !in.eof( ) && !found )
        if ( client.id == id )
                found = 1;
        else
                in.read((char*)&client, sizeof(client));
in.close( );
```

h.
```
in >> size;
for( ctr = 0; ctr < size; ++ctr )
        for( ctc = 0; ctc < size; ++ctc )
                in >> score[ctr][ctc];
for( ctr = 0; ctr < ctr < size; ++ctr ) {
        total = 0;
        for( ctc =0; ctc < size; ++ctc )
                total += score[ctr] [ctc];
        r_mean[ctr] = total / column;
}
```

i.
```
cout << "Enter the number of data values;     ";
cin >> size;
for ( ct = 0; ct < size; ++ct ) {
        cout << "Value:
        cin >> data[ct];
}
sum = 0;
for ( ct = 0; ct < size; ++ct )
        sum += data[ct];
mean = sum / size;
```

j.
```
for ( ctr = 0; ctr < n; ++ ctr ) {
        sum[ctr] = 0;
        for ( ctc = 0; ctc < n; ++ctc )
                sum[ctr] += data[ctr] [ctc];
}
for ( ctr = 0; ctr < n; ++ctr )
        cout << "Sum for row ", ctr, " is:       ", sum[ctr] << endl;
```

k.
```
factorial = 1;
for ( ct = 1; ct <= n; ++ ct )
        factorial *= ct;
```
O(n)

```
sum =0;
above = 0;
for( ctr = 0; ctr < n; ++ctr )
        for( ctc =0; ctc < n; ++ctc )
                if ( data[ctr][ctc] > limit ) {
                        sum += data[ctr][ctc];
                        ++above;
mean_above = sum / above;
```
$O(n^2)$

Determine the Big-0 run-time of:

a. selection sort

b. insertion sort

c. binary search (worst case behavior)

3. Suppose that it takes 0.004 seconds to run a program on a test data set of size n = 200. Assume that the number of items in the actual data set is 4000. What would be the expected run time (show your work) if the program is applied to the actual data set and the underlying algorithm is:

a) $0.004 \quad 200 \quad x \quad 4000$

b) $0.004 \quad 200^2 \quad x \quad 4000^2$

a. 0(n) 0.08 b. $0(n^2)$ 1.6

$x = \frac{0.004 \cdot 4000}{200} =$ $x = \frac{0.004 \times 4000^2}{200^2}$

4. Assume that it takes 0.0012 seconds to run a program on a test data set of size n = 50. Assume that the number of items in the actual data set is 80,000. What would be the expected run-time (show your work) if the program is applied to the actual data set and the underlying algorithm is:

$50 \quad 0.0012 \quad 80000 \quad x$

$x = \frac{80000 \times 0.0012}{50}$

a. 0(n) 1.92 b. $0(n^2)$ 3072

5. Assume that an algorithm consists of two parts, one part is $0(n^2)$ and the other in 0(n). What is the Big-0 run-time for the entire algorithm?

6. The Paradox of Speed:

   Suppose that computers become 16 times faster. If the same amount of computer time is to be used, how much can the data set size be increased if the underlying algorithm is:

   a. $0(n)$ b. $0(n^2)$ c) $O(1)$ d) $\log_2 n$

7. Determine the answer and explain why it is the answer:

   a. $0(n) + 0(n) = ?$

   b. $0(n) + 0(n^2) = ?$

   c. $0(\log n) + 0(n) = ?$