You may work in pairs for this assignment. If you choose to work with a partner, make sure only one of you submits a solution, and you paste a copy of the Partners Template that contains the names and PIDs of both students at the beginning of the file you submit.

You will submit your solution to this assignment to the Curator System (as `HW03`). Your solution must be either a plain text file (e.g., NotePad++) or a typed MS Word document; submissions in other formats will not be graded.

Partial credit will only be given if you show relevant work.

In all questions about complexity, functions are assumed to be nonnegative.

1. [36 points] The analysis of a certain algorithm leads to the following complexity function (for the average case):

$$T(N) = 3N\log^2(N) + 5N^2\log(N) + \log(N) + 17N + 2$$

Several computer science students offer their conclusions about the algorithm (quoted below). For each conclusion, state whether it is correct, or incorrect, or could be either correct or incorrect, based on the given information, and give a brief justification of your answer; feel free to cite any relevant theorems from the course notes.

First of all, let's settle the issue of what Θ-category T(N) belongs to:

$$\lim_{N\to\infty} \frac{3N\log^2 N + 5N^2\log N + \log N + 17N + 2}{N^2\log N} = \lim_{N\to\infty}\left(\frac{3N\log^2 N}{N^2\log N} + \frac{5N^2\log N}{N^2\log N} + \frac{\log N}{N^2\log N} + \frac{17N}{N^2\log N} + \frac{2}{N^2\log N}\right)$$

$$= \lim_{N\to\infty}\left(\frac{3\log N}{N} + 5 + \frac{1}{N^2} + \frac{17}{N\log N} + \frac{2}{N^2\log N}\right)$$

$$= 0 + 5 + 0 + 0 + 0$$

$$= 5$$

Therefore, T(N) is Θ(N² log N).

a) The algorithm, on average, is $O(N^2)$.

False. On average, N² is strictly O(N² log N); in other words, the Θ-bound on T(N) is strictly larger than N², so N² cannot be an upper bound for T(N).

b) The algorithm, on average, is $\Omega(N)$.

True. N² log N is Ω(N); in other words, N is certainly a lower bound for N² log N.

c) The algorithm, on average, is $\Omega(N^2 \log N)$.

True. T(N) is Θ(N² log N); so by definition T(N) is Ω(N² log N).

d) The algorithm, on average, is $\Theta(N^2 \log N)$.

True. Proved above.

e) The algorithm, on average, is $O(N^3)$.

True. On average, T(N) is Θ(N² log N); and we can easily show that N² log N is O(N³):

$$\lim_{N\to\infty} \frac{N^2 \log N}{N^3} = \lim_{N\to\infty} \frac{\log N}{N} = 0$$

f) The algorithm, on average, is $\Theta(N \log^2 N)$.

False. We showed that on average T(N) is Θ(N² log N), and one part of that proof showed that

$$\lim_{N\to\infty} \frac{N \log^2 N}{N^2 \log N} = \lim_{N\to\infty} \frac{\log N}{N} = 0$$

And, that shows that N² log N is strictly Ω(N log² N).

g) In the worst case, the algorithm could be $\Theta(N^3)$.

True. We aren't given any direct information about the worst case; however, it's certainly possible that the worst case could be much worse than the average case, and we have already seen that N³ is a strict upper bound for the average case.

h) In the worst case, the algorithm could be $\Omega(\log N)$.

True. The worst case can certainly be no better than the average case, and the average case is Θ(N² log N), which is certainly Ω(log N). In fact, the worst case must be Ω(log N).

i) In the worst case, the algorithm must be $\Omega(N)$.

True. Similar logic to the previous part… worst cannot be better than average, and average is certainly Ω(N).

j) The algorithm's best case performance is $\Omega(N^2 \log N)$.

May be true, may be false. The best case cannot be worse than the average case, and it could be the same; if so, this is true. But, the best case might be considerably better than the average case; if so, this is false.

k) The algorithm's best case performance cannot be $\Omega(N \log N)$.

False. The best case could be anything equal to, or better than, the average case.

l) The algorithm's best case performance is strictly $O(\log N)$.

May be true, may be false. While unlikely, it's possible the best case is Θ(1), in which case this is true. On the other hand, the best case might be equal to, or worse than, log N.

2. [24 points] Suppose that an algorithm takes 30 seconds for an input of 2^{24} elements (with some particular, but unspecified speed in instructions per second). Estimate how long the same algorithm, running on the same hardware, would take if the input contained 2^{30} elements, and that the algorithm's complexity function is:

a) $\Theta(N)$
b) $\Theta(\log N)$
c) $\Theta(N \log N)$
d) $\Theta(N^2)$

Assume that the low-order terms of the complexity functions are insignificant, and state your answers in the form HH:MM:SS.S (hours, minutes, seconds, tenths of a second). Be sure to show supporting work.

It helps to start with the following observation. Since the algorithm takes 30 seconds for an input of size 2^{24}, we know that if the hardware can execute S instructions per second $T(2^{24}) / S = 30$ seconds.

a) Suppose T(N) = N (we can ignore low-order terms). Then

$$T(2^{30}) / S = 2^{30} / S = 2^6 * (2^{24} / S) = 64 * 30 \text{ seconds} = 32 \text{ minutes}$$

So, the running time would be about 00:32:00.0.

b) Suppose T(N) = log N. Then

$$T(2^{24}) / S = \log(2^{24}) / S = 24 / S$$

and

$$T(2^{30}) / S = \log(2^{30}) / S = 30 / S = 1.25 * (24 / S) = 1.25 * 30 \text{ seconds} = 37.5 \text{ seconds}$$

So, the running time would be about 00:00:37.5.

c) Suppose T(N) = N log N. Then

$$T(2^{24}) / S = 2^{24} \log(2^{24}) / S = 24 * 2^{24} / S$$

and

$$T(2^{30}) / S = 2^{30} \log(2^{30}) / S = 30 * 2^6 * 2^{24} / S = 2^6 * 5/4 * (24 * 2^{24} / S)$$
$$= 80 * (24 * 2^{24} / S) = 80 * 30 \text{ seconds} = 40 \text{ minutes}$$

So, the running time would be about 00:40:00.0.

d) Suppose $T(N) = N^2$. Then

$$T(2^{24}) / S = 2^{48} / S$$

and

$$T(2^{30}) / S = 2^{60} / S = 2^{12} * 2^{48} / S = 4096 * 30 \text{ seconds} = 2048 \text{ minutes}$$

So, the running time would be 34:08:00.0.

3. [18 points] Use theorems from the course notes to solve the following problems. Show work to support your conclusions.

a) Find the "simplest" function g(n) such that

$$f(n) = 17n^{3/2} + 3n\log n + 1000 \text{ is } \Theta(g(n))$$

Since

$$\lim_{n\to\infty} \frac{17n^{3/2} + 3n\log n + 1000}{n^{3/2}} = \lim_{n\to\infty}\left(\frac{17n^{3/2}}{n^{3/2}} + \frac{3n\log n}{n^{3/2}} + \frac{1000}{n^{3/2}}\right)$$
$$= \lim_{n\to\infty}\left(17 + \frac{3\log n}{n^{1/2}} + \frac{1000}{n^{3/2}}\right) = 17 + \lim_{n\to\infty}\left(\frac{3\log n}{n^{1/2}}\right)$$
$$= 17 + \lim_{n\to\infty}\left(\frac{3/n\ln 2}{(1/2)n^{-1/2}}\right) = 17 + \frac{6}{\ln 2}\lim_{n\to\infty}\left(\frac{1}{n^{1/2}}\right)$$
$$= 17$$

we see that f(n) is $\Theta(n^{3/2})$.

b) Find the "simplest" function g(n) such that

$$f(n) = 5n\log^2 n + 8n^2 \log n \text{ is } \Theta(g(n))$$

Since

$$\lim_{n\to\infty} \frac{5n\log^2 n + 8n^2\log n}{n^2\log n} = \lim_{n\to\infty}\left(\frac{5n\log^2 n}{n^2\log n} + \frac{8n^2\log n}{n^2\log n}\right)$$
$$= \lim_{n\to\infty}\left(\frac{5\log n}{n} + 8\right) = 8 + \lim_{n\to\infty}\left(\frac{5/n\ln 2}{1}\right)$$
$$= 8 + \frac{5}{\ln 2}\lim_{n\to\infty}\left(\frac{1}{n}\right) = 8$$

we see that f(n) is $\Theta(n^2 \log n)$.

c) Find the "simplest" function g(n) such that

$$f(n) = \sqrt{n} + \log(n+1000) \text{ is } \Theta(g(n))$$

Since

$$\lim_{n\to\infty} \frac{\sqrt{n} + \log(n+1000)}{\sqrt{n}} = \lim_{n\to\infty}\left(1 + \frac{\log(n+1000)}{\sqrt{n}}\right) = 1 + \lim_{n\to\infty}\left(\frac{1/(n+1000)\ln 2}{1/2n^{-1/2}}\right)$$
$$= 1 + \frac{2}{\ln 2}\lim_{n\to\infty}\left(\frac{n^{1/2}}{n+1000}\right) = 1$$

we see that f(n) is $\Theta(n^{1/2})$.

4. [12 points] Using the counting rules from the course notes, find the exact-count complexity function T(n) for the following algorithm. Show details of your analysis, and simplify your answer. In simplifying, you may discard the floor notation.

```
x = 100;                                  // 1
y = 0;                                    // 1
for (r = 1; r <= n; r++) {                // 1 before; 2 per pass; 1 to exit
   x = x + r;                             // 2
   for (c = 2; c <= r; c = 2*c) {         // 1 before; 3 per pass; 1 to exit
      if ( x > y / c )                    // 2
         y = y + r / c;                   // 3
      else
         y = y - c;                       // 2
   }
}
```

The costs per line are indicated above. They lead to the following formula for T(n):

$$T(n) = 3 + \sum_{r=1}^{n}\left(2+2+1+\sum_{p=1}^{\log r}\left(3+2+\max(3,2)\right)+1\right)+1$$

$$= \sum_{r=1}^{n}\left(\sum_{p=1}^{\log r} 8+6\right)+4$$

$$= \sum_{r=1}^{n}\left(8\log r+6\right)+4$$

$$= 8\left(\log 1+\log 2+\cdots+\log n\right)+6n+4$$

$$= 8\log\left(n!\right)+6n+4$$

Detail: the bounds on the inner loop come from the following observations. Number the passes through the inner loop starting at p = 1. Now, c takes on the values 2, 4, 8, and so forth. So, on pass p, $c = 2^p$. Therefore, the loop continues as long as

$$c \le r \leftrightarrow 2^p \le 2^{\log r} \leftrightarrow p \le \log r$$

This is similar to the analysis of binary search shown in the course notes.

5. [10 points] Use the definition of Θ to prove that, if b is any positive constant, then

$$\log(n+b) \text{ is } \Theta(\log(n))$$

proof:

From the definition of Θ, we must show that there are constants $C_1 > 0$, $C_2 > 0$ and $N > 0$ such that whenever $n > N$ we have that

$$C_1 \log n \le \log(n+b) \le C_2 \log n$$

Now, b > 0, so n + b > n; and log() is an increasing function, so log(n+b) > log(n). Therefore, we can show the left-hand inequality by taking $C_1 = 1$ and $N = 1$.
For the other side, suppose that n > b. (That's fair, since b is a constant.) Then n + b < n + n, and so

$$\log(n+b) \le \log(n+n) = \log(2n) = \log(n) + 1 \le \log(n) + \log(n) = 2\log(n)$$

So, the right-hand inequality holds if we take $C_2 = 2$ and say n > b.

Putting it all together, we have that

$$\text{if } n > b,\ \log n \le \log(n+b) \le 2\log n$$

Therefore, log(n + b) is Θ(log(n)).