You will submit your solution to this assignment to the Curator System (as `HW2`). Your solution must be either a plain text file (e.g., NotePad) or a MS Word document; submissions in other formats will not be graded.

Except as noted, credit will only be given if you show relevant work.

---

1. [20 points] Using the rules given in the course notes, perform an exact count complexity analysis, for the worst case, of the body of the following function. (Take the cost of `list.length` to be 2.)

```
int part(int[] list, int barrierIdx) {

   int barrier, maxIdx, temp;

   barrier = list[barrierIdx];          // 1  2
   maxIdx  = list.length - 1;           // 2  4

   temp = list[barrierIdx];             // 3  2
   list[barrierIdx] = list[maxIdx];     // 4  3
   list[maxIdx] = temp;                 // 5  2

   barrierIdx = 0;                      // 6  1

   for (int i = 0; i < maxIdx; i++) {   // 7  1 before, 2 per pass, 1 to exit
      if ( list[i] < barrier ) {        // 8  2
         temp = list[barrierIdx];       // 9  2
         list[barrierIdx] = list[i];    //10  3
         list[i] = temp;                //11  2
         barrierIdx++;                  //12  1
      }
   }
   temp = list[maxIdx];                 //13  2
   list[maxIdx] = list[barrierIdx];     //14  3
   list[barrierIdx] = temp;             //15  2

   return barrierIdx;                   //16  1
}
```

The input size is obviously the number of elements to be processed in the list, so N == maxIdx.

$$T(N) = 2+4+2+3+2+1+1+\sum_{i=0}^{N-2}\left(2+2+2+3+2+1\right)+1+2+3+2+1$$

$$= \sum_{i=0}^{N-2}(12)+24$$

$$= 12(N-1)+24$$

$$= 12N+12$$

2. [35 points] For each part, determine the simplest possible function g(n) such that the given function is $\Theta(g)$. No justification is necessary.

a) $a(n) = 3 + 14n + 47n^2$ n^2

(by Theorem 13)

b) $b(n) = 14n^2 + 3n\log n$ n^2

(by Theorem 13)

Hint: the last three take a little analysis.

c) $c(n) = n^{0.9} + \log n$ $n^{0.9}$

Just apply Theorem 8 (guessing one way or the other):

$$\lim_{n\to\infty}\frac{n^{0.9}+\log n}{n^{0.9}} = \lim_{n\to\infty}\left(1+\frac{\log n}{n^{0.9}}\right) = 1+\lim_{n\to\infty}\frac{1/n\ln 2}{0.9n^{-0.1}} = 1+\lim_{n\to\infty}\frac{\ln 2}{0.9n^{0.9}} = 1+0=1$$

d) $d(n) = 3n^2\log n + n^3$ n^3

$$\lim_{n\to\infty}\frac{3n^2\log n+n^3}{n^3} = \lim_{n\to\infty}\left(\frac{3\log n}{n}+1\right) = 1+\lim_{n\to\infty}\frac{3/n\ln 2}{1} = 1+\lim_{n\to\infty}\frac{3}{n\ln 2} = 1+0=1$$

e) $e(n) = \dfrac{1+n}{5n}$ **1**

$$\lim_{n\to\infty}\frac{\frac{1+n}{5n}}{1} = \lim_{n\to\infty}\frac{1+n}{5n} = \lim_{n\to\infty}\left(\frac{1}{5n}+\frac{1}{5}\right) = \frac{1}{5}$$

3. Suppose that $f(n)$ and $g(n)$ are non-negative functions and that $f(n)$ is $O(n^2)$ and $g(n)$ is $O(n)$.

a) [10 points] Define a function $S(n)$ by $S(n) = f(n) + g(n)$. What's the "smallest" function $M(n)$ such that $S(n)$ is guaranteed to be $O(M(n))$? Justify your conclusion with a proof! (If you want to make your argument with limits, you may assume that any related limits do exist.)

It should seem reasonable that this would be n^2 since that's the larger of the two bounds; you can check that by applying the limit theorem:

$$\lim_{n\to\infty}\frac{S(n)}{n^2} = \lim_{n\to\infty}\frac{f(n)+g(n)}{n^2} = \lim_{n\to\infty}\left(\frac{f(n)}{n^2} + \frac{1}{n}\frac{g(n)}{n}\right) = \lim_{n\to\infty}\frac{f(n)}{n^2} + \lim_{n\to\infty}\frac{1}{n} \times \lim_{n\to\infty}\frac{g(n)}{n}$$
$$= L_1 + 0 \times L_2 = L_1$$

(Taking L_1 and L_2 to be the respective limits from the given big-O facts. Note that we are not told that the facts are strict bounds, so it's entirely possible that L_1 and/or L_2 is zero. However, the conclusion still follows even in that case.)

Alternatively, you can apply the definition of big-O. From the given facts, we know that:

$$\exists C_1 > 0 \text{ and } N_1 > 0 \text{ such that for all } n > N_1, f(n) \le C_1 n^2$$
$$\exists C_2 > 0 \text{ and } N_2 > 0 \text{ such that for all } n > N_2, g(n) \le C_2 n$$

Therefore, if we let $N = \max(N_1, N_2)$ and $C = \max(C_1, C_2)$ then it follows from above that

$$\text{for all } n > N, f(n) + g(n) \le Cn^2 + Cn = C(n^2 + n)$$

Now from the theorems in the notes, $n^2 + n$ is $O\left(n^2\right)$ and therefore by transitivity, $S(n)$ is $O\left(n^2\right)$.

b) [10 points] Define a function $P(n)$ by $P(n) = f(n) \cdot g(n)$. What's the "smallest" function $Z(n)$ such that $P(n)$ is guaranteed to be $O(Z(n))$? Justify your conclusion with a proof! (If you want to make your argument with limits, you may assume that any related limits do exist.)

It should seem reasonable that this would be n^3 since that's the product of the two bounds; you can check that by applying the limit theorem:

$$\lim_{n\to\infty}\frac{S(n)}{n^3} = \lim_{n\to\infty}\frac{f(n)\cdot g(n)}{n^3} = \lim_{n\to\infty}\left(\frac{f(n)}{n^2}\frac{g(n)}{n}\right) = \lim_{n\to\infty}\frac{f(n)}{n^2} \times \lim_{n\to\infty}\frac{g(n)}{n}$$
$$= L_1 \times L_2$$

Alternatively, you can apply the definition of big-O. From the given facts, we know that:

$$\exists C_1 > 0 \text{ and } N_1 > 0 \text{ such that for all } n > N_1, f(n) \le C_1 n^2$$
$$\exists C_2 > 0 \text{ and } N_2 > 0 \text{ such that for all } n > N_2, g(n) \le C_2 n$$

Therefore, if we let $N = \max(N_1, N_2)$ and $C = C_1 \cdot C_2$ then it follows from above that

$$\text{for all } n > N, f(n) \cdot g(n) \le Cn^2 \cdot n = Cn^3$$

Therefore, $P(n)$ is $O\left(n^3\right)$.

c) [5 points] What can you conclude about the relationship between $f(n)$ and $g(n)$? For example, can you conclude that $f(n)$ is $O(g(n))$? or that $g(n)$ is $O(f(n))$? or some other relationship? Explain your conclusion.

Actually, we can't conclude anything. The given facts leave the possibility that $f(n)$ and $g(n)$ could be any two functions that are loosely (or tightly) bounded by n^2 and n, respectively. For example, it's possible that

$$f(n) = n^2 \text{ and } g(n) = n$$

or that

$$f(n) = n \text{ and } g(n) = n$$

or that

$$f(n) = 1 \text{ and } g(n) = n$$

None of those cases would yield the same conclusions about how the two functions are related.

4. [20 points] Suppose that executing an algorithm on input of size N requires executing T(N) = N log N + 16N instructions. How long would it take to execute this algorithm on hardware capable of carrying out 2^{24} instructions per second if $N = 2^{28}$? (Give your answer in hours, minutes and seconds, to the nearest second.)

The number of operations required would be given by

$$T(2^{28}) = 2^{28}\log 2^{28} + 16\cdot 2^{28} = 2^{28}\cdot 28 + 16\cdot 2^{28} = 44\cdot 2^{28}$$

So, the number of seconds required would be given by

$$\frac{T(2^{28})}{2^{24}} = \frac{44\cdot 2^{28}}{2^{24}} = 44\cdot 2^4 = 704$$

This is 11 minutes and 44 seconds.