CSE 15 Homework 7
Solutions (Total 20 points)

Type your answers in a text file and submit it in CatCourses.
You can also write your answers on papers and scan them into image files for submission.

Section 3.2

2. **(1pt each sub-problem)**

b) Yes, since $x^2 + 1000 \leq x^2 + x^2 = 2x^2$ for all $x > \sqrt{1000}$. The witnesses are $C = 2$ and $k = \sqrt{1000}$.

d) No. If there were a constant C such that $x^4/2 \leq Cx^2$ for sufficiently large x, then we would have $C \geq x^2/2$. This is clearly impossible for a constant to satisfy.

f) Yes, since $\lfloor x \rfloor \lceil x \rceil \leq x(x+1) \leq x \cdot 2x = 2x^2$ for all $x > 1$. The witnesses are $C = 2$ and $k = 1$.

6. **(1pt)**

We can use the following inequalities, valid for all $x > 1$ (note that making the denominator of a fraction smaller makes the fraction larger).

$$\frac{x^3 + 2x}{2x+1} \leq \frac{x^3 + 2x^3}{2x} = \frac{3}{2}x^2$$

This proves the desired statement, with witnesses $k = 1$ and $C = 3/2$.

8. **(1pt each sub-problem)**

a) Since $x^3 \log x$ is not $O(x^3)$ (because the $\log x$ factor grows without bound as x increases), $n = 3$ is too small. On the other hand, certainly $\log x$ grows more slowly than x, so $2x^2 + x^3 \log x \leq 2x^4 + x^4 = 3x^4$. Therefore $n = 4$ is the answer, with $C = 3$ and $k = 0$.

d) This is similar to the previous part, but this time $n = -1$ will do, since for large x, $f(x) \approx 1/x$. Formally we can write $f(x) \leq 6x^3/x^3 = 6$ for all $x > 1$, so witnesses are $C = 6$ and $k = 1$.

12. **(1pt)**

We showed that $x \log x$ is $O(x^2)$ in Exercise 2c. To show that x^2 is not $O(x \log x)$ it is enough to show that $x^2/(x \log x)$ is unbounded. This is the same as showing that $x/\log x$ is unbounded. First let us note that $\log x < \sqrt{x}$ for all $x > 16$. This can be seen by looking at the graphs of these functions, or by calculus. Therefore the fraction $x/\log x$ is greater than $x/\sqrt{x} = \sqrt{x}$ for all $x > 16$, and this clearly is not bounded.

18. **(1pt)**

$$1^k + 2^k + \cdots + n^k \leq n^k + n^k + \cdots + n^k = n \cdot n^k = n^{k+1}$$

20. **(1pt)**

They both are. For the first we have $\log(n+1) < \log(2n) = \log n + \log 2 < 2\log n$ for $n > 2$. For the second one we have $\log(n^2+1) < \log(2n^2) = 2\log n + \log 2 < 3\log n$ for $n > 2$.

22. **(1pt)**

The ordering is straightforward when we remember that exponential functions grow faster than polynomial functions, that factorial functions grow faster still, and that logarithmic functions grow very slowly. The order is $(\log n)^3$, $\sqrt{n}\log n$, $n^{99}+n^{98}$, n^{100}, 1.5^n, 10^n, $(n!)^2$.

Section 3.3

2. **(1pt)**

The statement $t := t + i + j$ is executed n^2 times, so the number of operations is $O(n^2)$. (Specifically, $2n^2$ additions are used, not counting any arithmetic needed for bookkeeping in the loops.)

4. **(1pt)**

The value of i keeps doubling, so the loop terminates after k iterations as soon as $2^k > n$. The value of k that makes this happen is $O(\log n)$, because $2^{\log n} = n$. Within the loop there are two additions or multiplications, so the answer to the question is $O(\log n)$.

8. **(1pt)**

If we successively square k times, then we have computed x^{2^k}. Thus we can compute x^{2^k} with only k multiplications, rather than the $2^k - 1$ multiplications that the naive algorithm would require, so this method is much more efficient.

18. **(0.5 pt each sub-problem)**

We are asked to compute $(2n^2 + 2^n)\cdot 10^{-9}$ for each of these values of n. When appropriate, we change the units from seconds to some larger unit of time.

a) 1.224×10^{-6} seconds b) approximately 1.05×10^{-3} seconds

c) approximately 1.13×10^{6} seconds, which is about 13 days (nonstop)

d) approximately 1.27×10^{21} seconds, which is about 4×10^{13} years (nonstop)

20. **(0.5 pt each sub-problem)**

In each case we want to compare the function evaluated at $2n$ to the function evaluated at n. The most desirable form of the comparison (subtraction or division) will vary.

a) Notice that

$$\log\log 2n - \log\log n = \log \frac{\log 2 + \log n}{\log n} = \log \frac{1 + \log n}{\log n}.$$

If n is large, the fraction in this expression is approximately equal to 1, and therefore the expression is approximately equal to 0. In other words, hardly any extra time is required. For example, in going from $n = 1024$ to $n = 2048$, the number of extra milliseconds is $\log 11/10 \approx 0.14$.

c) This time it makes more sense to use a ratio comparison, rather than a difference comparison. Because $100(2n)/(100n) = 2$, we conclude that twice as much time is needed for the larger problem.

e) Because $(2n)^2/n^2 = 4$, we see that four times as much time is required for the larger problem.

g) The relevant ratio is $2^{2n}/2^n$, which equals 2^n. If n is large, then this is a huge number. For example, in going from $n = 10$ to $n = 20$, the number of milliseconds increases over 1000-fold.

22. **(1pt each sub-problem)**

a) The number of comparisons does not depend on the values of a_1 through a_n. Exactly $2n - 1$ comparisons are used, as was determined in Example 1. In other words, the best case performance is $O(n)$.
b) In the best case $x = a_1$. We saw in Example 4 that three comparisons are used in that case. The best case performance, then, is $O(1)$.
c) It is hard to give an exact answer, since it depends on the binary representation of the number n, among other things. In any case, the best case performance is really not much different from the worst case performance, namely $O(\log n)$, since the list is essentially cut in half at each iteration, and the algorithm does not stop until the list has only one element left in it.