This homework has a total of 4 problems on 4 pages. Solutions should be submitted to GradeScope before 3:00pm on Wednesday, September 6, 2017.

It will be marked out of 20, you can earn up to 21 = 1 + 5 + 8 + 3 + 4 points.

**Late policy:** each student can use up to two late days on each homework, for a total of four late days across all four homeworks.

If you choose not to submit a typed write-up, please write neat and legibly.

Collaboration is allowed/encouraged on problems, however each student must independently complete their own write-up, and list all collaborators. No credit will be given to solutions obtained verbatim from the Internet or other sources.

0. [1 point, only if all parts are completed]

(a) Submit either a scan or photo of your homework in a single pdf file on GradeScope.

(b) Words on the scan are clearly readable.

(c) The answer to each question except question 0 should start on a new page.

1. (5 points) Big-O Notation

(a) (1 point) Formally prove that Big O is transitive by relation. That is, if $f(n) \leq O(g(n))$ and $g(n) \leq O(h(n))$, then, $f(n) \leq O(h(n))$.

**SOLUTION:**

By definition of big-O, there exists constants $C_1$ and $C_2$ such that for all $n$ we have

\[ f(n) \leq C_1 g(n) \]

and

\[ g(n) \leq C_2 h(n). \]

Multiplying the second one by $C_1$ then gives:

\[ C_1 g(n) \leq C_1 \cdot C_2 h(n), \]

which together with the first condition gives:

\[ f(n) \leq C_1 g(n) \leq C_1 \cdot C_2 h(n), \]
(b) (1 point) Find the complexity of the following set loops, where $n$ is given as input:

```c
i <-- n;
while(i > 1) {
    j = i;  //%%% CAUTION: this DOES NOT START AT 0
    while (j < n) {
        k <-- 0;
        while (k < n) {
            k = k + 2;
        }
        j <-- j * 2;
    }
    i <-- i / 2;
}
```

Express your answer using the $\Theta(\cdot)$ notation.

**SOLUTION:**
The innermost lopp takes $\Theta(n)$ whenever it is called.
The outer most loop takes $\Theta(\log n)$ steps, and between steps $\log n/2$ and $\log n$ (a total of $\Theta(\log n)$ of steps), we have $i < n^{1/2}$.

In each such step, $j$ gets doubled $\Theta(\log n)$ times before it reaches $n$. So we get a total of $\Theta(\log^2 n)$ iterations, each costing $\Theta(n)$, for a total of $\Theta(n \log^2 n)$.

(c) (1 point) Prove or disprove: there does not exist a pair of functions $f(n)$ and $g(n)$ such that $f(n) \leq O(g(n))$ and $f(n) \geq \Omega(g(n))$.

**SOLUTION:**
False, $f(n) = g(n) = n$ are a pair of such functions.

(d) (1 point) Prove or disprove: $n^2 \log^{10} n \leq O(n^{2.1})$.

**SOLUTION:**
True. This follows from combining $n^2 \leq O(n^2)$ with $\log^{10} n \leq O(n^{0.1})$.

(e) (1 point) Prove or disprove: $2^{2n} \leq O(2^n)$.

**SOLUTION:**
False, for any constant $C$, once we have $2^n > C$, we get $2^{2n} > C2^n$.

2. (8 points) Master Theorem
The master theorem applies to algorithms with recurrence relations in the form of

$$T(n) = aT(n/b) + O(n^d)$$
for some constants $a > 0$, $b > 1$, and $d \geq 0$

Find the asymptotic (big-O notation) running time of the following algorithms using Master theorem if possible. State the runtime recurrence if it’s not given, and if Master theorem is applicable, explicitly state the parameters $a$, $b$ and $d$. Otherwise, give a quick reason that the recurrence relation is not solvable using Master theorem.

(a) (2 points) An algorithm with the run-time recurrence:

$$T(n) = 3T(n/4) + O(n)$$

**SOLUTION:**

$a = 3$, $b = 4$, $d = 1$.

$log_b a = log_4 3 \approx .79 < 1$, so $O(n)$.

(b) (2 points) An algorithm with the run-time recurrence:

$$T(n) = 8T(n/4) + O(n^{1.5})$$

**SOLUTION:**

$$T(n) = 3T(n/4) + O(1)$$

$a = 3$, $b = 4$, $d = 0$.

$log_b a = log_4 3 \approx .79 > 0$, so total runtime is $O(n^{.79})$.

(c) (2 points) An algorithm solves problems by diving a problem of size $n$ into 3 sub-problems of one-fourth the size and recursively solves the smaller sub-problems. It takes constant time to combine the solutions of the sub-problems.

**SOLUTION:**

$$T(n) = 3T(n/4) + O(1)$$

$a = 3$, $b = 4$, $d = 0$.

$log_b a = log_4 3 \approx .79 > 0$, so total runtime is $O(n^{.79})$.

**GRADING:**

There was a typo in the August 23 notes (now fixed) that said Master Theorem only applied for $d \geq 1$. Anyone who then claimed that Master Theorem did not apply for this problem because $d < 1$ should have still received full points.
(d) (2 points) An algorithm solves problems by diving a problem of size $n$ into $2^n$ sub-problems of half the size and recursively solves the smaller sub-problems. It takes linear time to combine the solutions of the sub-problems.

**SOLUTION:**

$$T(n) = 2^n T(n/2) + O(n).$$

Not solvable by Master theorem since $a = 2^n$ is not a constant.

3. (3 points) Divide and conquer

After learning about the Stooge sort ([https://en.wikipedia.org/wiki/Stooge_sort](https://en.wikipedia.org/wiki/Stooge_sort)), Buzz would like to improve its performance. This led to the Buzzsort, with pseudocode as follows:

(a) If the sequence length is at most 4, sort it using bubble sort.

(b) Else:
   
   i. Divide the list into 5 pieces evenly, by scanning the entire list.
   
   ii. (recursively) sort the first $3/5$ of the list.
   
   iii. (recursively) sort the last $3/5$ of the list.
   
   iv. (recursively) sort the first $3/5$ of the list.

For example, on the input sequence

$$1, 5, 3, 2, 4$$

The first recursive sort produces

$$1, 3, 5, 2, 4,$$

the second sort produces

$$1, 3, 2, 4, 5,$$

and the last produces

$$1, 2, 3, 4, 5.$$

(a) (2 points) Write down a runtime recurrence for Buzzsort and analyze its asymptotic running time.

**SOLUTION:**

$$T(n) = 3T\left(\frac{3}{5}n\right) + O(n).$$

This fits into the requirements of Master theorem with $a = 3$, $b = \frac{5}{3}$, and $d = 1$. $\log_{\frac{5}{3}} 3 \approx 2.15 > 1$, so the running time is $O(n^{2.16})$. 

4
(b) (1 point) Give an example sequence on 5 or 10 integers where Buzzsort does not terminate with the correct answer.

**SOLUTION:**

on the input sequence $5, 4, 3, 2, 1$.

The first recursive sort produces $3, 4, 5, 2, 1$,

the second sort produces $3, 4, 1, 2, 5$,

and the third produces $1, 3, 4, 2, 5$,

which is not sorted.

4. (4 points) Fast multiplication and convolution.

We show several additional applications of fast multiplications of integers. A degree $d$ polynomial is the function $p(x) = a_0 + a_1 x + a_2 x^2 + \ldots a_d x^d$.

These objects multiply just like integers, except without carries. That is if we multiply degree $d$ polynomials $p$ and $q$ with coefficients $a_0 \ldots a_d$ and $b_0 \ldots b_d$ respectively, the coefficient of $x^i$ in the result is

$$\sum_{\max\{0,i-d\} \leq j \leq \min\{i,d\}} a_j \cdot b_{i-j}.$$ 

(a) (1 point) Show that if $p(x)$ and $q(x)$ are degree $n$ polynomials with integer coefficients in the range $[0, n]$, all coefficients in the product $p(x) \cdot q(x)$ are integers in the range $[0, O(n^3)]$.

**SOLUTION:**

Each of the $a_j \cdot b_{i-j}$ term is at most $n^2$, $n$ of them gives $O(n^3)$.

(b) (2 points) Show using an extension of Karatsuba’s algorithm that two degree $n$ polynomials can still be multiplied in $O(n^{1.6})$ time or better. You may assume that integer arithmetics involving $\text{poly}(n)$ sized numbers take $O(1)$ time.

**SOLUTION:**

We modify Karatsuba’s algorithm by passing around polynomials. The operations of addition / subtraction still works in linear time, and the key identity becomes:

$$[p_1(x) \times x^k + p_2(x)] [q_1(x) \times x^k + q_2(x)] = (x^{2k} - x^k) p_1(x) q_1(x) - (x^k - 1) p_2(x) q_2(x) + x^k [p_1(x) + p_2(x)] [q_1(x) + q_2(x)].$$

So the same divide-and-conquer scheme still works.
(c) (1 points) The ‘shifted dot products’ of two sequences $y_0 \ldots y_n$ and $z_0 \ldots z_n$ for each shift $s$ is given by

$$\sum_{i=0}^{n-s} y_i z_{i+s}.$$ 

Show (via equations) that the $s$-shifted dot product is precisely the coefficient of $x^{n-s}$ in the product of the polynomials with coefficients

$$a_0 = y_0, a_1 = y_1, \ldots, a_n = y_n.$$ 

and

$$b_0 = z_n, b_1 = z_{n-1}, \ldots, b_n = z_0.$$ 

Note that the second polynomial takes the coefficients in the reverse order.

**Aside:** these values are quite useful in performing approximate string matching.

**SOLUTION:**

Plugging in $a_i = y_i$ and $b_i = z_{n-i}$ into the coefficient for $n - s$ in their polynomial product gives:

$$\sum_{0 \leq i \leq n-s} a_i b_{n-s-i} = \sum_{0 \leq i \leq n-s} y_i z_{n-(n-s-i)} = \sum_{0 \leq i \leq n-s} y_i z_{s+i}.$$ 

The last term is exactly the $s$-shifted dot product between $y$ and $z$. 