Name:

CSE 5211 Analysis of Algorithms
Spring 2018 Practicd Midterm Key

Score

1. (5 pts) Simplify the summation:

$$\sum_{0 \le k < n} [3^k + 3k - 2]$$

Answer: The sum is

$$\begin{aligned}\sum_{0 \le k < n} [3^k + 3k - 2] &= \sum_{0 \le k < n} 3^k + \sum_{0 \le k < n} 3k - \sum_{0 \le k < n} 2 \\ &= \left[\frac{3^n - 1}{2}\right] + 3\binom{n}{2} + 2n \\ &= \left[\frac{3^n - 1}{2}\right] + \frac{3n^2 + n}{2}\end{aligned}$$

Score

2. (10 pts) Solve the recurrence $T(n) = T(n-1) + 2n + 2$ with initial condition $T(0) = 1$.

Answer: The equation unrolls:

$$\begin{aligned}T(n) &= T(n-1) + 2n + 2 \\ &= [T(n-2) + 2(n-1) + 2] + 2n + 2 \\ &= [T(n-3) + 2(n-2) + 2] + 2(n-1) + 2 + 2n + 2 \\ &\vdots \\ &T(0) + 2[1 + 2 + 3 + \cdots + n] + 2(n) \\ &= 1 + n(n+1) + 2n \\ &= n^2 + 3n + 1\end{aligned}$$

Score

3. (5 pts) Find the generating function for the sequence $\langle 1, 3, 5, 7, \ldots \rangle$ of positive odd integers.

Answer: The generating function is

$$\begin{aligned} G(z) &= 1 + 3z + 5z^2 + 7z^3 + \cdots \\ &= \sum_{0 \le k} (2k+1) z^k \\ &= \sum_{0 \le k} 2kz^k + \sum_{0 \le k} z^k \\ &= 2z \sum_{0 \le k} (kz^{k-1}) + \frac{1}{1-z} \\ &= \frac{2z}{(1-z)^2} + \frac{1}{1-z} \\ &= \frac{z+1}{(1-z)^2} \end{aligned}$$

Score

4. (5 pts) What sequence is associated with the generating function

$$G(z) = \frac{2}{1-z^2}?$$

(Note this function is not $1/(1-z)^2$)

Answer: Since

$$\frac{1}{1-z} = \sum_{0 \le k} z^k$$

by substitution you get

$$\begin{aligned} G(z) &= \frac{2}{1-z^2} \\ &= \sum_{0 \le k} z^{2k} \\ & 1 + z^2 + z^4 + \cdots \end{aligned}$$

And the sequence is $\langle 1, 0, 1, 0, 1, 0, 1, \ldots \rangle$.

(`True` or `False`) Explain your answer.

Score

1. (5 pts) If $f(n) = O(n^2)$ then $f(n) = O(n)$.

   Answer: This is `False`. Here is a counterexample: Let

   $$f(n) = n^2 = O(n^2)$$

   But, there are no constants $c > 0$ and $m \geq 0$ such that

   $$f(n) = n^2 \leq cn \quad \forall\ n \geq m$$

   which is clearly `False` because it requires $n \leq c$ for all $n \geq m$.

Score

2. (5 pts) If $f(n)$ is $O(3^{\lg n})$, then $f(n)$ is $O(n^2)$.

   Answer: This is `True`. Here's the trick: $a^{\lg(b)} = b^{\lg(a)}$. You can see the identity by take the log of both sides and notice the equality. Then since $\lg(3) \approx 1.58496250072 < 2$ If $f(n)$ is $O(3^{\lg n})$, then $f(n)$ is $O(n^2)$.

Score

3. (5 pts) The function $f(n) = \lg(n!) = O(n \lg n)$.

   Answer: This is `True`. Notice

$$\begin{aligned}
\lg(n!) &= \lg(n(n-1)(n-2)\cdots 2 \cdot 1) && \text{(definition of factorial.)} \\
&= \sum_{1 \leq k \leq n} \lg(k) && \text{(log of product rule.)} \\
&\leq \sum_{1 \leq k \leq n} \lg(n) && \text{(log of product rule.)} \\
&= n \lg(n) && \text{(replacing } \lg k \text{ by the larger } \lg n.\text{)}
\end{aligned}$$

The code below is a (highly) edited implementation of Gaussian elimination Answer the questions that follow.

The code implements pivoting: rows are swapped to help minimize floating point errors. the macro

$$\text{mat\_elem(a, y, x, n) (a + ((y) * (n) + (x)))}$$

indexes into row x, column y of 2-dimensional array a, where n is number of columns in a row (the row length). It is a constant time operation.

```
1  #define mat_elem(a, y, x, n) (a + ((y) * (n) + (x)))
2  void swap_row(double *a, double *b, int r1, int r2, int n)
3  {
4    double tmp, *p1, *p2;
5    int i;
6    if (r1 == r2) return;       // no need to swap a row with itself.
7    for (i = 0; i < n; i++) { // for every column
8      p1 = mat_elem(a, r1, i, n);  // get value in row r1, column i
9      p2 = mat_elem(a, r2, i, n);  // get value in row r2, column i
10     tmp = *p1, *p1 = *p2, *p2 = tmp; // swap values
11   }
12   tmp = b[r1], b[r1] = b[r2], b[r2] = tmp;   // swap where the row lie
13 }
```

Score ☐

1. (5 pts) What is the worst-case time complexity of swap_row?

   Answer: It is $O(n)$: picking the values out of the array is constant time. Swapping the pointers is too. The loop executed n times.

```
1
2  int i, j, col, row, max_row,diag;
3  double max, tmp;
4
5  for (diag = 0; diag < n; diag++) {
6    max_row = diag, max = A(diag, diag);
7
8    for (row = diag + 1; row < n; row++) {
9      if ((tmp = fabs(A(row, diag))) > max) {max_row = row, max = tmp;}
10   }
11   swap_row(a, b, diag, max_row, n);
12
13   for (row = diag + 1; row < n; row++) {
14     tmp = A(row, diag) / A(diag, diag);
15     for (col = diag+1; col < n; col++) {A(row, col) -= tmp * A(diag, col);}
16     A(row, diag) = 0;
17     b[row] -= tmp * b[diag];
18   }
19 }
20 for (row = n - 1; row >= 0; row--) {
21   tmp = b[row];
22   for (j = n - 1; j > row; j--) { tmp -= x[j] * A(row, j);}
23   x[row] = tmp / A(row, row);
24 }
```

Score ☐

2. (5 pts) What is big-O complexity of line 22? (Use summation notation in expressing your answer.)

   Answer: Let r stand for the row variable. For counting let j increase. The result is

$$\sum_{r<j<n} 1 = n - r - 1$$

Score

3. (5 pts) What is the big-O complexity of the for loop from line 20 to line 24? (Use summation notation in expressing your answer.)

   Answer: Using the previous answer, This sum is

$$\sum_{0 \le r < n} (n - r - 1) = (n-1) + (n-2) + \cdots + 0 = \binom{n}{2}$$

Score

4. (5 pts) What is the big-O complexity of the for loop in line 15

   Answer: Let d and c stand for the variables diag and col

$$\sum_{d < c < n} 1 = n - d + 1$$

Score

5. (5 pts) What is the big-O complexity of the for loop in line 13 to line 18?

   Answer: The complexity is modeled by the sum

$$\begin{aligned}\sum_{d < r < n} [1 + \sum_{d < c < n} 1] &= \sum_{d < r < n} [1 + (n - d - 1)] \\ &= (n - d - 1)(n - d) \\ &= \binom{n-d}{2}\end{aligned}$$

Score

6. (5 pts) What is the worst-case big-O complexity of the for loop from line 5 to line 19?

   Answer: swap_row is called every time through the loop. In the worst-case, a row swap is required with a cost that is $O(n)$. The complexity is modeled by the sum

$$\sum_{0 \le d < n} [1 + (n - d - 1) + n + \binom{n-d}{2}] = \sum_{0 \le d < n} [(2n - d) + \binom{n-d}{2}]$$

   where the first 1 is for setting max_row. The $(n - d - 1)$ term is for testing if a row below the diagonal is a better pivot row (lines 8 to 10). The next n is the worst case swap. And, the binomial coefficient term is from the previous problem.

   Simplifying the sum yields:

$$\begin{aligned}\sum_{0 \le d < n} [1 + (n - d - 1) + n + \binom{n-d}{2}] &= \sum_{0 \le d < n} [(2n - d) + \binom{n-d}{2}] \\ &= [(2n) + (2n-1) + \cdots + (n+1)] + [\binom{n}{2} + \binom{n-1}{2} + \cdots + \binom{1}{2}] \\ &= [\frac{(3n+1)(n)}{2}) + \binom{n}{3} \\ &= O(n^3)\end{aligned}$$

Score

7. (5 pts) What is the best-case big-O complexity of the for loop from line 5 to line 19?

   Answer: In the best-case the call to swap_row is $O(1)$. The complexity is modeled by the sum

$$\begin{aligned}\sum_{0\le d<n}[1+(n-d-1)+\binom{n-d}{2}] &= \sum_{0\le d<n}[(n-d)+\binom{n-d}{2}] \\ &= [(n)+(n-1)+\cdots+(1)]+[\binom{n}{2}+\binom{n-1}{2}+\cdots+\binom{1}{2}] \\ &= [\binom{n+1}{2}+\binom{n}{3}] \\ &= O(n^3)\end{aligned}$$

8. [ts5 Which, if any for the following is `True`: This implementation of Gaussian elimination is: $O(n^3)$, $\Omega(n^3)$, $\Theta(n^3)$?

   Answer: They are all `True`.

1 The Little Haskells (a play on The Little Rascals)

Score

1. (5 pts) The list concatenation function is defined in the Haskell Prelude by

```
1 (++) :: [a] -> [a] -> [a]
2 (++) []     ys = ys
3 (++) (x:xs) ys = x : (xs ++ ys)
```

   (a) What initial cost would you assign to line 2?
   Answer: The time complexity of 2 is $O(1)$.

   (b) What recurrence equation describes line 3? You may assume the construction operator x:xs take constant $O(1)$ time.
   Answer: Let x:xs $= n$. The time complexity is modeled by the recurrence

$$T(n) = 1 + T(n-1)$$

   which has solution $T(n) = n$.

Score

2. (5 pts) The last function function is defined in the Haskell Prelude by

```
1 last [x]    = x
2 last (_:xs) = last xs
3 last []     = errorEmptyList "last"
```

   (a) What initial condition and recursion describes the time complexity of the code?
   Answer: For a single element list, the complexity is $T(1) = 1$, For a longer list, the recursion is $T(n) = 1 + T(n-1)$, where the 1 is cost of dropping the first element.

Score

3. (10 pts) Consider the following algorithm that returns the minimum and maximum values from an list of of values that can be ordered. What recurrence equation and initial conditions expresses the algorithm's time complexity. (Do not solve the equation.)

```
1  maxmin :: (Ord a) => [a] -> (a,a)
2  maxmin [] = (error "maxmin of empty list", undefined)
3  maxmin [x] = (x, x)
4  maxmin  [x,y]
5      | (x >= y)  = (x ,y)
6      | otherwise = (y, x)
7  maxmin (x:y:xs)
8          | (max > maxTail)  && (min < minTail) = (max, min)
9          | (max > maxTail)  && (min > minTail) = (max, minTail)
10         | (max < maxTail)  && (min < minTail) = (maxTail, min)
11         | (max < maxTail)  && (min > minTail) = (maxTail, minTail)
12       where first = maxmin [x, y]
13             max     = fst $ maxmin [x, y]
14             min     = snd $ maxmin [x, y]
15             residue = maxmin xs
16             maxTail = fst residue
17             minTail = snd residue
```

   Answer: Initial conditions: $T(1) = 1$, $T(2) = 2$ seem reasonable. Given a list x:y:ss of length n there is: One call to the function with 2 arguments. One call to the function with $n-2$ arguments. Plus several other constant time assignments and tests. The recurrence is

$$T(n) = 2 + T(n-2) + c \quad \text{or more simple} \quad T(n) = T(n-2) + C$$

Total Points: 95