**Answers to Quiz 5**

**Score Distribution** Average \( \approx 11.72 \). Median = 11.5. For the histogram, fractional scores were rounded down.

<table>
<thead>
<tr>
<th>Score</th>
<th>Count</th>
<th>Score</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>14-15</td>
<td>14</td>
<td>12-13</td>
<td>13</td>
</tr>
<tr>
<td>12-13</td>
<td>89</td>
<td>11-11</td>
<td>11</td>
</tr>
<tr>
<td>11-11</td>
<td>101</td>
<td>10-10</td>
<td>10</td>
</tr>
<tr>
<td>10-10</td>
<td>101</td>
<td>9-9</td>
<td>9</td>
</tr>
<tr>
<td>9-9</td>
<td>99</td>
<td>8-8</td>
<td>8</td>
</tr>
<tr>
<td>8-8</td>
<td>88</td>
<td>7-7</td>
<td>7</td>
</tr>
<tr>
<td>7-7</td>
<td>77</td>
<td>6-6</td>
<td>6</td>
</tr>
<tr>
<td>6-6</td>
<td>66</td>
<td>5-5</td>
<td>5</td>
</tr>
<tr>
<td>5-5</td>
<td>55</td>
<td>4-4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Problem 1 (9 points)** The length of a path is the number of edges in the path. A directed graph \( G = (\{1, \ldots, n\}, E) \) is forward-looking when: (i) every vertex except \( n \) has at least one edge leaving it and (ii) \( i < j \) for each \((i, j) \in E\). For example:

![Diagram of a directed graph](image)

Below, \( G \) will always be forward-looking. We are interested in the problem:

**Given:** \( G = (\{1, \ldots, n\}, E) \).

**Find:** the length of longest path from vertex 1 to \( n \).

**a. (2 points)** Give an example \( G \) on which the following greedy algorithm returns the wrong answer for the length of the longest path from 1 to \( n \). Explain your example. (Hint: Tinker with the example.)

**function** badFindLongest(\( G \))

\[ i \leftarrow 1; \ \ell \leftarrow 0; \ n \leftarrow \text{the number of vertices in } G \]

while \( i \neq n \) do

choose \((i, j) \in E \) with \( j \) is as small as possible

\[ i \leftarrow j; \ \ell \leftarrow \ell + 1 \]

return \( \ell \)

**b. (2 points)** Suppose \( i_1 < i_2 < \cdots < i_k \) (where \( i_k = n \)) are the vertices in a longest path in \( G \) from \( i_1 \) to \( n \). Explain why \( i_2, i_3, \ldots, i_k \) are the vertices in a longest path from \( i_2 \) to \( n \). (Hint: The forward-looking property is important here.)

**Problem 2 (7 points)** In the game Crawl4Cash there is an \( n \times n \) grid of squares with co-ordinates \((x, y)\) for \( 1 \leq x, y \leq n \). You have a pawn that begins at square (1,1) and moves according to the rules:

**Rule 1.** A pawn on \((x, y)\) with \( x, y < n \), must move to one of \((x+1, y)\), \((x, y+1)\), and \((x+1, y+1)\).

**Rule 2.** When the pawn is on square \((x, y)\) with \( x = n \) or \( y = n \), the game is over.

When a pawn moves on to \((x, y)\), you receive \( p(x, y) \) many dollars. (You receive no dollars from starting at \((1,1)\).) We want a dynamic programming algorithm that computes the maximum possible winnings for this game.

**a. (2 points)** What are the subproblems to solve?

**b. (2 points)** What is the recursive equation for this problem?

**c. (3 points)** Provide pseudo-code for the algorithm. What is the run-time of your algorithm? Justify this run-time.

**An answer for a.** C\([x, y]\) = the maximum possible winnings if we start on square \((x, y)\).

**Answer for b.** For \( x, y < n \), \( C[x, y] = \max \left\{ p(k, \ell) + C[k, \ell] \right\} \) for \((k, \ell) = (x + 1, y), (x, y + 1), (x + 1, y + 1)\).

**An answer for c.**

**function** c4c\( (p) \)

\[ k \leftarrow 1, 2, \ldots, n \] do

\[ C[k, n] \leftarrow 0; \ C[n, k] \leftarrow 0 \]

for \( x \leftarrow n - 1, n - 2, \ldots, 1 \) do

\[ C[x, y] \leftarrow \max \left\{ \text{as in part (b)} \right\} \]

return \( C[1, 1] \)

The max is constant time and the algorithm fills up each entry in a \( n \times n \) array. So the run time is \( O(n^2) \).

**Problem 3 (4 points)** Express the following as a linear programming problem, i.e., what are the variables, what is the objective function, what are the constraints? No need to solve the LP problem.

A potter makes cups and plates. It takes her 3 minutes to make a cup and 1.5 minutes to make a plate. Each cup uses 0.75 pounds of clay and each plate uses 1 pound of clay. Each day she has 10 hours (600 minutes) available for making the cups and plates and has a supply of 250 pounds of clay. She makes a profit of $2 on each cup and $1.50 on each plate.

**Answer for b.**

\[ x \leftarrow n - 1, n - 2, \ldots, 1 \] do

\[ C[x, y] \leftarrow \max \left\{ \text{as in part (b)} \right\} \]

return \( C[1, 1] \)

The max is constant time and the algorithm fills up each entry in a \( n \times n \) array. So the run time is \( O(n^2) \).
How many cups and how many plates should she make per day in order to maximize her profit?

**Answer.** Variables: \( c = \) number of cups made per day; \( p = \) number of plates made per day. **Objective function:** Maximize \( 2c + 1.5p \). **Constraints:**

\[
0.75c + p \leq 250; 3c + 1.5p \leq 600; \text{ and } c, p \geq 0.
\]

### Some Reference Math Facts

a. \( a^m \cdot a^n = a^{m+n} \).

b. \( a^{m \cdot n} = (a^m)^n = (a^n)^m \).

c. \( a^m \cdot b^n = (a \cdot b)^{m \cdot n} \).

d. \( \log_a a^n = n \).

e. \( a^{\log_a n} = n \).

f. \( c \cdot \log_2 a = \log_2 a^c \).

g. \( \log_2 (a \cdot b) = (\log_2 a) + (\log_2 b) \).

h. \( (\log_a x) / (\log_a b) = \log_b x \).

i. \( a^{\log_a c} = c \log_a a \).

j. \( \sum_{k=1}^{n} k = \frac{1}{2} \cdot n \cdot (n + 1) \).

k. \( \sum_{k=0}^{n} a^k = \frac{a^{n+1} - 1}{a - 1} \).

l. \( \sum_{k=1}^{\infty} 2^{-k} = 1 \).

m. \( \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6} \).

n. \( \sum_{k=1}^{\infty} k \cdot 2^{-k} = 2 \).

o. For \( a > 1 \) and \( b, c > 0 \):

\[
\lim_{n \to \infty} n^b a^{-c n} = 0.
\]

p. The Master Recurrence Theorem: Suppose for all \( n > n_0 \), we have \( T(n) = a \cdot T(n/b) + c \cdot n^k \) for \( a, b \geq 1 \) and \( k \geq 0 \), then:

(i) if \( a < b^k \), then \( T(n) \in \Theta(n^k) \);

(ii) if \( a = b^k \), then \( T(n) \in \Theta(n^k \log n) \); and

(iii) if \( a > b^k \), then \( T(n) \in \Theta(n^k \log^k n) \).

Above, we can take \( n/b \) can be taken as either \( \lfloor n/b \rfloor \pm c \) or \( \lceil n/b \rceil \pm c \) for constant \( c \).