Homework Assignment No. 3 Grading Guide

CSCI 4470/6470 Algorithms, CS@UGA, Spring 2018

Due Thursday March 1, 2018

Four questions with total 110 points.

1. (30 points) Consider the Casino Dice Decoding Problem (ref. pages 35-38 of lecture note 3, NewNote3.pdf). Note that the problem resembles the Assembly line problem. Fill out a dynamic programming table that computes functions $p(S, i, F)$ and $p(S, i, L)$ for sequence of digits $S = 6661234$. You can choose to write a program to or manually calculate for the table. Show how the computation can tell the most probable hidden sequence of dice that may have been used to roll the given 7 digits.
Answer:

The student can code a program or manually to fill out a DP table to compute the function \( p \).

- the table is \( 2 \times 7 \), with either row storing the function values for fair and loaded die respectively. [10 points]
- the table should contain the information for tracing back solutions (hidden dice), the info can be placed within the same table (using arrows or numbers) to indicate which option has been used to compute every cell. [10 points]
- correctness of the probabilities computed for each cell and correctness of trace back result [10 points].

Examples:

![Manually computed](image)

![Computed by a program](image)
2. (20 points) Fill out a dynamic programming table that computes for the 0-1 knapsack problem on the following input data: \( B = 10 \), \( i = 1, 2, 3, 4 \), and

\[
\begin{align*}
  s_1 &= 3, v_1 = 50 \\
  s_2 &= 5, v_2 = 40 \\
  s_3 &= 3, v_3 = 40 \\
  s_4 &= 2, v_4 = 10 \\
\end{align*}
\]

Answer:

- the table is \( 5 \times 11 \), with \( i \)th row corresponding to the first \( i \) items, and the \( j \)th column corresponding to the available size \( j \) [5 points].
- correctness of the probabilities computed for each cell (note: no trace back result was required). [15 points].
3. (40 points) Consider following Article Grading problem. A teacher grades students' articles based on number of letter corrections on an article. The teacher may correct an article with the following four types of editing:

- a letter is kept +5 points
- a letter substitution −2 points
- a letter insertion −3 points
- a letter deletion −3 points

You are required to formulate a dynamic programming solution that can compute the maximum number of points for any article \( X \) based on comparison between \( X \) and the edited version \( Y \). This problem can be thought as a generalization of the LCS problem, where the situation of "a letter is kept" is rewarded 1 point while no penalty applies on a letter "substitution", "insertion", or "deletion".

Let the original article and edited version be \( X \) and \( Y \), respectively. Follow the idea used in the problem analysis for LCS, consider any prefix \( x_1 \ldots x_i \) of \( X \) and any prefix \( y_1 \ldots y_j \) of \( Y \). Assume \( \text{Grade}(i, j) \) to be the maximum score gained when prefix \( x_1 \ldots x_i \) has been edited to \( y_1 \ldots y_j \). There are four possibilities of editing from letter \( x_i \) to letter \( y_j \):

1. \( x_i \) is kept (only if \( x_i = y_j \));
2. \( x_i \) was substituted with \( y_j \);
3. \( y_j \) was inserted after the position \( x_i \);
4. \( x_i \) was deleted (\( x_i \) does show in the edited version anymore);

**Part A.** Based on these 4 cases, you are required to formulate \( \text{Grade}(i, j) \) into a recursive function, i.e., being defined by the same function \( \text{Grade} \) (with smaller parameter values, however).

**Part B.** Write an iterative algorithm to compute for function \( \text{Grade} \); the algorithm needs to include steps to remember the choices of editing.
Answer:

• Part A [20 points]

Define \( \text{Grade}(i, j) \) to be the maximum score gained when prefix \( x_1 \ldots x_i \) has been edited to \( y_1 \ldots y_j \). Then recursively,

\[
\text{Grade}(i, j) = \max \begin{cases} 
\text{Grade}(i - 1, j - 1) + 5 & \text{when } x_i = y_j \\
\text{Grade}(i - 1, j - 1) - 2 & \text{when } x_i \text{ is replaced with } y_j \\
\text{Grade}(i - 1, j) - 3 & \text{when } x_i \text{ is deleted} \\
\text{Grade}(i, j - 1) - 3 & \text{when } y_j \text{ is inserted}
\end{cases}
\]

The base case is \( \text{Grade}(0, 0) = 0 \), other cases are not necessarily but not wrong:

\[
\begin{align*}
\text{Grade}(i, 0) &= -3 \times i, \text{ for all } i = 0, 1, \ldots m \\
\text{Grade}(0, j) &= -3 \times j, \text{ for all } j = 0, 1, \ldots n
\end{align*}
\]

• Part B

The algorithm resembles the one for LCS problem. Details about adding up function values are different. [10 points]

The algorithm needs to have the part to memorize which of the 4 options are used to compute \( \text{Grade}(i, j) \) for each entry \((i, j)\). [10 points]
4. (20 points) Consider a coin change problem with the coin set \( \{C_1, C_2, C_5\} \), where \( C_k \) represents the coin of \( k \) cents. Prove that this problem has the greedy choice property. You may have to prove for the cases of cents \( \geq 5 \), of cents \( 2 \leq \) but \( < 5 \), and of cents \( \leq 1 \), respectively.

Answer:

The greedy choice property for this coin change problem is that there is an optimal solution for \( m \) cents will use the highest possible value coin.

Proof.

We prove by separating the cases \( m \geq 5 \), \( 2 \leq m < 5 \), and \( m \leq 1 \).

(a) \( m \geq 5 \). Assume \( A \) to be any optimal solution.

If \( c_5 \in A \), then we prove the claim.

If \( c_5 \notin A \), there are following scenarios:

- \( c_2 \notin A \), there are at least 5 \( c_1 \)'s in \( A \)

\[ B = A - \{c_1, c_1, c_1, c_1, c_1\} \cup \{c_5\} \]

is also a solution but \( |B| < |A| \) contradict that \( A \) is optimal.

- one \( c_2 \) and at least 3 \( c_1 \)'s in \( A \)

\[ B = A - \{c_1, c_1, c_1, c_2\} \cup \{c_5\} \]

is also a solution but \( |B| < |A| \) contradict that \( A \) is optimal.

- two \( c_2 \)'s and at least 1 \( c_1 \)'s in \( A \)

\[ B = A - \{c_1, c_2, c_2\} \cup \{c_5\} \]

is also a solution but \( |B| < |A| \) contradict that \( A \) is optimal.

- three \( c_2 \)'s in \( A \)

\[ B = A - \{c_2, c_2, c_2\} \cup \{c_5, c_1\} \]

is also a solution but \( |B| < |A| \) contradict that \( A \) is optimal.

Conclusion: \( c_5 \in A \), thus we prove the claim.

(b) \( 2 \leq m < 5 \). Assume \( A \) to be any optimal solution.

If \( A \) contains \( c_2 \), we prove the claim.

If \( A \) does not contain \( c_2 \), then there are at least 2 \( c_1 \)'s in \( A \)

\[ B = A - \{c_1, c_1\} \cup \{c_2\} \]

is also a solution but \( |B| < |A| \) contradict that \( A \) is optimal.

Conclusion: \( c_2 \in A \), thus we prove the claim.
(c) $m \leq 1$. Assume $A$ to be any optimal solution.
   If $m = 1$, obvious $c_1$ is in any optimal solution.