CS/ECE 374 P17

Pengxu Zheng, Junquan Chen, Jiawei Tang

TOTAL POINTS

100 / 100

QUESTION 1

1 Problem 17.A. 70 / 70

✓ **+ 70 pts** **Correct**

+ 42 pts Correct Recurrence: English description (7) + final answer (7) + base case (7) + recursive case (21).

+ 28 pts DP implementation detail: data structure (7) + evaluation order (14) + running time analysis (7)

+ 7 pts English description: correct and clear English description of the variable/what the algorithm is computing. If this is missing, extra 15 points off.

+ 7 pts Final answer: how to call your algorithm to get the final answer or which variable value (on which parameters) to return

+ 7 pts Correct base case(s)

+ 21 pts Correct recursive case(s). If recursive case is wrong, no credits for DP implementation detail.

+ 7 pts Correct memoization data structure

+ 14 pts Correct evaluation order

+ 7 pts Correct and right running time analysis

+ 17.5 pts IDK

+ 0 pts Incorrect; Not understanding the question (see comments below)

- 10 pts Using code (that is hard to read) rather than pseudocode

- 15 pts Extra penalty for not having English description

QUESTION 2

2 Problem 17.B. 30 / 30

✓ **+ 30 pts** **Correct**

+ 10 pts English description of backtracking idea

+ 10 pts Appropriate auxiliary data structure

+ 10 pts Correct implementation detail. If implementation is too hand-waving, this does not apply.

+ 0 pts Incorrect (see comments below)

+ 7.5 pts IDK

- 5 pts Using code (that is hard to read) instead of pseudocode

HW Solution

CS/ECE 374: Algorithms & Models of Computation, Spring 2019 Version: **1.0**

Submitted by:

- **≪Jiawei Tang≫**: **≪jiaweit2≫**
- **≪Junquan Chen≫**: **≪junquan2≫**
- **≪Pengxu Zheng≫**: **≪pzheng5≫**

17

Solution:

17.A. Let $MLD(i,j)$ denote the minimized L1-distance, where $0 \leq i \leq m$, $0 \leq j \leq n$. There are two cases: b_j will either have a match to a_i(the optimal solution) or b_j will not have a match to a_i. This function obeys the following recurrence:

$$\text{MLD(i,j)} = \begin{cases} \infty, & \text{if } j = n+1 \text{ and } i \neq m+1 \\ 0, & \text{if } i = m+1 \\ \min(MLD(i+1,j+1) + |a_i - b_j|, MLD(i,j+1)) & \text{otherwise} \end{cases}$$

We need to compute $MLD(0,0)$. We can memorize the function MLD into an array $M[0..m{+}1, 0..n{+}1]$. Each entry $MLD[i,j]$ depends on entries in the next column so we fill the array in reverse column-major order.

for $i \leftarrow 0$ to $n+1$ **do**
  $M[m+1, i] \leftarrow \infty$
end for
for $i \leftarrow 0$ to $m+1$ **do**
  $M[i, n+1] \leftarrow \infty$
end for
$M[m+1, n+1] \leftarrow 0$ //above are base cases
for $j \leftarrow n$ down to 0 **do**
  for $i \leftarrow m$ down to 0 **do**
    $M[i,j] = \min(M[i+1,j+1] + |a_i - b_j|, M[i,j+1])$
  end for
end for

There are $O(mn)$ subproblems and each requires $O(1)$ time. Therefore the total runtime is $O(mn)$.

1 Problem 17.A. **70 / 70**

✓ **+ 70 pts** **Correct**

+ 42 pts Correct Recurrence: English description (7) + final answer (7) + base case (7) + recursive case (21).

+ 28 pts DP implementation detail: data structure (7) + evaluation order (14) + running time analysis (7)

+ 7 pts English description: correct and clear English description of the variable/what the algorithm is computing. If this is missing, extra 15 points off.

+ 7 pts Final answer: how to call your algorithm to get the final answer or which variable value (on which parameters) to return

+ 7 pts Correct base case(s)

+ 21 pts Correct recursive case(s). If recursive case is wrong, no credits for DP implementation detail.

+ 7 pts Correct memoization data structure

+ 14 pts Correct evaluation order

+ 7 pts Correct and right running time analysis

+ 17.5 pts IDK

+ 0 pts Incorrect; Not understanding the question (see comments below)

- 10 pts Using code (that is hard to read) rather than pseudocode

- 15 pts Extra penalty for not having English description

17.B. We will record the last 3 js that satisfy $M[i+1, j+1] + |a_i - b_j| < M[i, j+1]$. To modify the algorithm in 17.A, we first define an empty array A for storing the potential optimal subsequence. Then we change what we have earlier inside of the two for loops. The modified pseudocode is as below:

$A = [\,]$
for $i \leftarrow 0$ to $n+1$ **do**
  $M[m+1, i] \leftarrow \infty$
end for
for $i \leftarrow 0$ to $m+1$ **do**
  $M[i, n+1] \leftarrow \infty$
end for
$M[m+1, n+1] \leftarrow 0$ //above are base cases
for $j \leftarrow n$ down to 0 **do**
  for $i \leftarrow m$ down to 0 **do**
    if $M[i+1, j+1] + |a_i - b_j| < M[i, j+1]$ **then**
      if $j \notin A$ **then**
        A.append(j)
      end if
    end if
  end for
end for

The optimal subsequence is the last m of A, which is $A[-m:]$.

2 Problem 17.B. 30 / 30

- ✓ **+ 30 pts** **Correct**
- **+ 10 pts** English description of backtracking idea
- **+ 10 pts** Appropriate auxiliary data structure
- **+ 10 pts** Correct implementation detail. If implementation is too hand-waving, this does not apply.
- **+ 0 pts** Incorrect (see comments below)
- **+ 7.5 pts** IDK
- **- 5 pts** Using code (that is hard to read) instead of pseudocode