CS/ECE 374 P19

Pengxu Zheng, Jiawei Tang

TOTAL POINTS

45 / 100

QUESTION 1

1 19.A. 25 / 25

✓ **- 0 pts Correct**

- 5 pts Direct proof: Minor errors

- 15 pts Direct proof: Major errors, but has the right idea

- 25 pts Direct proof: Incorrect

- 10 pts Proof by contradiction: Assumption is wrong or ambiguous

- 5 pts Proof by contradiction: A contradiction is obtained, but with minor errors in proof

- 10 pts Proof by contradiction: No contradiction is obtained, but correctly identifies strings of length at least n can be made shorter in some way

- 15 pts Proof by contradiction: No contradiction is obtained

- 25 pts Weak induction used (deadly sin)

- 5 pts Induction: Missing/mistake in base case

- 10 pts Induction: Inductive hypothesis missing/stated incorrectly

- 10 pts Induction: Mistake in inductive step

- 18.75 pts IDK

QUESTION 2

2 19.B. 20 / 25

- 0 pts Correct

✓ **- 5 pts Direct proof: Minor errors**

- 15 pts Direct proof: Major errors, but has the right idea

- 25 pts Direct proof: Incorrect

- 10 pts Proof by contradiction: Assumption is wrong or ambiguous

- 5 pts Proof by contradiction: A contradiction is obtained, but with minor errors in proof

- 10 pts Proof by contradiction: No contradiction is obtained, but correctly identifies strings of length larger than (n+1)(m+1) can be made shorter in some way.

- 15 pts Proof by contradiction: No contradiction is obtained

- 25 pts Weak induction used (deadly sin)

- 5 pts Induction: Missing/mistake in base case

- 10 pts Induction: Inductive hypothesis missing/stated incorrectly

- 10 pts Induction: Mistake in inductive step

- 18.75 pts IDK

(n-1)(m+1) + the m original characters

QUESTION 3

3 19.C. 0 / 50

+ 50 pts Correct

+ 30 pts Correct Recurrence: English description (5) + final answer (5) + base cases (5) + recursive cases (15).

+ 20 pts DP implementation details: data structure (5) + evaluation order (10) + running time analysis (5)

+ 5 pts English description: correct and clear English description of the variables/what the algorithm is computing. If this is missing, extra 10 points off (see below)

✓ **+ 5 pts Final answer: how to call your algorithm to get the final answer or which variable value (on which parameters) to return**

✓ **+ 5 pts Correct base case(s)**

+ 15 pts Correct recursive case(s). If recursive case is wrong, no credits for the DP implementation details.

+ 5 pts Correct memoization data structure

+ 10 pts Correct evaluation order

+ 5 pts Correct and right running time analysis

+ **12.5 pts** IDK

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

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

✓ **- 10 pts Extra penalty for not having English description**

+ **25 pts** Product of DFAs: Correctly builds a DFA for recognizing superstrings of w that uses O(|w|) states.

+ **5 pts** Product of DFAs: Correctly builds a DFA for recognizing superstrings of w that uses O(2^{|w|}) states.

+ **0 pts** Product of DFAs: Refers to, but does not construct, a DFA that recognizes superstrings of w

+ **5 pts** Product of DFAs: Correctly forms the product of the given DFA with the superstring DFA

+ **10 pts** Product of DFAs: Computes the length of the shortest path from the start state of the product machine to any accepting state (using, say, BFS)

+ **10 pts** Product of DFAs: Correct runtime analysis

HW Solution

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

Submitted by:

- **≪Jiawei Tang≫**: **≪jiaweit2≫**
- **≪Pengxu Zheng≫**: **≪pzheng5≫**

19

Solution:

19.A. Since M only has n states in total, we can assume a situation where we need the longest string w'' as input to transition a state q to another state q' without being in a state for more than once. This rule of transitioning to states at most once will match the requirement of the problem to find the shortest string that transition a state to another state but now we are trying to find the maximized such string. If $|w''| \geq n$, at least one state will be transitioned to for at least two times which violates our rule. Thus, w'' is of length at most $n-1$.

19.B. Since s is the superstring of w, we can add any string before or after w or between any character of w. There are $m+1$ positions to add any string. Then from 19.A, we know that the length of the shortest string w'' to transition one state to another state is at most $n-1$. For each position, we can have one such w''. Therefore, the shortest string in this case for each position can be at most $n-1$. Therefore, for x, it is of length at most $(n-1)(m+1) < (n+1)(m+1)$.

19.C. We create a $m \times n$ matrix M for memorization. I will have a function $f(q,i)$, where $q \in Q, 0 \leq i \leq m$. We will start by the base case $f(a,m) = \epsilon$ where $a \in A$. The final answer should be $f(s,0)$. My plan is to iterate through every symbol in the alphabet until we get to the final answer. We have two cases, the symbol is in w or not for each time.

```
shortestString = A string has infinite length
for q ∈ A do
  i = m
  while i ≥ 0 do
    for c in Σ do
      Find q1 s.t. δ(q1, c) = q, if not found, then continue // It needs O(n)
      if |M[q, i].append(c)| < |M[q, i − 1].append(wi)| then
        M[q1, i] = M[q, i].append(c)
      else
        M[q1, i − 1] = M[q, i − 1].append(wi)
        i− = 1
      end if
    end for
  end while
  if M[S, 0] ≠ null and M[S, 0] < shortestString then
    shortestString = M[q, 0]
  end if
end for
```

We will output *shortestString* in the end. Since $\Sigma = O(1)$, there are $O(|A|)$ subproblems, each with $O(mn)$. Thus, the total time is $O(|A|mn)$.

1 19.A. **25 / 25**

✓ **- 0 pts Correct**

- 5 pts Direct proof: Minor errors

- 15 pts Direct proof: Major errors, but has the right idea

- 25 pts Direct proof: Incorrect

- 10 pts Proof by contradiction: Assumption is wrong or ambiguous

- 5 pts Proof by contradiction: A contradiction is obtained, but with minor errors in proof

- 10 pts Proof by contradiction: No contradiction is obtained, but correctly identifies strings of length at least n can be made shorter in some way

- 15 pts Proof by contradiction: No contradiction is obtained

- 25 pts Weak induction used (deadly sin)

- 5 pts Induction: Missing/mistake in base case

- 10 pts Induction: Inductive hypothesis missing/stated incorrectly

- 10 pts Induction: Mistake in inductive step

- 18.75 pts IDK

HW Solution

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

Submitted by:

- **≪Jiawei Tang≫**: **≪jiaweit2≫**
- **≪Pengxu Zheng≫**: **≪pzheng5≫**

19

Solution:

19.A. Since M only has n states in total, we can assume a situation where we need the longest string w'' as input to transition a state q to another state q' without being in a state for more than once. This rule of transitioning to states at most once will match the requirement of the problem to find the shortest string that transition a state to another state but now we are trying to find the maximized such string. If $|w''| \geq n$, at least one state will be transitioned to for at least two times which violates our rule. Thus, w'' is of length at most $n-1$.

19.B. Since s is the superstring of w, we can add any string before or after w or between any character of w. There are $m+1$ positions to add any string. Then from 19.A, we know that the length of the shortest string w'' to transition one state to another state is at most $n-1$. For each position, we can have one such w''. Therefore, the shortest string in this case for each position can be at most $n-1$. Therefore, for x, it is of length at most $(n-1)(m+1) < (n+1)(m+1)$.

19.C. We create a $m \times n$ matrix M for memorization. I will have a function $f(q,i)$, where $q \in Q, 0 \leq i \leq m$. We will start by the base case $f(a,m) = \epsilon$ where $a \in A$. The final answer should be $f(s,0)$. My plan is to iterate through every symbol in the alphabet until we get to the final answer. We have two cases, the symbol is in w or not for each time.

```
shortestString = A string has infinite length
for q ∈ A do
  i = m
  while i ≥ 0 do
    for c in Σ do
      Find q1 s.t. δ(q1, c) = q, if not found, then continue // It needs O(n)
      if |M[q, i].append(c)| < |M[q, i − 1].append(wi)| then
        M[q1, i] = M[q, i].append(c)
      else
        M[q1, i − 1] = M[q, i − 1].append(wi)
        i− = 1
      end if
    end for
  end while
  if M[S, 0] ≠ null and M[S, 0] < shortestString then
    shortestString = M[q, 0]
  end if
end for
```

We will output *shortestString* in the end. Since $\Sigma = O(1)$, there are $O(|A|)$ subproblems, each with $O(mn)$. Thus, the total time is $O(|A|mn)$.

2 19.B. **20 / 25**

- **0 pts** Correct

✓ - **5 pts** **Direct proof: Minor errors**

- **15 pts** Direct proof: Major errors, but has the right idea
- **25 pts** Direct proof: Incorrect
- **10 pts** Proof by contradiction: Assumption is wrong or ambiguous
- **5 pts** Proof by contradiction: A contradiction is obtained, but with minor errors in proof
- **10 pts** Proof by contradiction: No contradiction is obtained, but correctly identifies strings of length larger than (n+1)(m+1) can be made shorter in some way.
- **15 pts** Proof by contradiction: No contradiction is obtained
- **25 pts** Weak induction used (deadly sin)
- **5 pts** Induction: Missing/mistake in base case
- **10 pts** Induction: Inductive hypothesis missing/stated incorrectly
- **10 pts** Induction: Mistake in inductive step
- **18.75 pts** IDK

(n-1)(m+1) + the m original characters

HW Solution

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

Submitted by:

- **≪Jiawei Tang≫**: **≪jiaweit2≫**
- **≪Pengxu Zheng≫**: **≪pzheng5≫**

19

Solution:

19.A. Since M only has n states in total, we can assume a situation where we need the longest string w'' as input to transition a state q to another state q' without being in a state for more than once. This rule of transitioning to states at most once will match the requirement of the problem to find the shortest string that transition a state to another state but now we are trying to find the maximized such string. If $|w''| \geq n$, at least one state will be transitioned to for at least two times which violates our rule. Thus, w'' is of length at most $n-1$.

19.B. Since s is the superstring of w, we can add any string before or after w or between any character of w. There are $m+1$ positions to add any string. Then from 19.A, we know that the length of the shortest string w'' to transition one state to another state is at most $n-1$. For each position, we can have one such w''. Therefore, the shortest string in this case for each position can be at most $n-1$. Therefore, for x, it is of length at most $(n-1)(m+1) < (n+1)(m+1)$.

19.C. We create a $m \times n$ matrix M for memorization. I will have a function $f(q,i)$, where $q \in Q, 0 \leq i \leq m$. We will start by the base case $f(a,m) = \epsilon$ where $a \in A$. The final answer should be $f(s,0)$. My plan is to iterate through every symbol in the alphabet until we get to the final answer. We have two cases, the symbol is in w or not for each time.

```
shortestString = A string has infinite length
for q ∈ A do
  i = m
  while i ≥ 0 do
    for c in Σ do
      Find q1 s.t. δ(q1, c) = q, if not found, then continue // It needs O(n)
      if |M[q, i].append(c)| < |M[q, i − 1].append(wi)| then
        M[q1, i] = M[q, i].append(c)
      else
        M[q1, i − 1] = M[q, i − 1].append(wi)
        i− = 1
      end if
    end for
  end while
  if M[S, 0] ≠ null and M[S, 0] < shortestString then
    shortestString = M[q, 0]
  end if
end for
```

We will output *shortestString* in the end. Since $\Sigma = O(1)$, there are $O(|A|)$ subproblems, each with $O(mn)$. Thus, the total time is $O(|A|mn)$.

3 19.C. **0 / 50**

+ **50 pts** Correct

+ **30 pts** Correct Recurrence: English description (5) + final answer (5) + base cases (5) + recursive cases (15).

+ **20 pts** DP implementation details: data structure (5) + evaluation order (10) + running time analysis (5)

+ **5 pts** English description: correct and clear English description of the variables/what the algorithm is computing. If this is missing, extra 10 points off (see below)

✓ **+ 5 pts Final answer: how to call your algorithm to get the final answer or which variable value (on which parameters) to return**

✓ **+ 5 pts Correct base case(s)**

+ **15 pts** Correct recursive case(s). If recursive case is wrong, no credits for the DP implementation details.

+ **5 pts** Correct memoization data structure

+ **10 pts** Correct evaluation order

+ **5 pts** Correct and right running time analysis

+ **12.5 pts** IDK

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

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

✓ **- 10 pts Extra penalty for not having English description**

+ **25 pts** Product of DFAs: Correctly builds a DFA for recognizing superstrings of w that uses O(|w|) states.

+ **5 pts** Product of DFAs: Correctly builds a DFA for recognizing superstrings of w that uses O(2^{|w|}) states.

+ **0 pts** Product of DFAs: Refers to, but does not construct, a DFA that recognizes superstrings of w

+ **5 pts** Product of DFAs: Correctly forms the product of the given DFA with the superstring DFA

+ **10 pts** Product of DFAs: Computes the length of the shortest path from the start state of the product machine to any accepting state (using, say, BFS)

+ **10 pts** Product of DFAs: Correct runtime analysis