CS/ECE 374 P21

Jiawei Tang, Pengxu Zheng

TOTAL POINTS

100 / 100

QUESTION 1

1 21.A. 50 / 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

QUESTION 2

2 21.B. 50 / 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

HW7 Solution

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

Submitted by:

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

21

Solution:

21.A.

We define the function l-feasible(P, Q, R, l) to compute the possible path constituted of l-legal configurations of (p_i, q_j, r_k). The notation of l-legal(p_i, q_j, r_k) function bears the assumption that it can be calculated via O(1) time, as stated in the problem statement.

l-feasible(P, Q, R, l) = NULL, if $\Delta(p_1, q_1, r_1)$, $\Delta(p_n, q_n, r_n) >$ l.
l-feasible(P, Q, R, l) = 1, for i = j = k = 1 in (p_i, q_j, r_k).
l-feasible(P, Q, R, l) = l-legal(p_i, q_j, r_k)* [l-feasible(p_{i-1}, q_j, r_k) +
l-feasible(p_i, q_{j-1}, r_k) + l-feasible(p_i, q_j, r_{k-1}) +
l-feasible(p_{i-1}, q_{j-1}, r_k) + l-feasible(p_{i-1}, q_j, r_{k-1}) +
l-feasible(p_i, q_{j-1}, r_{k-1}) + l-feasible($p_{i-1}, q_{j-1}, r_{k-1}$)]

The pseudocode is as follows:

```
def l-feasible(P, Q, R, l):
    int arr[n+1][n+1][n+1];
    for p_i in P:
        for q_j in Q:
            for r_k in R:
                if i, j, k = 1:
                    arr[1][1][1] = 1
                else:
                    arr[i][j][k] = l-legal(p_i, q_j, r_k)* [l-feasible(p_{i-1}, q_j, r_k) +
                        l-feasible(p_i, q_{j-1}, r_k) + l-feasible(p_i, q_j, r_{k-1}) +
                        l-feasible(p_{i-1}, q_{j-1}, r_k) + l-feasible(p_{i-1}, q_j, r_{k-1}) +
                        l-feasible(p_i, q_{j-1}, r_{k-1}) + l-feasible(p_{i-1}, q_{j-1}, r_{k-1})]
    return arr
```

The array arr stands for the memoization structure in this problem. Since there should be n*n*n elements in arr in total, we conclude that our algorithm runs in $O(n^3)$ time.

21.B.

To calculate l, we need to do modification to our solution in part A as follows:

arr[i][j][k] = max ($\Delta(p_i, q_j, r_k)$, min (
l-feasible(p_{i-1}, q_j, r_k) +
l-feasible(p_i, q_{j-1}, r_k) + l-feasible(p_i, q_j, r_{k-1}) +
l-feasible(p_{i-1}, q_{j-1}, r_k) + l-feasible(p_{i-1}, q_j, r_{k-1}) +
l-feasible(p_i, q_{j-1}, r_{k-1}) + l-feasible($p_{i-1}, q_{j-1}, r_{k-1}$))

The runtime will remain unchanged.

1 21.A. **50 / 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

HW7 Solution

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

Submitted by:

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

21

Solution:

21.A.

We define the function l-feasible(P, Q, R, l) to compute the possible path constituted of l-legal configurations of (p_i, q_j, r_k). The notation of l-legal(p_i, q_j, r_k) function bears the assumption that it can be calculated via O(1) time, as stated in the problem statement.

l-feasible(P, Q, R, l) = NULL, if $\Delta(p_1, q_1, r_1)$, $\Delta(p_n, q_n, r_n) >$ l.
l-feasible(P, Q, R, l) = 1, for i = j = k = 1 in (p_i, q_j, r_k).
l-feasible(P, Q, R, l) = l-legal(p_i, q_j, r_k)* [l-feasible(p_{i-1}, q_j, r_k) +
l-feasible(p_i, q_{j-1}, r_k) + l-feasible(p_i, q_j, r_{k-1}) +
l-feasible(p_{i-1}, q_{j-1}, r_k) + l-feasible(p_{i-1}, q_j, r_{k-1}) +
l-feasible(p_i, q_{j-1}, r_{k-1}) + l-feasible($p_{i-1}, q_{j-1}, r_{k-1}$)]

The pseudocode is as follows:

```
def l-feasible(P, Q, R, l):
    int arr[n+1][n+1][n+1];
    for p_i in P:
        for q_j in Q:
            for r_k in R:
                if i, j, k = 1:
                    arr[1][1][1] = 1
                else:
                    arr[i][j][k] = l-legal(p_i, q_j, r_k)* [l-feasible(p_{i-1}, q_j, r_k) +
                        l-feasible(p_i, q_{j-1}, r_k) + l-feasible(p_i, q_j, r_{k-1}) +
                        l-feasible(p_{i-1}, q_{j-1}, r_k) + l-feasible(p_{i-1}, q_j, r_{k-1}) +
                        l-feasible(p_i, q_{j-1}, r_{k-1}) + l-feasible(p_{i-1}, q_{j-1}, r_{k-1})]
    return arr
```

The array arr stands for the memoization structure in this problem. Since there should be n*n*n elements in arr in total, we conclude that our algorithm runs in $O(n^3)$ time.

21.B.

To calculate l, we need to do modification to our solution in part A as follows:

arr[i][j][k] = max ($\Delta(p_i, q_j, r_k)$, min (
l-feasible(p_{i-1}, q_j, r_k) +
l-feasible(p_i, q_{j-1}, r_k) + l-feasible(p_i, q_j, r_{k-1}) +
l-feasible(p_{i-1}, q_{j-1}, r_k) + l-feasible(p_{i-1}, q_j, r_{k-1}) +
l-feasible(p_i, q_{j-1}, r_{k-1}) + l-feasible($p_{i-1}, q_{j-1}, r_{k-1}$))

The runtime will remain unchanged.

2 21.B. **50 / 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