Midterm 2 Solution

Name:

SID:

Exam Room:

SID of student to your left:

SID of student to your right:

Do not turn this page until your instructor tells you to do so.

- After the exam starts, please *write your SID on the front of every page.* We may deduct points if your name is missing from any page. You will not be allowed to fill in your name after time is called.
- For short question, your answers must be written clearly inside the box region. Any answer outside the box will not be graded. For longer question, if you run out of space, you must clearly mention in the space provided for the question if part of your answers are elsewhere.
- Try to answer all questions. Not all parts of a problem are weighted equally. Before you answer any question, read the problem carefully. Be precise and concise in your answers.
- You may use the blank page at the back for scratch work, but we will not look at it for grading.
- You may consult only *two sheet of notes.* Apart from that, you may not look at books, notes, etc. Calculators, phones, computers, and other electronic devices are NOT permitted.
- There are **14** pages (7 double sided sheets) on the exam. Notify a proctor immediately if a page is missing.
- **Any algorithm covered in the lecture can be used as a blackbox.**
- **You have 110 minutes: there are 8 questions on this exam worth a total of 110 points.**
- Indicate the discussion you attend on page 2 **after** the exam starts. Fill in the square.
- Good luck!

Select the following by filling in the square. If you go to different sections approximately the same number of times during the semester, feel free to select multiple section times.

TA and section time:

☐ I don't go to discussion section

☐ 9am, Raymond, Etcheverry 3109

☐ 10am, Simin, Wheeler 220

☐ 10am, Zhen, Wheeler 220

☐ 10am, Alex, Soda 310

☐ 11am, Jerry, Hearst Field Annex B5

☐ 11am, Aditya Baradwaj, Etcheverry 3119

☐ 11am, Ching, Soda 310

☐ 12pm, Aditya Mishra, Hearst Field Annex B5

☐ 12pn, Aditya Baradwaj, Cory 289

☐ 12pm, Kevin, Soda 310

☐ 1pm, Raymond, Hearst Field Annex B5

☐ 1pm, Jingcheng, Etcheverry 3113

☐ 2pm, Edmund, Etcheverry 3111

☐ 2pm, Ching, Etcheverry 3113

☐ 3pm, Gary, Leconte 385

☐ 3pm, Mudit, Etcheverry 3113

☐ 4pm, Nathan, Dwinelle 215

☐ 4pm, Kevin, Wheeler 24

☐ 4pm, Alex, Dwinelle 242

☐ 5pm, Pasin, Etcheverry 3119

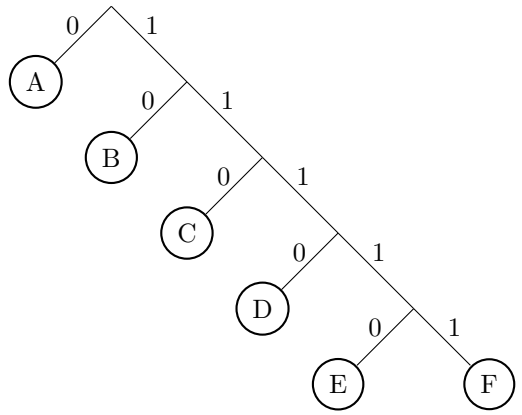
☐ 5pm, Peihan, Dwinelle 79

SID:

1. Huffman Encoding (6 points)

A document consists of letters $\{A, B, C, D, E, F\}$ occurring with various frequencies. Suppose the following is the tree corresponding to an optimal Huffman encoding of the letters $\{A, B, C, D, E, F\}$ (the optimal tree need not be unique).

Suppose the frequencies of E and F are 1 and 1 (both occur only once) respectively. Here frequency refers to the number of occurrences in the document. What is the smallest possible frequency for each of the following letters?



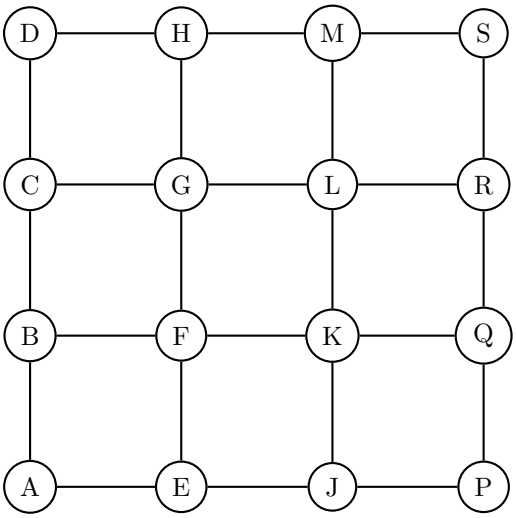
Letter	Frequency
A	5
B	3
C	2

Explanation:

- If $f_D < f_E = f_F = 1$, then swapping D with either E or F gives a better encoding. Hence, $f_D \geq 1$.
- If $f_C < f_E + f_F = 2$, then swapping C with the tree joining E and F gives a better encoding. Hence, $f_C \geq 2$.
- If $f_B < f_D + f_E + f_F$, the swapping B with the tree joining D, E and F gives a better encoding. Hence, we have $f_B \geq f_D + f_E + f_F \geq 3$.
- If $f_A < f_C + f_D + f_E + f_F$, the swapping A with the tree joining C, D, E and F gives a better encoding. Hence, we have $f_A \geq f_C + f_D + f_E + f_F \geq 5$.
- To see that $f_D = 1, f_C = 2, f_B = 3, f_A = 5$ can indeed result in the specified tree, consider the case where $f_A = 5, f_B = 3, f_C = 2, f_D = f_E = f_F = 1$. By running Huffman encoding algorithm, we can see that it produces the above tree.

2. Max Flow (10 points)

Consider the following undirected graph with all edge capacities equal to 1. An undirected edge of capacity 1 can carry flow of up to 1 unit in either direction.



(a) What is the value of the maximum flow from A to S?

2

Explanation: a flow of value two is $A \to B \to C \to D \to H \to M \to S$ and $A \to E \to J \to P \to Q \to R \to S$ (each of value one). To see that this flow is optimal, observe that the cut $\{A\}$ and the rest of the graph has value only two.

(b) What is the value of the maximum flow from F to L?

4

Explanation: a flow of value four is $F \to G \to L, F \to K \to L, F \to B \to C \to D \to H \to M \to L$ and $F \to E \to J \to P \to Q \to R \to L$ (each of value one). To see that this flow is optimal, observe that the cut $\{F\}$ and the rest of the graph has value only four.

(c) Call an edge to be critical, if removing the edge reduces the value of the maximum flow from F to L.

For each of the following edges, exhibit a minimum cut they are part of, thereby proving that they are critical. Specify the minimum cut by marking the vertices on one side of the cut in the table.

(a) Edge FB

Vertices on B side of cut :

A	B	C	D	E	F	G	H	J	K	L	M	P	Q	R	S
X	X	X	X	X		X	X	X	X	X	X	X	X	X	X

(b) Edge HM

Vertices on H side of cut :

A	B	C	D	E	F	G	H	J	K	L	M	P	Q	R	S
X	X	X	X	X	X	X	X								

3. Alternative Recurrence Relations (20 points)

For the longest increasing subsequence and edit distance problems, here are candidate alternate definitions of subproblems. For each of these subproblems, **fill in** the square for either "Recurrence relation below" OR "NO recurrence relation". If there exists a valid recurrence relation which solves the problem, write

the recurrence relation in the answer box. For parts (a) - (c), you are trying to solve the longest increasing subsequence problem and for parts (d) and (e), you are trying to solve the edit distance problem.

(To avoid guessing, if you incorrectly say "NO recurrence relation" then you lose twice as many points).

(a) $T[i]$ = length of longest increasing sequence in $A[i, \ldots, n]$ starting at $A[i]$ (i.e. $A[i]$ is part of the longest increasing sequence).

$T[i] = \max_{i<j\leq n}\{1 + T[j]\}$

(b) $T[i]$ = length of longest increasing sequence in $A[1, \ldots, i]$.

No recurrence relation.

(c) $T[i]$ = length of longest increasing sequence in $A[i, \ldots, n]$

No recurrence relation.

(d) $E[i, j]$ = edit distance of $A[i, \ldots, j]$ to $B[i, \ldots j]$.

No recurrence relation.

(e) $E[i, j, k, \ell]$ = edit distance of $A[i, \ldots, j]$ to $B[k, \ldots, \ell]$.

$$E[i,j,k,l] = \min \begin{cases} E[i, j-1, k, l] + 1 \\ E[i, j, k, l-1] + 1 \\ E[i+1, j, k, l] + 1 \\ E[i, j, k+1, l] + 1 \\ E[i, j-1, k, l-1] + \text{diff}\{A[j], B[l]\} \\ E[i+1, j, k+1, l] + \text{diff}\{A[i], B[k]\} \end{cases}$$

4. HORN-SAT (6 points)

On running the Horn-SAT algorithm on a horn-SAT formula, the following assignment is obtained:

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
T	T	T	F	F	F	F	F	F	T

Suppose we modify the horn-SAT formula in the following ways, indicate whether the new formula is Necessarily Unsatisfiable or Necessarily Satisfiable or Can't Say. **Fill in** one of the choices.

Explanation: the variables that are assigned true are necessarily true, whereas we cannot say that about the variables assigned false. Hence, the only case where we can say for sure that the formula is unsatisfiable is when the added constraint is violated and all the variables involved are the true variables.

(a) Add new constraint $x_1 \wedge x_2 \longrightarrow x_4$

Can't Say

Note: an example of a formula resulting is satisfiable has clauses being x_1, x_2, x_3 and x_{10}. An example of a formula resulting is unsatisfiable has clauses being x_1, x_2, x_3, x_{10} and $x_1 \wedge x_2 \rightarrow \overline{x_4}$.

(b) Add new constraint $x_1 \wedge x_2 \longrightarrow \overline{x_3}$

Necessarily Unsatisfiable

5. Duality (12 points)

(a) Write down the dual for the following linear program:

$$\begin{aligned} \text{Maximize} \quad & x_1 + 3x_3 \\ \text{subject to} \quad & \\ & 2x_1 + 4x_2 \leq 120 \\ & 5x_1 - 6x_2 \leq 20 \\ & 7x_1 + 8x_2 + 9x_3 \leq 100 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Use y_i as decision variables for the dual.

Objective function:

$$\text{Minimize} \quad 120y_1 + 20y_2 + 100y_3$$

Constraints:

$$\begin{aligned} \text{subject to} \quad & \\ & 2y_1 + 5y_2 + 7y_3 \geq 1 \\ & 4y_1 - 6y_2 + 8y_3 \geq 0 \\ & 9y_3 \geq 3 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

Use below space for your work (we will use it while awarding partial credit):

$$\begin{aligned} & (y_1) \cdot (2x_1 + 4x_2) \leq 120 \cdot (y_1) \\ + & \Big[(y_2) \cdot (5x_1 - 6x_2) \leq 20 \cdot (y_2) \Big] \\ + & \Big[(y_3) \cdot (7x_1 + 8x_2 + 9x_3) \leq 100 \cdot (y_3) \Big] \end{aligned}$$

$$\begin{aligned} y_1(2x_1 + 4x_2) + y_2(5x_1 - 6x_2) + y_3(7x_1 + 8x_2 + 9x_3) &\leq 120y_1 + 20y_2 + 100y_3 \\ (2y_1 + 5y_2 + 7y_3)x_1 + (4y_1 - 6y_2 + 8y_3)x_2 + 9y_3x_3 &\leq 120y_1 + 20y_2 + 100y_3 \end{aligned}$$

(b) How does the dual change if we replace the constraint $2x_1 + 4x_2 \leq 120$ by an equality $2x_1 + 4x_2 = 120$? (No need to write out the dual again, just specify the change/changes to be made)

Remove nonnegativity constraint for y_1. Thus we have

$$\begin{aligned} \text{Minimize} \quad & 120y_1 + 20y_2 + 100y_3 \\ \text{subject to} \quad & 2y_1 + 5y_2 + 7y_3 \geq 1 \\ & 4y_1 - 6y_2 + 8y_3 \geq 0 \\ & 9y_3 \geq 3 \\ & y_2, y_3 \geq 0 \end{aligned}$$

Alternative Solution 1
We can turn $2x_1 + 4x_2 = 120$ into two constraints: $2x_1 + 4x_2 \leq 120$ and $2x_1 + 4x_2 \geq 120$. If we add y_4 for new constraint $2x_1 + 4x_2 \geq 120$,

$$\begin{aligned} & (y_1) \cdot (2x_1 + 4x_2) \leq 120 \cdot (y_1) \\ + & \Big[(y_2) \cdot (5x_1 - 6x_2) \leq 20 \cdot (y_2)\Big] \\ + & \Big[(y_3) \cdot (7x_1 + 8x_2 + 9x_3) \leq 100 \cdot (y_3)\Big] \\ + & \Big[(y_4) \cdot (-2x_1 - 4x_2) \leq -120 \cdot (y_4)\Big] \end{aligned}$$

$$\begin{aligned} y_1(2x_1 + 4x_2) + y_2(5x_1 - 6x_2) + y_3(7x_1 + 8x_2 + 9x_3) + y_4(-2x_1 - 4x_2) &\leq 120y_1 + 20y_2 + 100y_3 - 120y_4 \\ (2y_1 + 5y_2 + 7y_3 - 2y_4)x_1 + (4y_1 - 6y_2 + 8y_3 - 4y_4)x_2 + 9y_3x_3 &\leq 120y_1 + 20y_2 + 100y_3 - 120y_4 \end{aligned}$$

Dual becomes

$$\begin{aligned} \text{Minimize} \quad & 120y_1 + 20y_2 + 100y_3 - 120y_4 \\ \text{subject to} \quad & 2y_1 + 5y_2 + 7y_3 - 2y_4 \geq 1 \\ & 4y_1 - 6y_2 + 8y_3 - 4y_4 \geq 0 \\ & 9y_3 \geq 3 \\ & y_1, y_2, y_3, y_4 \geq 0 \end{aligned}$$

6. Lecture Scheduling (25 points)

A university wants to renovate lecture halls with the student's tuition. However, it must also schedule all the lectures of the existing courses. As a scheduling officer, you want to determine what is the minimum number of classrooms (lecture halls) that you need to schedule N lectures. Lectures are given to you in start-time finish-time pairs:

$$A[(s_1, f_1), (s_2, f_2), \ldots, (s_N, f_N)]$$

The university would like to schedule these lectures in the available classrooms without any conflicts – A conflict occurs if start time of a lecture is less than the finish time of an already scheduled lecture in a classroom.

(a) Design a greedy algorithm to find the minimum number of classrooms that are needed to schedule all lectures such that there are no conflicts.

i. What does the greedy algorithm do at each step? (Describe the main idea of the algorithm.

Schedule lectures by earliest start time first in the next available classroom. If current lecture cannot be scheduled in any existing classroom, add a new classroom and schedule it there. Answer will be the number of classroom needed.

Alternatively, can find the maximum conflict for a time slot. This is the minimum number of classrooms needed.

ii. Suppose your greedy algorithm finds a schedule with c classrooms, why is there no schedule with $c-1$ classrooms? (briefly argue in two-three sentences)

- Classroom c is open because lecture i is incompatible with all $c-1$ classrooms.
- These $c-1$ lectures each end after s_i.
- Since lecture i also end after s_i, we have c lectures that end after s_i. By sorting by start time, these lectures do not start after s_i.
- There are c lectures overlapping at s_i + some time
- All schedules must use $\geq c$ classrooms. Thus c is optimal at this step and optimal after scheduling all lectures.

(b) After running the greedy algorithm, the administration realized that not all lectures can be scheduled in the available c classrooms. Therefore, the university has now decided to cancel some of these lectures.

The administration would like to try out solving a linear program to determine the set of lectures to be cancelled. Suppose t_i is the loss in tuition from cancelling the i^{th} lecture, write a linear program to determine the set of lectures to be cancelled so as to minimize total loss in tuition.

i. What are the variables of the linear program?

Let x_i indicate whether lecture i is scheduled or not.

ii. What is the objective function?

$$\text{Minimize} \quad \sum_{i=1}^{N} t_i(1-x_i)$$

iii. What are the constraints?

Let C_i denote the set of lectures that conflicts with lecture i at the beginning:

$$C_i = \{j : s_j < s_i < f_j\}.$$

$$\begin{aligned} x_i + \sum_{j \in C_i} x_j &\leq c, \quad \forall i \in [N] \\ x_i &\leq 1, \quad \forall i \in [N] \\ x_i &\geq 0, \quad \forall i \in [N]. \end{aligned}$$

Explanation: at the beginning of every lecture, check if there are enough classrooms.

iv. It turns out that the optimal solution to the above linear program has a total tuition loss $= T$, but the university was forced to incur a loss strictly larger than T. When can this happen?

In general, the solution found by a linear program may not be integral. A fractional solution may assign 0.5 classroom to a lecture. To round it to an integral solution, this may incur a loss in the

objective function. This will happen if there is a non-integral corner point in the feasible region of a linear program, because every corner point is the "furthest" point in some direction, so it will be the unique optimal solution for some objective function; but on the other hand, since it is non-integral, so every integral solution nearby will necessarily be non-optimal for the same objective function.

7. Taco Trucks (20 points)

You are running a Taco truck company with 4 trucks. There are n cities that your company operates in. On day 0, all trucks are parked in city numbered 1. For the next m days, there is one street festival each day in one of the cities. Specifically, on day i, there is a street festival in city $c[i]$. In order to cater to a street festival on day i you need at least one truck to go the city $c[i]$.

Let $d[i, j]$ denote the distance between cities i and j.

Given the schedule $c[\cdot]$ of the street festivals and the distances $d[\cdot, \cdot]$ between the cities, design a dynamic programming algorithm that finds the optimal way to move the trucks around, so as to cater to every street festival while minimizing the total distance driven by all the trucks put together.

(a) What are the subproblems?

For all $1 \leq x, y, z, w \leq n$ and $1 \leq i \leq m$, let $C(i, x, y, z, w)$ be the total minimum distance serving the requests from $i + 1$ day onwards, starting with the trucks at cities x, y, z, w after day i.

Alternative Solution:

For all $1 \leq x, y, z, w \leq n$ and $1 \leq i \leq m$, let $C(i, x, y, z, w)$ be the total minimum distance covered by the optimal strategy for the following goal: serve the first i requests, and end up with trucks at cities x, y, z, w on day i.

(We could keep track of the locations of only 3 of the food trucks by defining $C(i, x, y, z)$ as the the accumulated distance traveled up until day i, where x, y, z are the current locations of each of the 3 food trucks that are not at $c[i]$ (since we know that at least one truck has to be at $c[i]$). This could save space used by the algorithm, although this is not required to receive full credit for the problem.)

(b) What are the base cases? Give an expression for the final answer in terms of the subproblems.

Base case: $C(m, x, y, z, w) = 0$ for all x, y, z, w.

Final answer: $C(0, 1, 1, 1, 1)$.

Alternative Solution:

Base case: $C(0, 1, 1, 1, 1) = 0$ and $C(0, x, y, z, w) = \infty$ for other x, y, z, w.

Final answer: $\min_{1 \leq x,y,z,w \leq n} C(m, x, y, z, w)$.

(c) What is the recurrence relation?

For the first definition of $C(\cdot, \cdot, \cdot, \cdot, \cdot)$, we can use

$$C(i, x, y, z, w) = \min \begin{cases} C(i+1, c[i+1], y, z, w) + d[x, c[i+1]] \\ C(i+1, x, c[i+1], z, w) + d[y, c[i+1]] \\ C(i+1, x, y, c[i+1], w) + d[z, c[i+1]] \\ C(i+1, x, y, z, c[i+1]) + d[w, c[i+1]]. \end{cases}$$

Alternative Solution:

For the alternative definition of $C(\cdot, \cdot, \cdot, \cdot, \cdot)$, we can use

$$C(i, x, y, z, w) = \begin{cases} \infty & \text{if } c[i] \notin \{x, y, z, w\} \\ \min_{1 \leq x', y', z', w' \leq n} C(i-1, x', y', z', w') + d[x', x] + d[y', y] + d[z', z] + d[w', w] & \text{otherwise.} \end{cases}$$

or

$$C(i,x,y,z,w) = \begin{cases} \infty & \text{if } c[i] \notin \{x,y,z,w\} \\ \min_{1\le t\le n} \begin{cases} C(i-1,t,y,z,w)+d[t,x] \\ C(i-1,x,t,z,w)+d[t,y] \\ C(i-1,x,y,t,w)+d[t,z] \\ C(i-1,x,y,z,t)+d[t,w] \end{cases} & \text{otherwise.} \end{cases}$$

Remark:

- *While the different recurrence relations above result in different running time of the algorithms, we give all recurrence relations full points. In general any recurrence relations that are correct and result in a polynomial time algorithm receive full points for this problem.*
- *Strictly speaking, for the alternative solution, we have to be more specific in the definition in that we do not allow the trucks to move without going to a cater target. While this definition and the one above result in the same final answer, the base case and the recurrence relation can be different. For instance, for the original definition, $C(0,x,y,z,w)$ when $(x,y,z,w) \neq (1,1,1,1)$ should be $d[1,x]+d[1,y]+d[1,z]+d[1,w]$, whereas for the more specific definition, the value should be ∞. (Note also that the two recurrence relations for the alternative definition can result in different values in the DP table, but both give correct final answer.) Nevertheless, we are lenient regarding this and any combination of the two definitions/base cases/recurrences will result in full score, as long as they produce the correct final answer.*

Pitfalls:

There are many ways to write the recurrence relations for both definitions of $C(\cdot,\cdot,\cdot,\cdot,\cdot)$. Instead of listing them all, we list a few of common pitfalls that we have seen (and how we grade them) below:

- For the alternative definition, some students forget the case where $c[i] \notin \{x,y,z,w\}$, for which the value of $C(i,x,y,z,w)$ should be ∞. This results in 0/2 points in "part (c) item 2" in the rubric.
- Regarding the previous point, some solutions track the condition $c[i] \in \{x,y,z,w\}$ implicitly. For instance, a recurrence relation below only looks at $C(i-1,x',y',z',w')$ where $c[i-1] \in \{x',y',z',w'\}$:

  $$C(i,x,y,z,w) = \min_{\substack{1\le x',y',z',w'\le n \\ c[i-1]\in\{x,y,z,w\}}} C(i-1,x',y',z',w') + d[x',x] + d[y',y] + d[z',z] + d[w',w].$$

  While this is almost correct, there is still one issue: the final answer must be minimum of $C(m,x,y,z,w)$ over all x,y,z,w such that at least one of them is $c[m]$. This is because the implicit checking does not apply in the last step. Since this is a minor point, we take of only 0.5 points for this (i.e. 1.5/2 in "part (c) item 2" in the rubric).
- For the alternative definition of $C(\cdot,\cdot,\cdot,\cdot,\cdot)$, some students use the following recurrence relation:

  $$C(i,x,y,z,w) = \begin{cases} \infty & \text{if } c[i] \notin \{x,y,z,w\} \\ \min \begin{cases} C(i-1,c[i-1],y,z,w)+d[t,x] \\ C(i-1,x,c[i-1],z,w)+d[t,y] \\ C(i-1,x,y,c[i-1],w)+d[t,z] \\ C(i-1,x,y,z,c[i-1])+d[t,w] \end{cases} & \text{otherwise.} \end{cases}$$

  This is actually incorrect, since it is possible that the truck that moves between day $i-1$ and day i is not the truck at $c[i-1]$. (A counterexample: if there are two days with $c[1]=2, c[2]=3$ and $d[1,2]=d[1,3]=1$ while $d[2,3]=2$, the above DP gives final answer of 3 whereas the actual optimum is 2; in the optimal solution, one truck at city 1 moves to city 2 on the first day and one truck at city 1 moves to city 3 on the second day.) Such solutions receive 2/3 points in "part (c) item 4" in the rubric.

- Solutions similar to the previous item but using $c[i]$ instead of $c[i-1]$ will only receive 1/3 points in "part (c) item 4" in the rubric.

SID:

8. Floyd-Warshall Paths (11 points)

Suppose $d(\cdot,\cdot,\cdot)$ is the dynamic programming table for an execution of the Floyd-Warshall algorithm on a graph G with n nodes. Recall that the subproblems $d(i,j,k)$ for $i,j,k \in \{1,\ldots,n\}$ is defined as follows:

$d(i,j,k) =$ length of the shortest path from i to j that only uses vertices in $\{1,\ldots,k\}$ as intermediate vertices

The recurrence relation is given by:

$$d(i,j,k) = \min\{d(i,j,k-1), d(i,k,k-1) + d(k,j,k-1)\}$$

Design an algorithm that given two vertices i and j, recovers the shortest path from i to j using the $d(.,.,.)$ table. The run-time of your algorithm to find the shortest path from i to j, must be equal to $O\left((\#\text{ of hops on shortest path from } i \text{ to } j) \cdot \log n\right)$.

Write Pseudocode below

```
def findShortest(i, j):
        if i == j: (Equivalent to check d(i,j,n) == d(i,j,0))
                return []
        else:
                Binary Search for largest k where d(i, j, k - 1)>d(i, j, n)
        return findShortest(i, k) + [k] + findShortest(k, j)
```