Design and Analysis of Algorithm

Dynamic Programming
(0/1 knapsack problem, All pair shortest path)

Lecture – 55-56

Overview

- Dynamic Programming is a general algorithm design technique for solving problems defined by recurrences with overlapping subproblems
- Invented by American mathematician "Richard Bellman) in the year 1950s to solve optimization problems and later assimilated by Computer Science.
- "Programming" here means "planning"

Dynamic Programming

- "Method of solving complex problems by breaking them down into smaller sub-problems, solving each of those sub-problems just once, and storing their solutions."
- The problem solving approach looks like Divide and conquer approach.(which is not true)

Dynamic Programming

Difference between Dynamic programming and Divide and Conquer approach.

Divide & Conquer	Dynamic Programming
1. Partitions a problem into independent smaller sub-problems	1. Partitions a problem into overlapping sub-problems
2. Doesn't store solutions of sub-problems. (Identical sub-problems may arise - results in the same computations are performed repeatedly.)	2. Stores solutions of sub-problems: thus avoids calculations of same quantity twice
3. Top down algorithms: which logically progresses from the initial instance down to the smallest sub-instances via intermediate sub-instances.	3. Bottom up algorithms: in which the smallest sub-problems are explicitly solved first and the results of these used to construct solutions to progressively larger sub-instances

Dynamic Programming

Is a Four-step methods

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

Dynamic Programming

Problems:

1. 0/1 Knapsack Problem
2. Floyd-Warshall Algorithm
3. Matrix Chain Multiplication
4. Longest Common Sub-sequence

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

- As the name suggests, items are indivisible here.
- We can not take the fraction of any item.
- We have to either take an item completely or leave it completely.
- It is solved using dynamic programming approach.

Lets solve with four step methods:

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Step 1: Characterize the structure of an optimal solution

Let there are n number of objects, their profit values are $\langle v_{1,}, v_2, v_3, \ldots \ldots, v_n \rangle$, weight are $\langle w_{1,}, w_2, w_3, \ldots \ldots, w_n \rangle$. The maximum capacity of Knapsack is "M".

The 0/1 knapsack problem can states as follows

$Maximize \ \sum_{i=1}^{n} v_i x_i$ (i.e. sum of the profit should be maximize)

$Subject\ to \ \sum_{i=1}^{n} w_i x_i \leq M$ (i.e. Sum of the weights should be less than or equal to capacity of the bag.)

$Where, x_i \in \{0,1\}$

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Step 2: Recursively define the value of optimal solution.

$Let\ c[i, j]\ is\ an\ two\ dimensional\ array, where$

$i = 0,1,2, \dots \dots, n$ (i.e. number of objects)

$j = 0,1,2, \dots \dots, M$ (i.e. maximum weight of knapsack)

$Then$

$$c[i, j] = \begin{cases} 0 & if\ i = 0\ \&j = 0 \\ c[i-1, j] & if\ \ 0 \le w_i > j \\ \max(c[i-1, j], c[i-1, j-w_i] + v_i) & if\ i > 0\ \&j \ge w_i \end{cases}$$

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Step 3: Compute optimal solution for 0/1 knapsack problem.

$\mathbf{0/1}\ \boldsymbol{Knapsack}\ (v, w, n, M)$

1. $For\ j = 0\ to\ M$
2. $\quad C[0, j] = 0$
3. $\quad keep[0, j] = 0$
4. $For\ i = 1\ to\ n$
5. $\quad For\ j = 0\ to\ M$
6. $\quad\quad if\ ((j \geq w[i]) \&\& (c[i-1, j - w[i]] + v[i]\ ) > c[i-1, j]))$
7. $\quad\quad\quad then\ \ c[i, j] = c\left[\mathrm{i} - 1, \mathrm{j} - \mathrm{w[i]}\right] + \mathrm{v[i]}$
8. $\quad\quad\quad\quad keep[i, j] = 1$
9. $\quad\quad else\ \ c[i, j] = c[\mathrm{i} - 1, \mathrm{j}]$
10. $\quad\quad\quad keep[i, j] = 0$
11. $Return\ c[n, M]$

Dynamic Programming

Problem 1: 0/1 Knapsack Problem (Implementation)

Consider-

Knapsack weight capacity = w

Number of items each having some weight and value = n

0/1 knapsack problem is solved using dynamic programming in the following steps-

Step-01:

- Draw a table say 'c' with (n+1) number of rows and (w+1) number of columns.
- Fill all the boxes of 0th row and 0th column with zeroes as shown in figure

0	1	2	3	...	w
1	0	0	0	...	0
2	0				
3	0				
....	....				
....	....				
n	0				

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Step-02:

- Start filling the table row wise top to bottom from left to right by using the following formula-

$$c\,(i\,,j) \;=\; \max\{\,c\,(\,i-1\,,j\,), c(\,i-1\,,j - w_i) + v_i\}$$

- Here, $c(i\,,j)$ = maximum value of the selected items if we can take items 1 to i and have weight restrictions of j.
- This step leads to completely filling the table.
- Then, value of the last box represents the maximum possible value that can be put into the knapsack.

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Step-03:

- To identify the items that must be put into the knapsack to obtain that maximum profit, Consider the last column of the table.
- Start scanning the entries from bottom to top.
- On encountering an entry whose value is not same as the value stored in the entry immediately above it, mark the row label of that entry.
- After all the entries are scanned, the marked labels represent the items that must be put into the knapsack.

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

Item	Weight	Profit
1	2	1
2	3	2
3	4	5
4	5	6

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

			0	1	2	3	4	5	6	7	8
P_i	w_i	0									
1	2	1									
2	3	2									
5	4	3									
6	5	4									

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0								
2	3	2	0								
5	4	3	0								
6	5	4	0								

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0								
2	3	2	0								
5	4	3	0								
6	5	4	0								

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0								
2	3	2	0								
5	4	3	0								
6	5	4	0								

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0		1						
2	3	2	0								
5	4	3	0								
6	5	4	0								

As maximum weight available is 2

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0								
5	4	3	0								
6	5	4	0								

As maximum weight available is 2

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0			2	2				
5	4	3	0								
6	5	4	0								

As maximum weight available is 2

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

			0	1	2	3	4	5	6	7	8
P_i	w_i	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0			2	2	3			
5	4	3	0								
6	5	4	0								

As maximum weight available is 2

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0								
6	5	4	0								

As two possible weight are available (i.e. 3 and 5)

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0				5				
6	5	4	0								

As maximum weight is available is : 4

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0				5	5	6		
6	5	4	0								

As maximum weight is available is : 6

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0				5	5	6	7	
6	5	4	0								

As maximum weight is available is : 7

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0								

As maximum weight is available is : 6

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

As maximum weight is available is : 5

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

			0	1	2	3	4	5	6	7	8
P_i	w_i	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

As maximum weight is available is : 8

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

Apply the following formula for calculating the C Table

$$c\,(i\,,j) = \max\{\,c\,(\,i-1\,,j\,), c(\,i-1\,,j - w_i) + v_i\}$$

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Step 4: Construct / print the optimal solution of 0/1 knapsack problem.

$\mathbf{0/1}\ \boldsymbol{Knapsack\ solution}(n, M)$

1. $k = M$
2. $For\ i = n\ down\ to\ 1$
3. $\quad if\ (keep[i, k] == 1)$
4. $\qquad then\ print\ i$
5. $\qquad\quad k = k - w[i]$

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	0	1	1	1	1	1	1
5	4	3	0	0	0	0	1	1	1	1	1
6	5	4	0	0	0	0	1	1	0	0	1

Keep array

Dynamic Programming

Problem 1: 0/1 Knapsack Problem

Example 1: For the given set of items and knapsack capacity of 8 kg, find the optimal solution for the 0/1 knapsack problem making use of dynamic programming approach.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	0	1	1	1	1	1	1
5	4	3	0	0	0	0	1	1	1	1	1
6	5	4	0	0	0	0	1	1	0	0	1

Keep array

Dynamic Programming

Problem 1: 0/1 Knapsack Problem (Complexity)

Time complexity of 0/1 Knapsack problem is $O(nM)$.

where, n is the number of items and M is the capacity of knapsack.

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

- The all pair shortest path problem is the problem of finding a path between two vertices or nodes in a graph such that the sum of the weights of its constituents edges is minimized.
- This problem is also known as All pair shortest path problem.
- Floyd-Warshall Algorithm is an example of dynamic programming approach.
- The advantages of Floyd-Warshall Algorithm are:
  - Easy to implement and extremely simple.

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm (Requirements)

- Graph must be weighted directed graph.
- Edge weights can be positive or negative.
- There should be no negative cycle.
  - (A negative cycle is a cycle whose edges sum give a negative value)
- This algorithm is best suited for dense graphs because, it's complexity depends on the number of vertices in the given graph

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm (Algorithm)

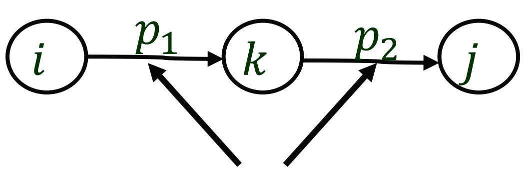
- Graph must be weighted directed graph.
- Edge weights can be positive or negative.
- There should be no negative cycle.
  - (A negative cycle is a cycle whose edges sum give a negative value)
- This algorithm is best suited for dense graphs because, it's complexity depends on the number of vertices in the given graph

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Step 1: Characterize the structure of an optimal solution

- For path $p = \langle v_1, v_2, v_3, \dots \dots, v_l \rangle$, an intermediate vertex is any vertex of p other than $v_1\ to\ v_l$.
- Let d_{ij}^{k} =shortest path weight of any path $i \leadsto j$ with intermediate vertices in $\{1,2,3, \dots \dots, k\}$.
- Consider a shortest path $i \leadsto j$ with all intermediate vertices in $\{1,2,3, \dots \dots, k\}$:
  - If k is not an intermediate vertex, then all intermediate vertices of p are $\{1,2,3, \dots \dots, k-1\}$.
  - If k is an intermediate vertex:



$all\ intermediate\ vertices\ in$
$\{1,2,3, \dots \dots, k-1\}$

Dynamic Programming

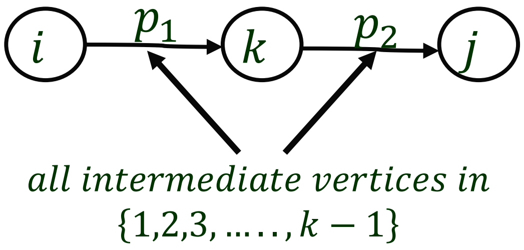
Problem 2: Floyd-Warshall Algorithm

Step 2: Recursively define the value of optimal solution.

$$d_{ij}^{k} = \begin{cases} w_{ij} & if\ k = 0 \\ \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}) & if\ k \geq 1 \end{cases}$$

Because for any path, all intermediate vertices are in the set $\{1,2,3, \ldots.., n\}$, the matrix $D^n = d_{ij}^n$ give the final answer:

$$d_{ij}^{n} = \sigma(i,j), \qquad for\ all\ i,j\ \in V$$



Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Step 3: Compute optimal solution for 0/1 knapsack problem.

$\boldsymbol{Floyd\_warshall}(w)$

1. $n = w.rows$
2. $D^0 = w$
3. $For\ k = 1\ to\ n$
4. $\quad let\ D^k = d_{ij}^k\ \ be\ an\ new\ n\ x\ n\ \ matrix$
5. $\quad For\ i = 1\ to\ n$
6. $\quad\quad For\ i = 1\ to\ n$
7. $\quad\quad\quad d_{ij}^k = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
8. $Return\ D^n$

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Step 4: Construct / print the optimal solution of 0/1 knapsack problem.

- Need to calculate predecessor matrix Π from the weight matrix D.
- Compute Π at the same time with D.
- Recursively calculate Π_{ij}^{k}
  - $\Pi_{ij}^{0} = \begin{cases} NULL & if\ i = j\ or\ w_{ij} = \infty \\ i & if\ i \neq j\ or\ w_{ij} = \infty \end{cases}$
  - $\Pi_{ij}^{k} = \begin{cases} \Pi_{ij}^{k-1} & if\ d_{ij}^{k-1} \leq d_{ik}^{k-1} + d_{kj}^{k-1} \\ \Pi_{kj}^{k-1} & if\ d_{ij}^{k-1} > d_{ik}^{k-1} + d_{kj}^{k-1} \end{cases}$

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Step 4: Construct / print the optimal solution of 0/1 knapsack problem.

$\boldsymbol{Floyd\_warshall}(w)$

1. $n = w.rows$
2. $D^0 = w$
3. $Init\_predeecessors\ (\Pi^0)$
4. $For\ k = 1\ to\ n$
5. $\quad For\ i = 1\ to\ n$
6. $\quad\quad For\ j = 1\ to\ n$
7. $\quad\quad\quad if\left(d_{ij}^{k-1} \leq d_{ik}^{k-1} + d_{kj}^{k-1}\right)$
8. $\quad\quad\quad\quad d_{ij}^{k} = d_{ij}^{k-1} \quad and \quad \Pi_{ij}^{k} = \Pi_{ij}^{k-1}$
9. $\quad\quad\quad else \quad d_{ij}^{k} = d_{ik}^{k-1} + d_{kj}^{k-1}\ and \quad \Pi_{ij}^{k} = \Pi_{kj}^{k-1}$
10. $Return\ D^n$ and Π^n

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Step 4: Construct / print the optimal solution of 0/1 knapsack problem.

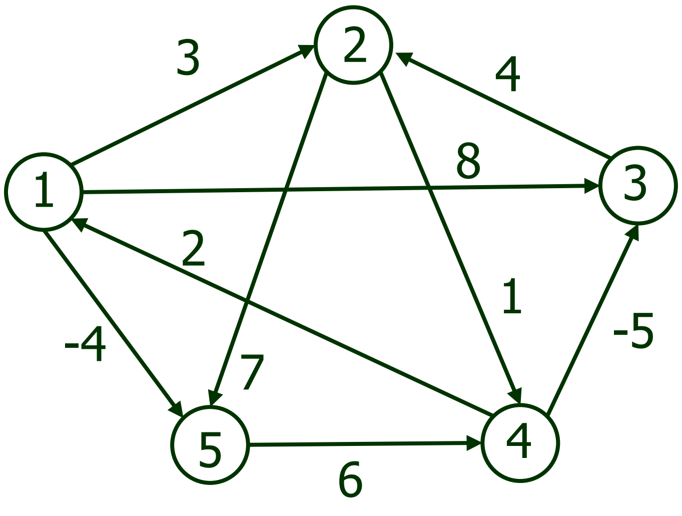
$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

1. $If\ (i = j)$
2. $\quad then\ print\ i$
3. $else\ \ if\ \Pi_{ij} = Null$
4. $\qquad then\ print$ " No path from i to J"
5. $\qquad else$
6. $\qquad Print\_all\_pairs\_shortest\_path(\Pi, i, \Pi_{ij})$
7. $\qquad print\ j$

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-



Using Floyd-Warshall Algorithm, find the shortest path distance between every pair of vertices.

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

Step-01:

Remove all the self loops and parallel edges (keeping the lowest weight edge) from the graph.

In the given graph, there are neither self edges nor parallel edges.

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

Step-02:

- Write the initial distance matrix.
- It represents the distance between every pair of vertices in the form of given weights.
- For diagonal elements (representing self-loops), distance value = 0.
- For vertices having a direct edge between them, distance value = weight of that edge.
- For vertices having no direct edge between them, distance value = ∞.

Dynamic Programming

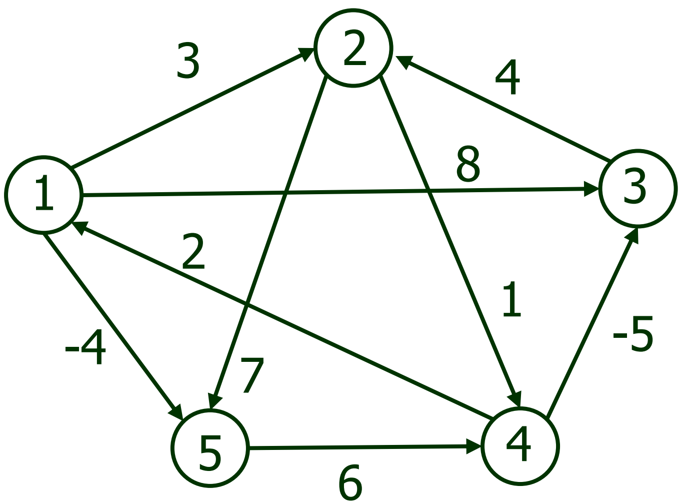
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

Step-02:

- Initial distance matrix for the given graph is-



$D^0 =$

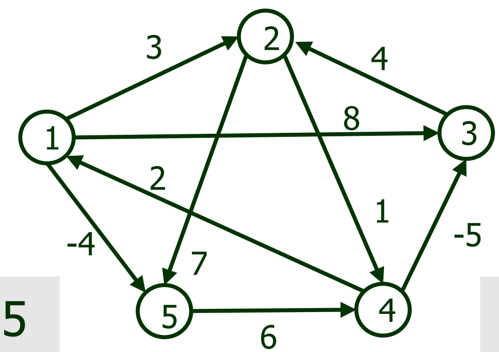
	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:



$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

Step-03:

- Using Floyd-Warshall Algorithm generate the value of $D^1, D^2\ D^3$, and D^4 martixces.
- First Generate D^1 from D^0 and Π^1 from Π^0

Dynamic Programming

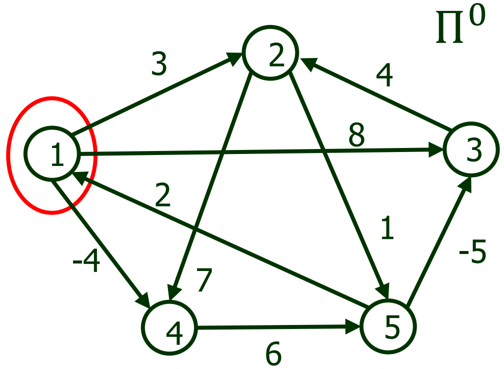
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞				
3	∞				
4	2				
5	∞				

$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N				
3	N				
4	4				
5	N				

Dynamic Programming

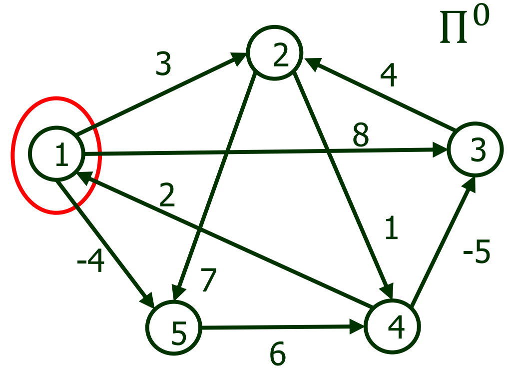
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2				
5	∞	∞	∞	6	0

$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4				
5	N	N	N	5	0

Dynamic Programming

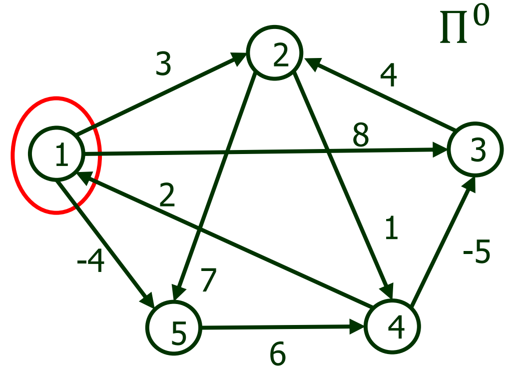
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2			0	
5	∞	∞	∞	6	0

$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4			N	
5	N	N	N	5	0

Dynamic Programming

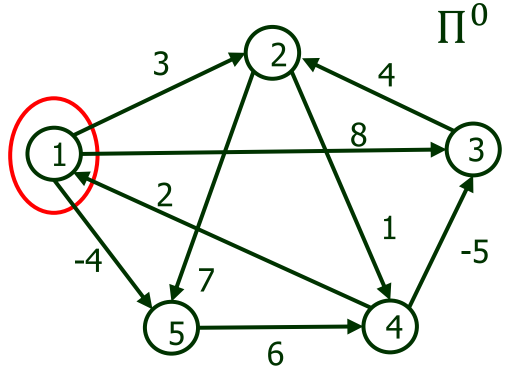
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5		0	
5	∞	∞	∞	6	0

$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1		N	
5	N	N	N	5	0

Dynamic Programming

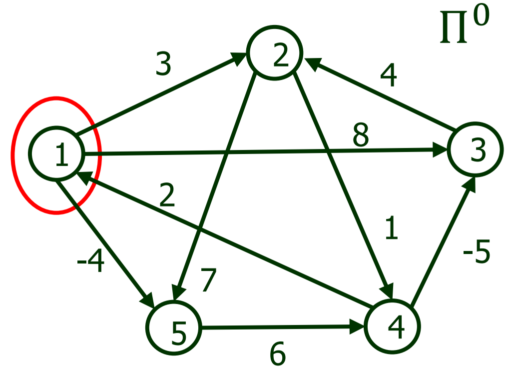
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	
5	∞	∞	∞	6	0

$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	
5	N	N	N	5	0

Dynamic Programming

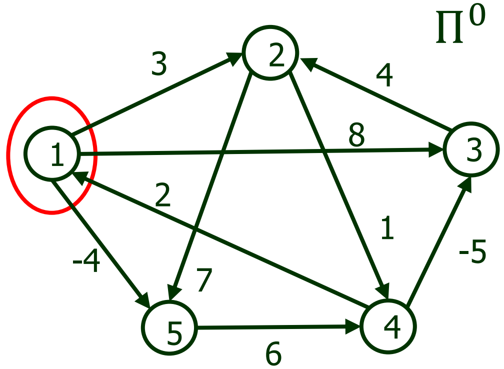
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



$\Pi^0 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	N	4	N	N
5	N	N	N	5	0

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

Dynamic Programming

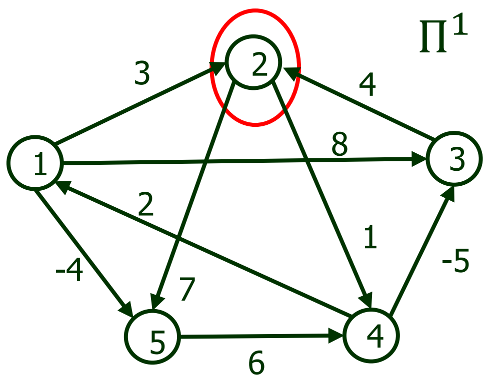
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1		3			
2	∞	0	∞	1	7
3		4			
4		5			
5		∞			

$\Pi^2 =$

	1	2	3	4	5
1		1			
2	N	N	N	2	2
3		3			
4		1			
5		N			

Dynamic Programming

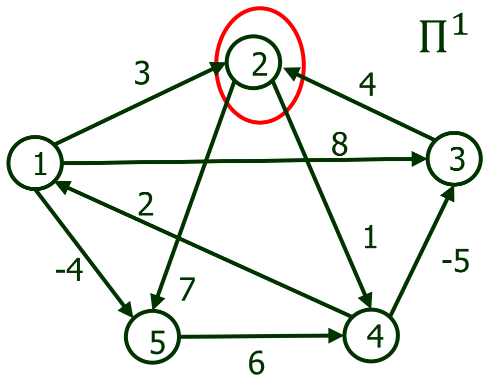
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3			
2	∞	0	∞	1	7
3		4	0		
4		5		0	
5		∞			0

$\Pi^2 =$

	1	2	3	4	5
1	N	1			
2	N	N	N	2	2
3		3	N		
4		1		N	
5		N			N

Dynamic Programming

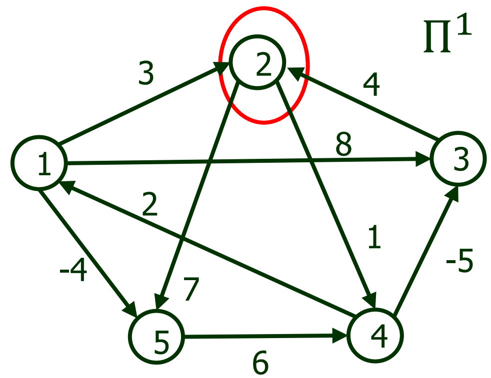
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3	8		
2	∞	0	∞	1	7
3	∞	4	0		
4	2	5	-5	0	
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1		
2	N	N	N	2	2
3	N	3	N		
4	4	1	4	N	
5	N	N	N	5	N

Dynamic Programming

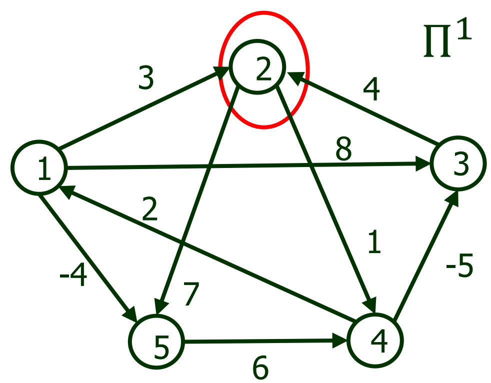
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	
2	∞	0	∞	1	7
3	∞	4	0		
4	2	5	-5	0	
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	
2	N	N	N	2	2
3	N	3	N		
4	4	1	4	N	
5	N	N	N	5	N

Dynamic Programming

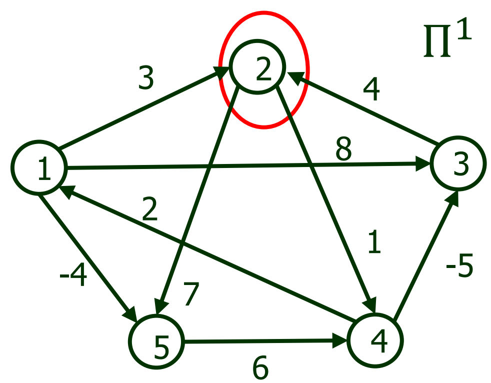
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0		
4	2	5	-5	0	
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N		
4	4	1	4	N	
5	N	N	N	5	N

Dynamic Programming

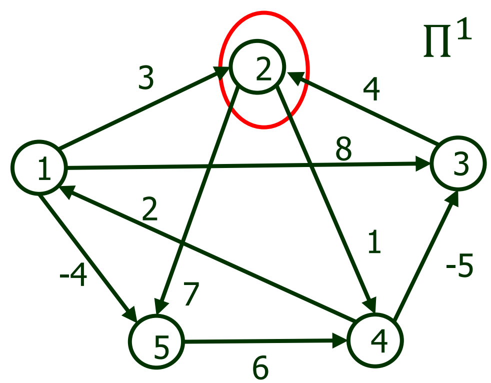
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	
4	2	5	-5	0	
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	
4	4	1	4	N	
5	N	N	N	5	N

Dynamic Programming

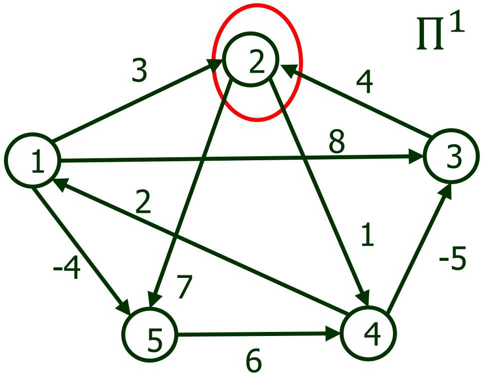
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	
5	N	N	N	5	N

Dynamic Programming

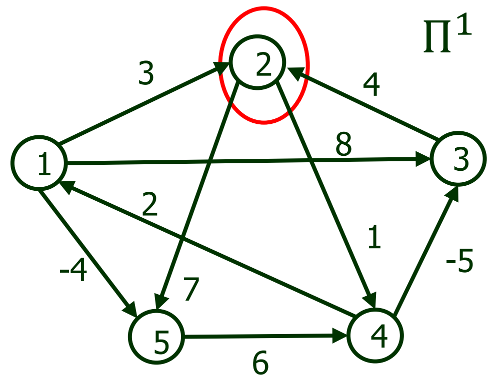
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^1 =$

	1	2	3	4	5
1	N	1	1	N	1
2	N	N	N	2	2
3	N	3	N	N	N
4	4	1	4	N	1
5	N	N	N	5	0

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

Dynamic Programming

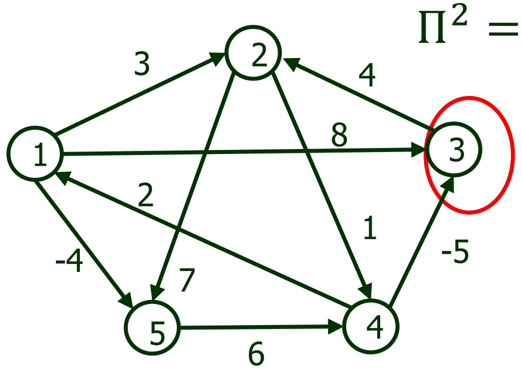
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1			8		
2			∞		
3	∞	4	0	5	11
4			-5		
5			∞		

$\Pi^3 =$

	1	2	3	4	5
1			1		
2			N		
3	N	3	N	2	2
4			4		
5			N		

Dynamic Programming

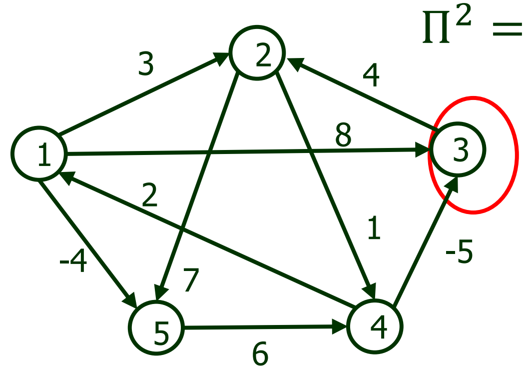
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1	0		8		
2		0	∞		
3	∞	4	0	5	11
4			-5	0	
5			∞		0

$\Pi^3 =$

	1	2	3	4	5
1	N		1		
2		N	N		
3	N	3	N	2	2
4			4	N	
5			N		N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

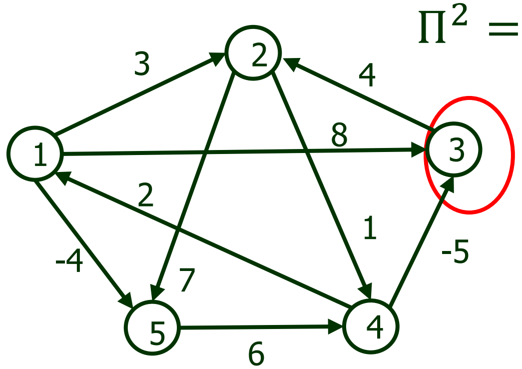
Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N



$D^3 =$

	1	2	3	4	5
1	0		8		
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2		-5	0	
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N		1		
2	N	N	N	2	2
3	N	3	N	2	2
4	4		4	N	
5	N	N	N	5	N

Dynamic Programming

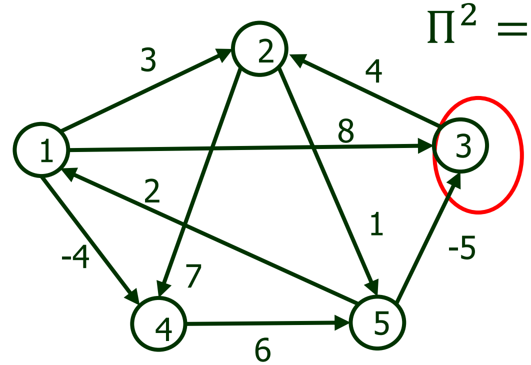
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1	0	3	8		
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2		-5	0	
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1		
2	N	N	N	2	2
3	N	3	N	2	2
4	4		4	N	
5	N	N	N	5	N

Dynamic Programming

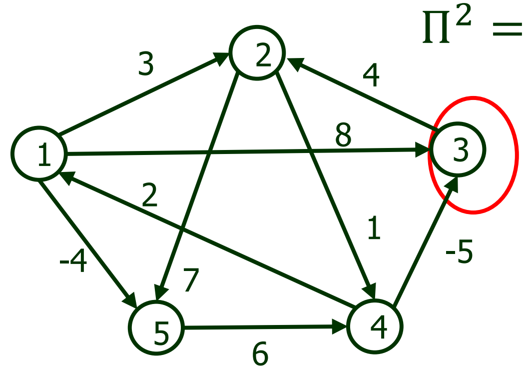
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2		-5	0	
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	
2	N	N	N	2	2
3	N	3	N	2	2
4	4		4	N	
5	N	N	N	5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

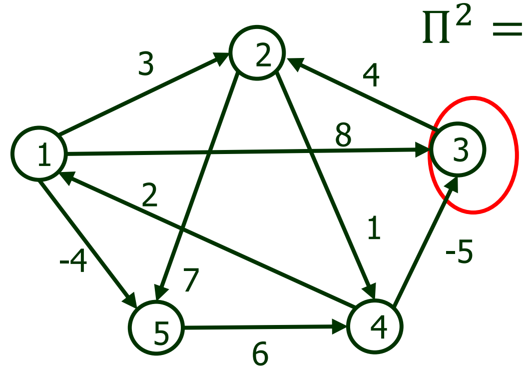
Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N



$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2		-5	0	
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4		4	N	
5	N	N	N	5	N

Dynamic Programming

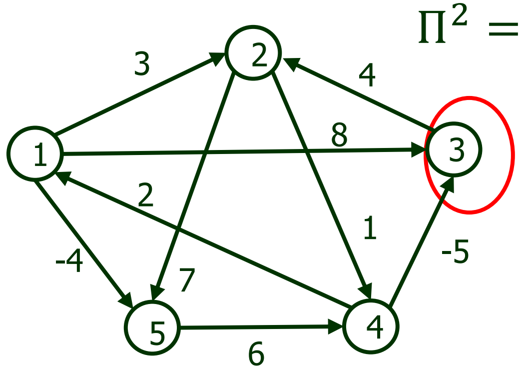
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	
5	N	N	N	5	N

Dynamic Programming

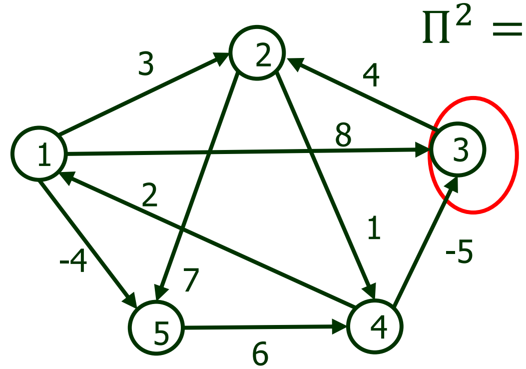
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^2 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	1	4	N	1
5	N	N	N	5	N

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

Dynamic Programming

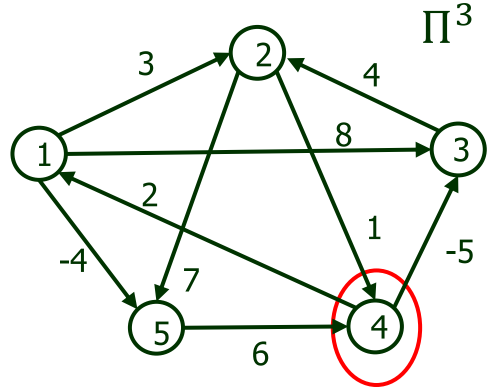
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1				4	
2				1	
3				5	
4	2	-1	-5	0	-2
5				6	

$\Pi^4 =$

	1	2	3	4	5
1				2	
2				2	
3				2	
4	4	3	4	N	1
5				5	

Dynamic Programming

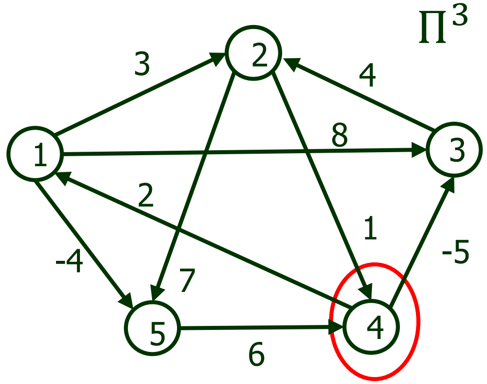
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0			4	
2		0		1	
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N			2	
2		N		2	
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

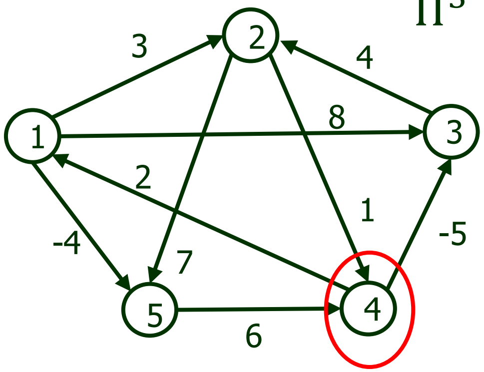
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3		4	
2		0		1	
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1		2	
2		N		2	
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

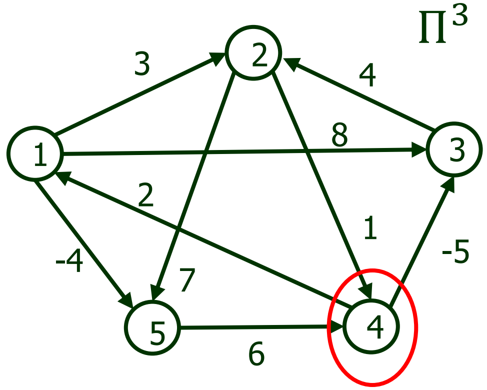
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	
2		0		1	
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	
2		N		2	
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

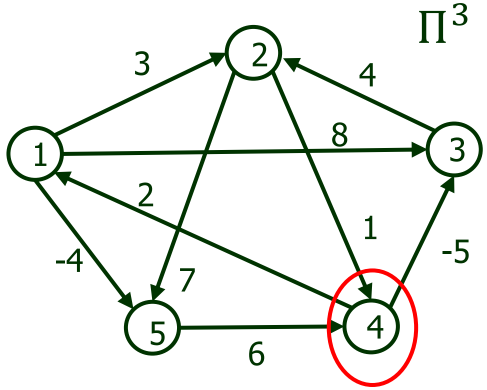
Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N



$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2		0		1	
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2		N		2	
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

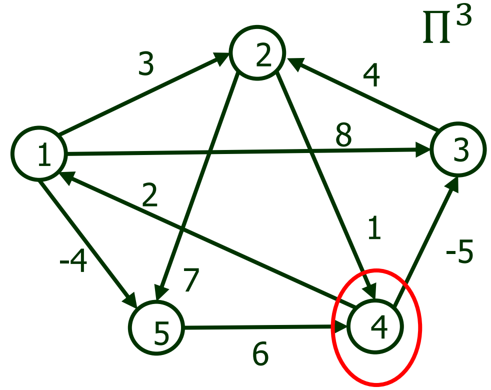
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0		1	
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N		2	
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

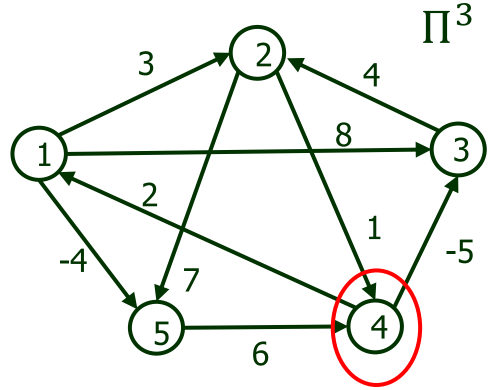
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

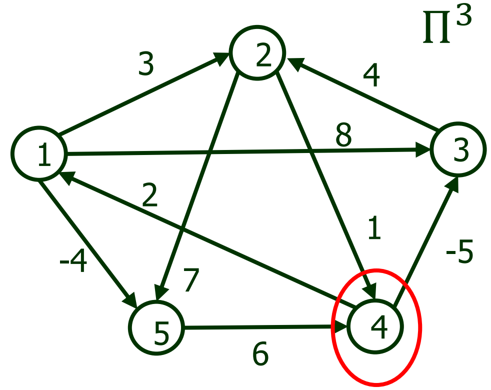
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3			0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3			N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

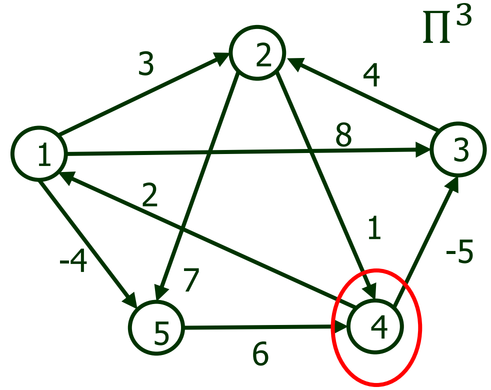
Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N



$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7		0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4		N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

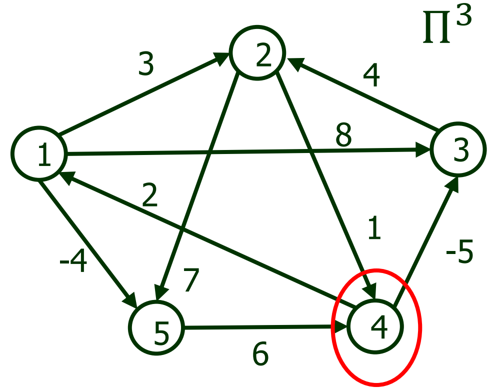
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4	3	N	2	
4	4	3	4	N	1
5				5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

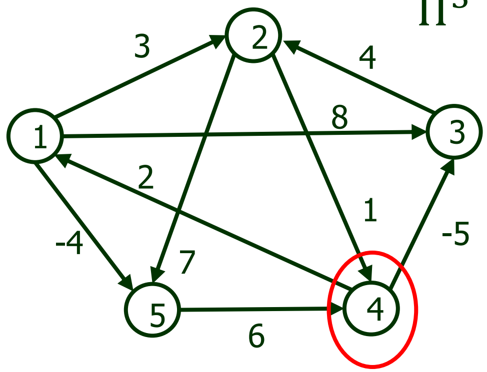
Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N



$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5				6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4	3	N	2	1
4	4	3	4	N	1
5				5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

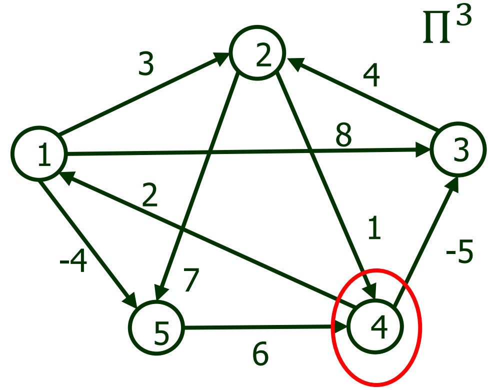
Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N



$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8			6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4	3	N	2	1
4	4	3	4	N	1
5	4			5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

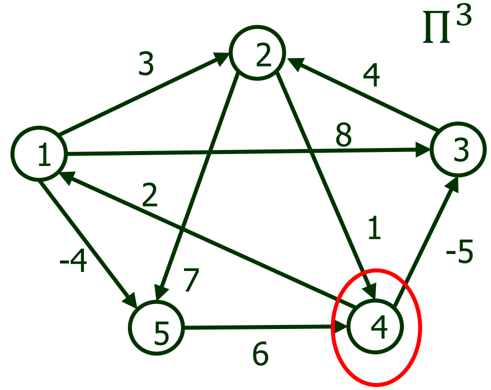
Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0

$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N



$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5		6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3		5	N

Dynamic Programming

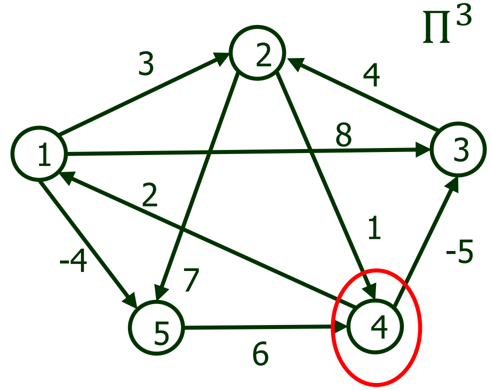
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^3 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	-1	-5	0	-2
5	∞	∞	∞	6	0



$\Pi^3 =$

	1	2	3	4	5
1	N	1	1	2	1
2	N	N	N	2	2
3	N	3	N	2	2
4	4	3	4	N	1
5	N	N	N	5	N

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

Dynamic Programming

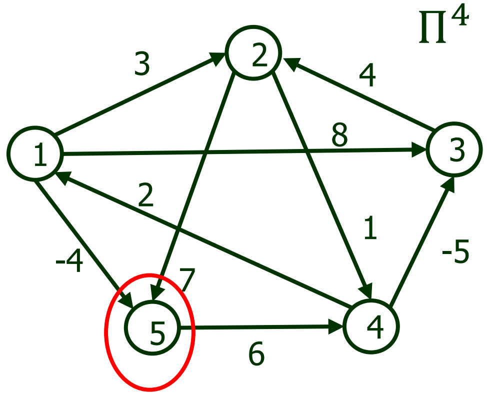
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1					-4
2					-1
3					3
4					-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1					1
2					4
3					1
4					1
5	4	3	4	5	N

Dynamic Programming

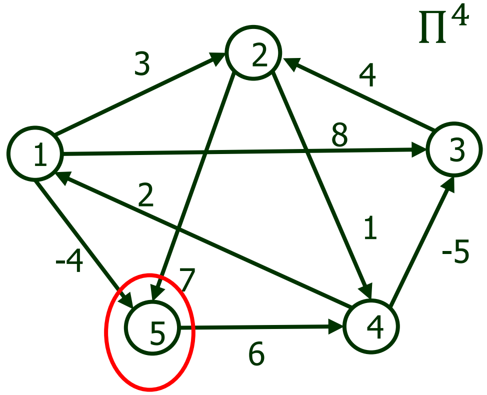
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	4
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0				-4
2		0			-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N				1
2		N			4
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

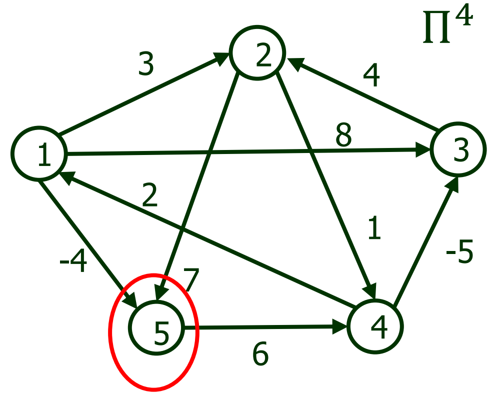
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1			-4
2		0			-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3			1
2		N			1
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

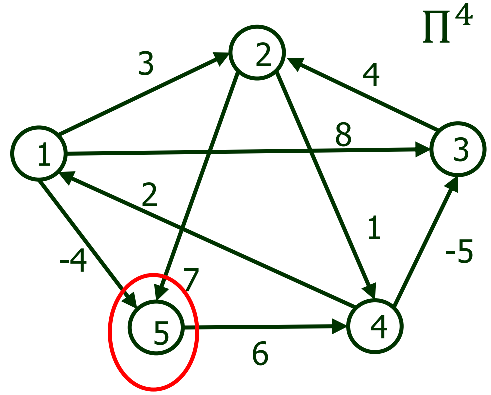
Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N



$D^5 =$

	1	2	3	4	5
1	0	1	-3		-4
2		0			-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4		1
2		N			1
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

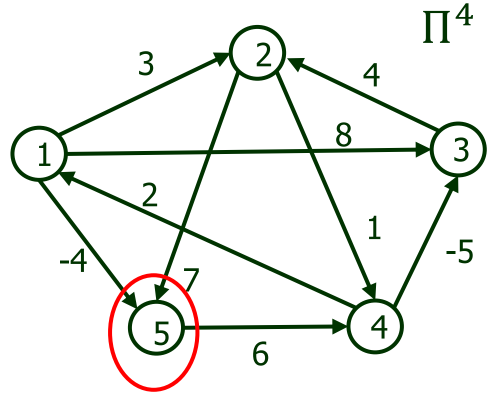
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2		0			-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2		N			1
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

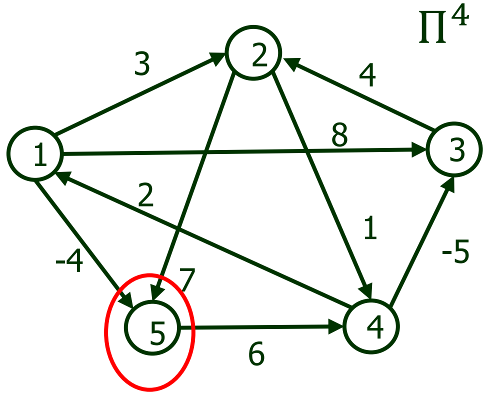
Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N



$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0			-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N			1
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

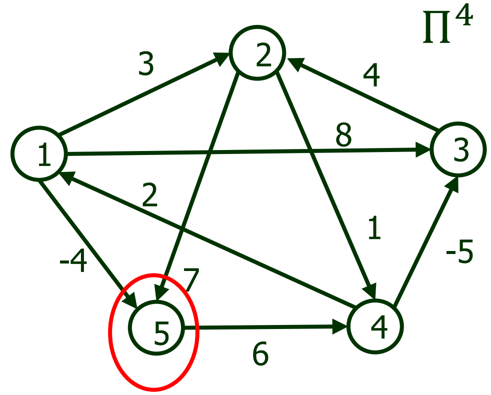
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4		-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4		1
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

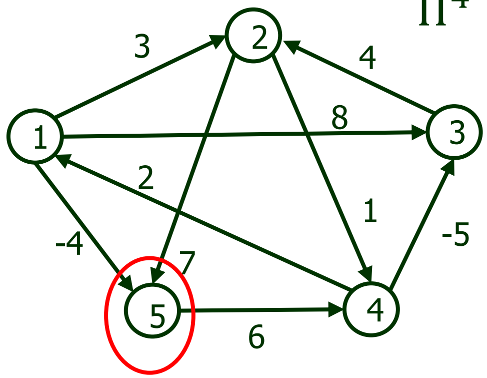
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3			0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3			N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

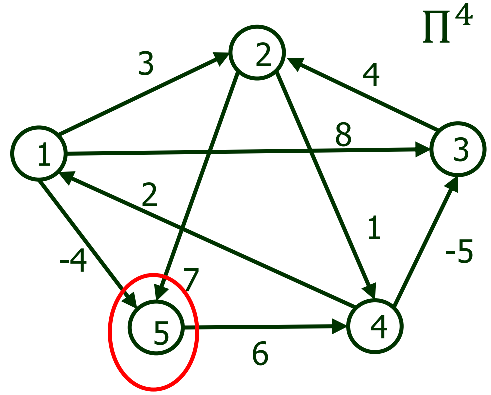
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7		0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4		N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

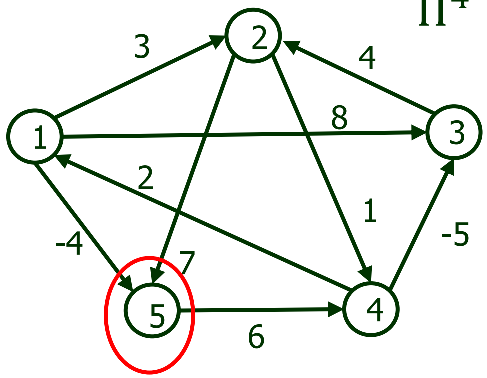
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0		3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N		1
4				N	1
5	4	3	4	5	N

Dynamic Programming

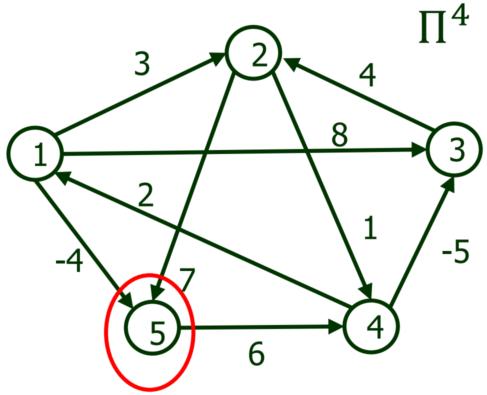
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4				0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4				N	1
5	4	3	4	5	N

Dynamic Programming

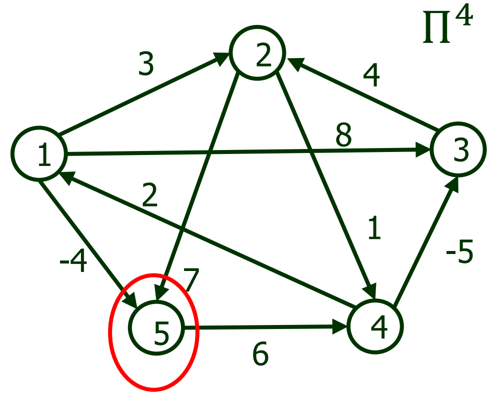
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2			0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4			N	1
5	4	3	4	5	N

Dynamic Programming

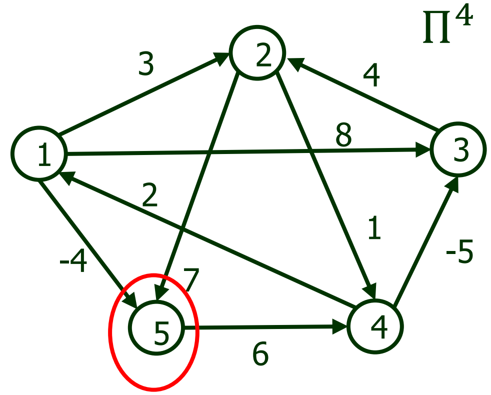
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1		0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3		N	1
5	4	3	4	5	N

Dynamic Programming

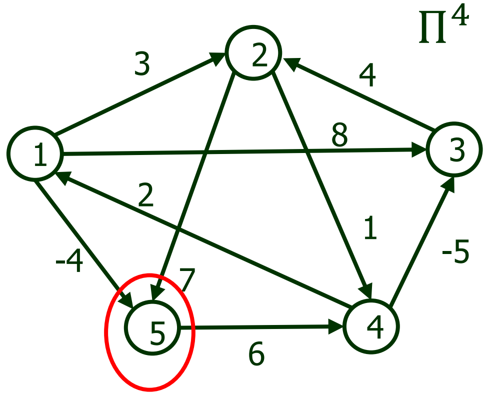
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^4 =$

	1	2	3	4	5
1	0	3	-1	4	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^4 =$

	1	2	3	4	5
1	N	1	4	2	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\Pi^5 =$

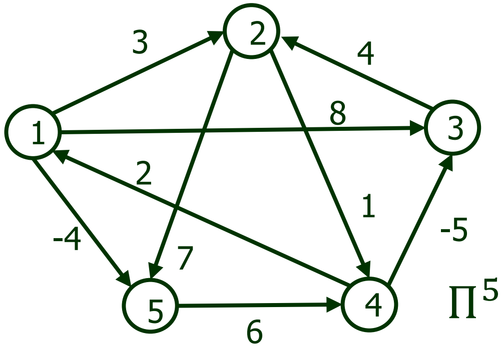
	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:



$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.E $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

Dynamic Programming

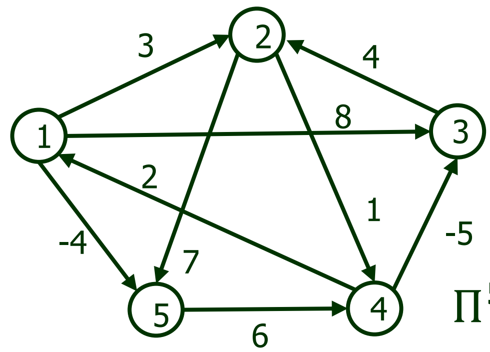
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3)}$

Dynamic Programming

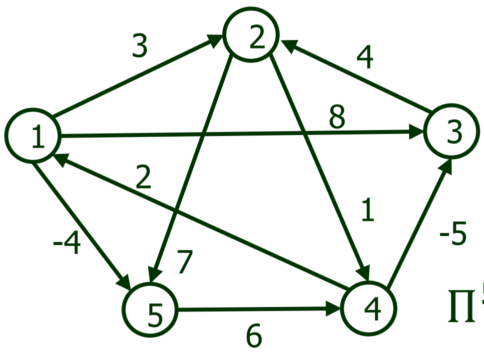
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4)}$

Dynamic Programming

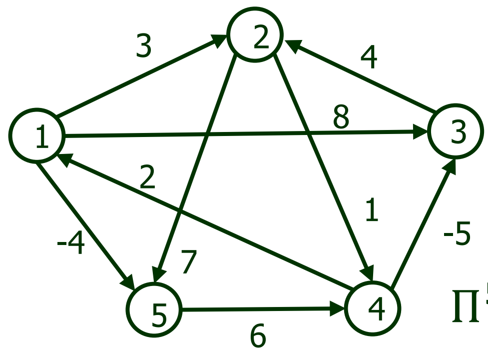
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 5)}$

Dynamic Programming

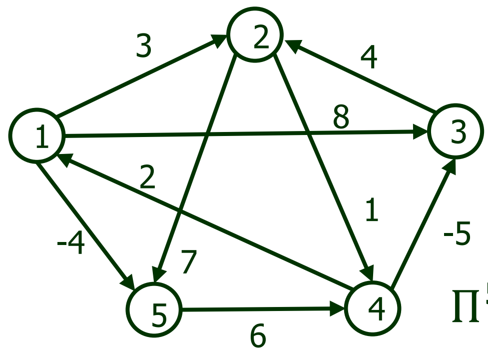
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\hookrightarrow \boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3)}$

$\hookrightarrow \boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4)}$

$\hookrightarrow \boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 5)}$

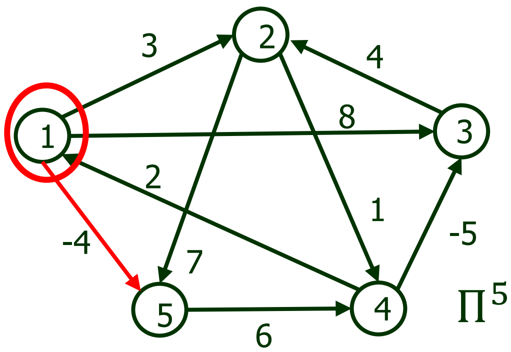
$\hookrightarrow \boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 1)}$

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:



$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0

$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path}(\boldsymbol{\Pi}, \boldsymbol{i}, \boldsymbol{j})$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path}(\boldsymbol{\Pi}, \mathbf{1}, \mathbf{2})$

$\boldsymbol{Print\_all\_pairs\_shortest\_path}(\boldsymbol{\Pi}, \mathbf{1}, \mathbf{3})$

$\boldsymbol{Print\_all\_pairs\_shortest\_path}(\boldsymbol{\Pi}, \mathbf{1}, \mathbf{4})$

$\boldsymbol{Print\_all\_pairs\_shortest\_path}(\boldsymbol{\Pi}, \mathbf{1}, \mathbf{5})$

$\boldsymbol{Print\_all\_pairs\_shortest\_path}(\boldsymbol{\Pi}, \mathbf{1}, \mathbf{1}) \Longrightarrow \mathbf{1}$

Dynamic Programming

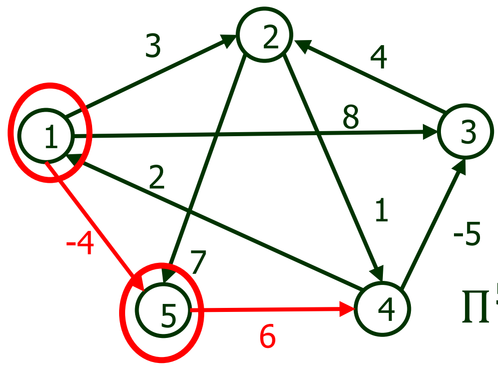
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 5) \Longrightarrow 5}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 1) \Longrightarrow 1}$

Dynamic Programming

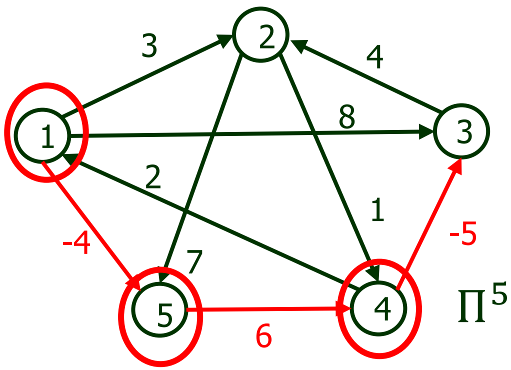
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4) \Longrightarrow 4}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 5) \Longrightarrow 5}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 1) \Longrightarrow 1}$

Dynamic Programming

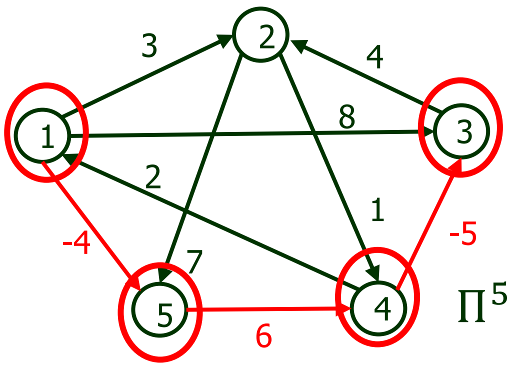
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2)}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3) \Rightarrow 3}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4) \Rightarrow 4}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 5) \Rightarrow 5}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 1) \Rightarrow 1}$

Dynamic Programming

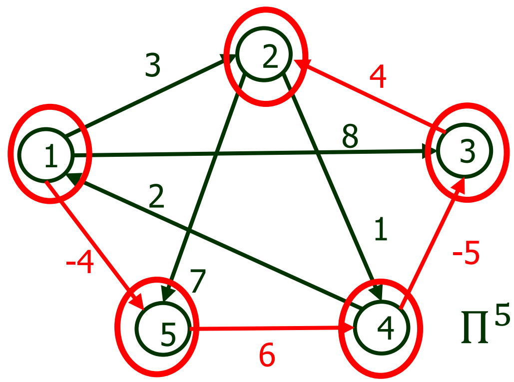
Problem 2: Floyd-Warshall Algorithm

Example 1: Consider the following directed weighted graph-

Solution:

$D^5 =$

	1	2	3	4	5
1	0	1	-3	2	-4
2	3	0	-4	1	-1
3	7	4	0	5	3
4	2	-1	-5	0	-2
5	8	5	1	6	0



$\Pi^5 =$

	1	2	3	4	5
1	N	3	4	5	1
2	4	N	4	2	1
3	4	3	N	2	1
4	4	3	4	N	1
5	4	3	4	5	N

For printing Shortest path from 1 to 2 use

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, i, j)}$

i.e $\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 2) \Longrightarrow 2}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 3) \Longrightarrow 3}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 4) \Longrightarrow 4}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 5) \Longrightarrow 5}$

$\boldsymbol{Print\_all\_pairs\_shortest\_path(\Pi, 1, 1) \Longrightarrow 1}$

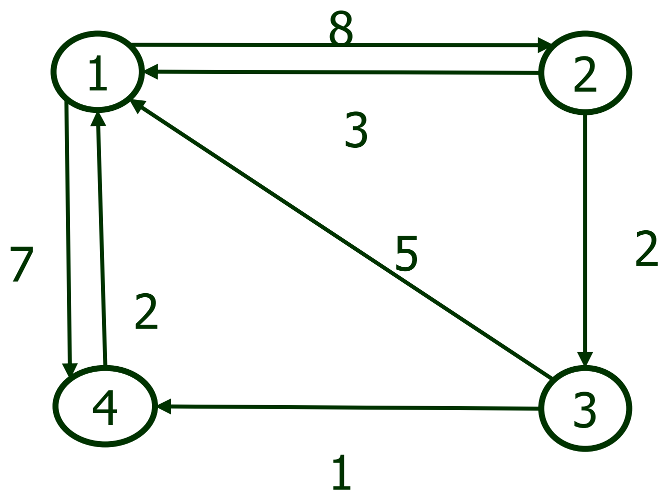
Dynamic Programming

Problem 2: Floyd-Warshall Algorithm (Analysis)

1. Floyd-Warshall Algorithm consists of three loops over all the nodes.
2. The inner most loop consists of only constant complexity operations.
3. Hence, the asymptotic complexity of Floyd Warshall algorithm is $O(n^3)$.
4. Here, n is the number of nodes in the given graph.

Dynamic Programming

Problem 2: Floyd-Warshall Algorithm (Home Assignment)



Thank u