**23MAT116**

**Discrete Mathematics**

**Lab Report**



**Department computer science and engineering**

**Amrita school of computing**

**Amrita vishwa vidyapeetham, Amaravati campus**

**Name:** K.v.durga prasad

**Class/Sec:** CSE-B **Verified by Roll no:** AV.SC.U4CSE24140

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. no** | **Questions** | **Page no** | **Date** | **Signature** |
| 1 | Write Matlab code to generate a truth table that consists of 3 statements: p, q, r . | **3** |  |  |
| 2 | Write recursive program for Fibonacci series in Matlab. | **4** |  |  |
| 3 | Implement the binary search as a recursive function in Matlab. | **5** |  |  |
| 4 | Write a Matlab program for permutation and combinations. Apply this implementation to the following problem. | **6** |  |  |
| 5 | Write a Matlab program to compute fn for n = 1,2,….,10. The recurrence for this question is f(0) = 25, fn = f(n-1) + 7 – 7(n+1)/n , n>= 2. | **7** |  |  |
| 6 | Create a directed graph using an edge list, and then find the equivalent adjacency matrix representation of the graph. | **8** |  |  |
| 7 | Create a graph using an edge list, and then calculate the graph incidence matrix | **9** |  |  |
| 8 | Create a directed graph using an edge list, and then calculate the incidence matrix. | **10 to 11** |  |  |
| 9 | Create and plot a graph, and then find the degree of each node. | **12** |  |  |
| 10 | Create and plot a directed graph. Calculate the shortest path between nodes | **13 to 14** |  |  |
| 11 | Create and plot a graph with weighted edges. Find the shortest path between nodes, and specify two outputs to also return the length of the graph. | **15 to 16** |  |  |

**QUESTION-1:** Write Matlab code to generate a truth table that consists of 3 statements: p, q, r .

**CODE:**

truthvalues = dec2bin(0:7) - '0';

truthvalues = flip(truthvalues,2);

p = truthvalues(:,1);

q = truthvalues(:,2);

r = truthvalues(:,3);

disp(' p q r');

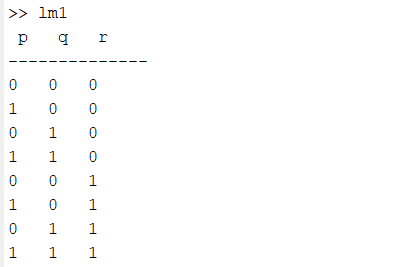
disp('--------------');

for i = 1:length(p)

fprintf('%d %d %d\n',p(i), q(i), r(i));

end

**OUTPUT:**

****

**QUESTION-2:** Write recursive program for Fibonacci series in Matlab.

**CODE:**

function f = fibonacci(n)

if n == 0

f = 0;

elseif n == 1

f = 1;

else

f = fibonacci(n-1) + fibonacci(n-2);

end

end

n = 6;

fprintf('Fibonacci sries up to %d terms:\n',n);

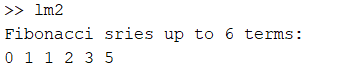
for i =0:n-1

fprintf('%d ', fibonacci(i));

end

fprintf('\n');

**OUTPUT:**

****

**QUESTION-3:** Implement the binary search as a recursive function in Matlab.

**CODE:**

function index = binarySearch(arr, target, low, high)

if low > high

index =-1; % Target not found

return;

end

mid = floor((low + high)/2);

if arr(mid)== target

index = mid; % Target found

elseif target < arr(mid)

index = binarySearch(arr, target, low, mid-1);

else

index = binarySearch(arr, target, mid+1, high);

end

end

arr = [1,3,5,7,9,11,13,15]; %Sorted array

target = 7;

index = binarySearch(arr, target, 1, length(arr));

if index ~= -1

fprintf('Element %d found at index %d.\n', target, index);

else

fprintf('Element %d not found in the array.\n', target);

end

**OUTPUT:**

****

**QUESTION-4:** Write a Matlab program for permutation and combinations. Apply this implementation to the following problem.

How many ways are there to select five players from a 10- member tennis team to make a trip to a match another school?

**CODE:**

function [perm, comb] = perm\_comb(n, r)

if r > n || n < 0 || r < 0

error('Invalid input: Ensure that 0 ≤ r ≤ n');

end

perm = factorial(n) / factorial(n - r);

comb = factorial(n) / (factorial(r) \* factorial(n - r));

end

%inputs

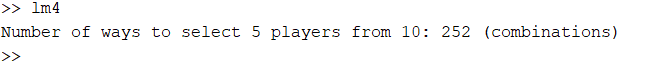
n =10; %total players

r = 5; % palyers to choose

[permutation, combination] = perm\_comb(n,r);

fprintf('Number of ways to select %d players from %d: %d (combinations)\n', r, n, combination);

**OUTPUT:**

****

**QUESTION-5:** Write a Matlab program to compute fn for n = 1,2,….,10. The recurrence for this question is f(0) = 25, fn = f(n-1) + 7 – 7(n+1)/n , n>= 2.

**CODE:**

f = zeros(1,11);

f(1) = 25;

f(2) = f(1) + 7 - (7 \* (2)) / 1;

for n = 3:11

f(n) = f(n-1) + 7 - (7\* (n)) / (n - 1);

end

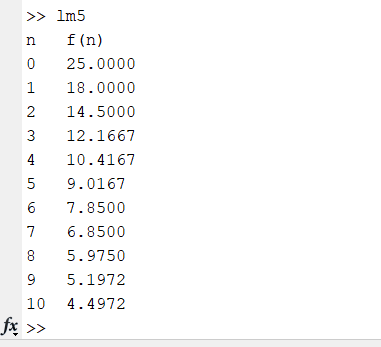
fprintf('n\tf(n)\n');

for n = 0:10

fprintf('%d\t%.4f\n',n,f(n+1));

end

**OUTPUT:**



**QUESTION-6:** Create a directed graph using an edge list, and then find the equivalent adjacency matrix representation of the graph.

**CODE:**

edgeList = [1 2;1 3;2 4;3 4;4 5;5 1];

G = digraph(edgeList(:,1), edgeList(:,2));

figure;

plot(G);

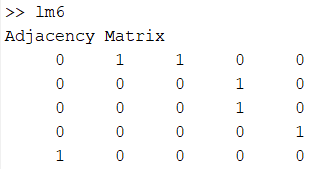
title('Directed graph');

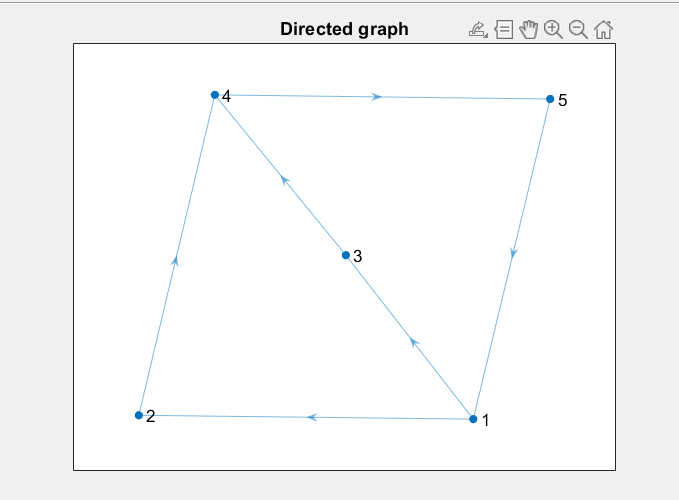
A = adjacency(G);

disp('Adjacency Matrix');

disp(full(A));

**OUTPUT:**





**QUESTION-7:** Create a graph using an edge list, and then calculate the graph incidence matrix

**CODE:**

edgeList = [1 2;1 3;2 4;3 4;4 5;5 1];

G = digraph(edgeList(:,1), edgeList(:,2));

figure;

plot(G);

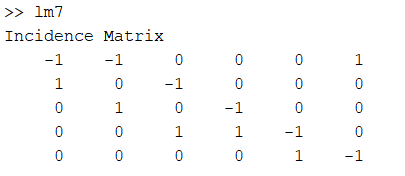
title('Unidirected graph');

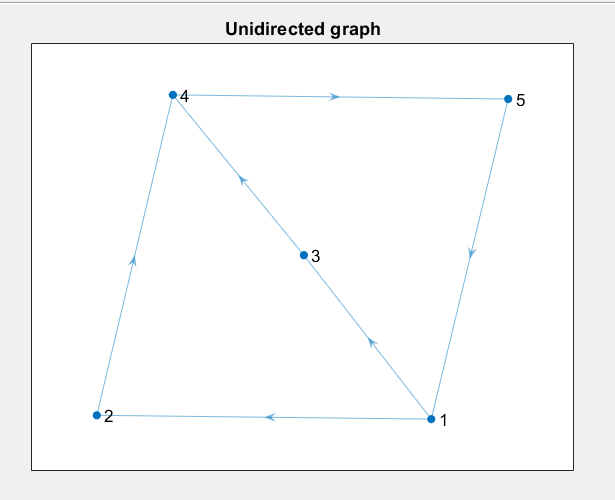
I = incidence(G);

disp('Incidence Matrix');

disp(full(I));

**OUTPUT:**





**QUESTION-8:** Create a directed graph using an edge list, and then calculate the incidence matrix.

**CODE:**

edgeList = [1 2;1 3;2 4;3 4;4 5;5 1];

G = digraph(edgeList(:,1), edgeList(:,2));

figure;

plot(G, 'Layout', 'force', 'EdgeColor','b','MarkerSize',8,'NodeColor','r');

title('Directed Graph');

num\_nodes = numnodes(G);

num\_edges = numedges(G);

incidence\_matrix = zeros(num\_nodes, num\_edges);

edges = G.Edges;

for i =1:num\_edges

source = edges.EndNodes(i,1);

target = edges.EndNodes(i,2);

incidence\_matrix(source,i) = -1;

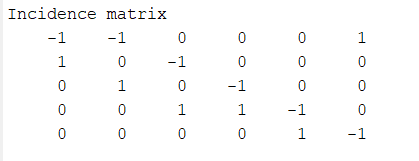
incidence\_matrix(target,i) = 1;

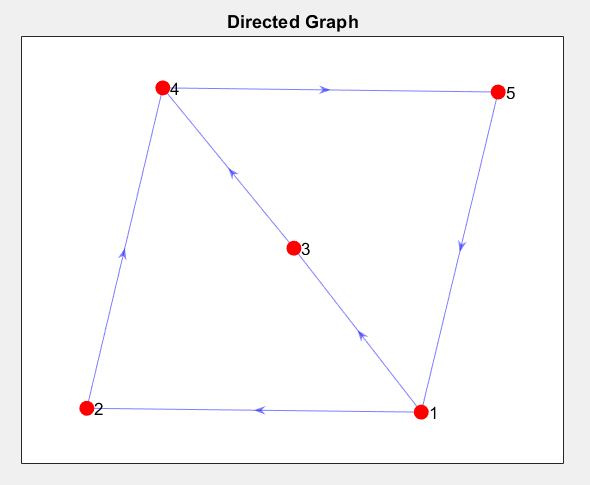
end

disp('Incidence matrix');

disp(incidence\_matrix);

**OUTPUT:**





**QUESTION-9:** Create and plot a graph, and then find the degree of each node.

**CODE:**

edgeList = [1 2;1 3;2 4;3 4;4 5;5 1];

G = graph(edgeList(:,1), edgeList(:,2));

figure;

plot(G, 'LineWidth', 1.5, 'NodeFontSize', 12);

title('Undirected Graph');

deg = degree(G);

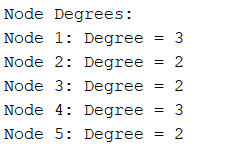
disp('Node Degrees:');

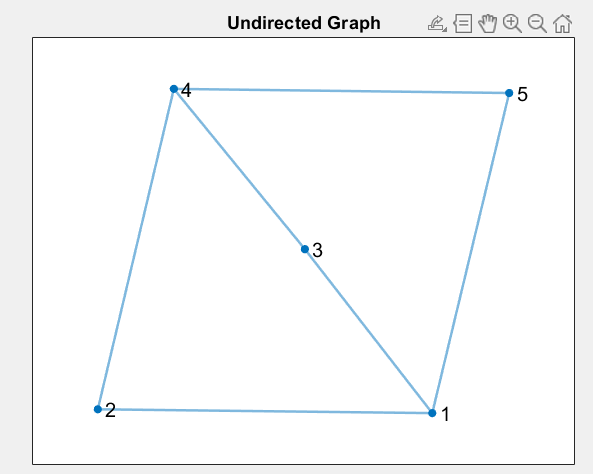
for i = 1:numnodes(G)

fprintf('Node %d: Degree = %d\n', i, deg(i));

end

**OUTPUT:**





**QUESTION-10:**  Create and plot a directed graph. Calculate the shortest path between nodes

**CODE:**

**% Define nodes**

**nodes = {'A', 'B', 'C', 'D', 'E'};**

**% Define edges and their weights (directed edges)**

**s = {'A', 'A', 'B', 'C', 'D'}; % source nodes**

**t = {'B', 'C', 'D', 'E', 'E'}; % target nodes**

**weights = [2, 5, 1, 2, 3]; % corresponding weights**

**% Create the directed graph**

**G = digraph(s, t, weights, nodes);**

**% Plot the graph**

**figure;**

**plot(G, 'EdgeLabel', G.Edges.Weight);**

**title('Directed Graph');**

**% Calculate shortest path from A to E**

**[startNode, endNode] = deal('A', 'E');**

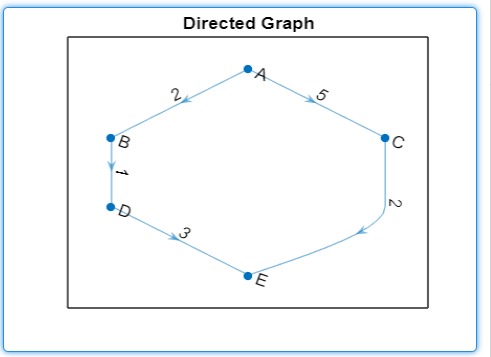
**[shortestPath, totalCost] = shortestpath(G, startNode, endNode);**

**% Display result**

**disp(['Shortest path from ', startNode, ' to ', endNode, ': ', strjoin(shortestPath, ' -> ')]);**

**disp(['Total path weight: ', num2str(totalCost)]);**

**OUTPUT:**



**QUESTION-11:** Create and plot a graph with weighted edges. Find the shortest path between nodes, and specify two outputs to also return the length of the graph.

**CODE:**

**% Define the graph**

**nodes = 8; % Number of nodes**

**edges = [1 2 1 3 2 4 3 5 4 6 5 7 6 8 7 8]; % Edge connections**

**weights = [5 2 3 4 1 7 2 9]; % Edge weights**

**G = graph(edges(1:2:end), edges(2:2:end), weights); % Create graph object**

**% Plot the graph**

**figure;**

**plot(G);**

**title('Weighted Graph');**

**% Calculate shortest path and length**

**[shortestPath, pathLength] = shortestpath(G, 1, 8);**

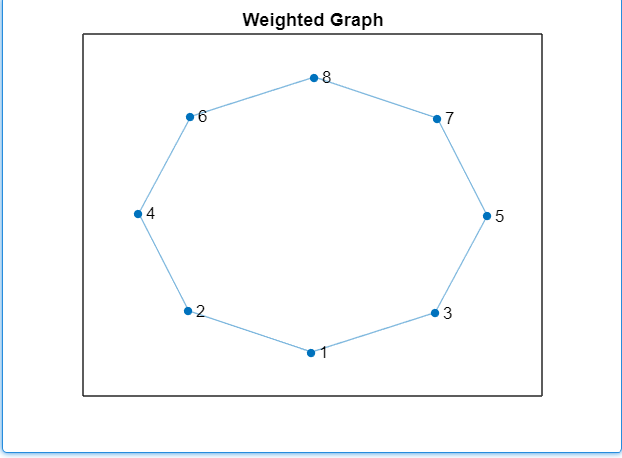
**disp(['Shortest Path: ', num2str(shortestPath)]);**

**disp(['Path Length: ', num2str(pathLength)]);**

**% Highlight the shortest path on the plot (Optional)**

**highlight(G, shortestPath, 'NodeColor', 'red', 'EdgeColor', 'red', 'LineWidth', 3);**

**OUTPUT:**

****