Exam Eligibility Assignment

Performance summary

✓ Assessed

Success status	Score	Attempts
Passed	96 of 100 points	1

Results

Course	**Network Theory and Dynamic Systems (SS 2025)** ID: 4669112833 / 111886790238809
Test	**Exam Eligibility Assignment** ID: 4933452135

This are your test results

Duration	2h 7m 58s 6/5/2025, 11:36 AM - 6/9/2025, 12:22 PM
Answered	10 of 10 questions (100%)
Your score	96 of 100 points (96%)

96%
Score: 96

1. Knowledge Tasks (20 Points) 1	1.2. Single Choice Questions (5 Points) 5	2. Practical Tasks (40 Points) 2	3. Programming Tasks (40 Points) 2
100% Score: 15 of 15	100% Score: 5 of 5	90% Score: 36 of 40	100% Score: 40 of 40
go to section ›	go to section ›	go to section ›	go to section ›

1. Knowledge Tasks (20 Points) 15 of 15 points (100%)

1.1. True/false (15 Points)

Status	Answered	
Your score	15 / 15	100%

Response

- Each correct answer gives **1** point.
- Each wrong answer deducts **1** point.
- Unanswered questions give **0** points.
- The maximum number of points is 15, and the minimum is 0.

Unanswered	Right	Wrong	
☐	☐	☑	In a directed network, the adjacency matrix is always symmetric.
☐	☑	☐	A bipartite network can never contain triangles in its structure.
☐	☐	☑	In a weighted network, the degree of a node is equal to the sum of the weights of its links.

☐	☑	☐	Removing a high-degree node (hub) from a scale-free network has a greater impact on connectivity than removing a random node.
☐	☑	☐	In a simple undirected graph, the sum of all node degrees is equal to twice the number of edges.
☐	☐	☑	The density of a network remains constant as the number of nodes increases, assuming the number of links grows linearly.
☐	☑	☐	The density of a complete undirected network is always 1.
☐	☑	☐	A node's clustering coefficient is always zero if it has less than two neighbors.
☐	☑	☐	A network with low average path length but also low clustering is more likely to resemble a random network than a small-world network.
☐	☐	☑	The average path length is always equal to the diameter of the network divided by two.
☐	☐	☑	In real networks, all nodes with high betweenness centrality also have high degree centrality.
☐	☑	☐	Increasing k in k-core decomposition yields a smaller, denser subnetwork of nodes with degree ≥ k.
☐	☑	☐	The robustness of a network can be assessed by tracking the size of the largest connected component as nodes are removed.
☐	☐	☑	The World Wide Web can be considered a strongly connected component.
☐	☑	☐	The damping factor in PageRank prevents rank sinks and guarantees convergence.

▸ Solution

‹ go back to overview

1.2. Single Choice Questions (5 Points) 5 of 5 points (100%)

- Each correct answer gives **1 point**.
- Each wrong answer **deducts 1 point**.

Q1 (1 Point)

Status	Answered		
Your score	1 / 1		100%

Response

Given a complete network A with N nodes, and a bipartite network B also with N nodes, which of the following holds true for any N>2:

- ○ Network A has the same number of links as network B
- ⊙ Network A has more links than network B
- ○ None of these hold true for all such N>2
- ○ Network A has fewer links than network B

▸ Solution

Q2 (1 Point)

Status	Answered		
Your score	1 / 1		100%

Response

The Watts–Strogatz model is useful in capturing which property of real-world social networks **not** found in Erdős– Rényi random graphs?

- ○ Short average path lengths
- ⊙ High clustering coefficient
- ○ Long average path lengths
- ○ Low clustering coefficient

▸ Solution

⦿ Q3 (1 Point)

Status	Answered		
Your score	1 / 1		100%

Response

In the context of social networks, what is the primary risk associated with high homophily?

- ✓ Formation of echo chambers and polarization
- ○ Decreased average degree
- ○ Increased network diameter
- ○ Randomness in connections

▸ Solution

⦿ Q4 (1 Point)

Status	Answered		
Your score	1 / 1		100%

Response

Which scenario would most likely increase the risk of rapid misinformation spread in a social network, based on homophily and assortativity principles?

- ○ Low homophily, high degree assortativity
- ○ Low homophily, low assortativity
- ○ Random connections
- ✓ High homophily, high assortativity

▸ Solution

⦿ Q5 (1 Point)

Status	Answered		
Your score	1 / 1		100%

Response

Which of the following best characterizes the degree distribution in random (ER) networks?

- ✓ Binomial
- ○ Power-law
- ○ Heavy-tailed
- ○ Exponential

▸ Solution

‹ go back to overview

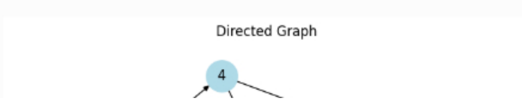
2. Practical Tasks (40 Points) 36 of 40 points (90%)

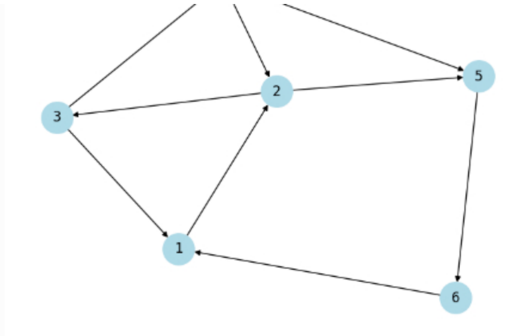
••• 2.1 Directed Graphs (20 Points)

Status	Answered		
Your score	16 / 20		80%

Response

You are given the following graph.



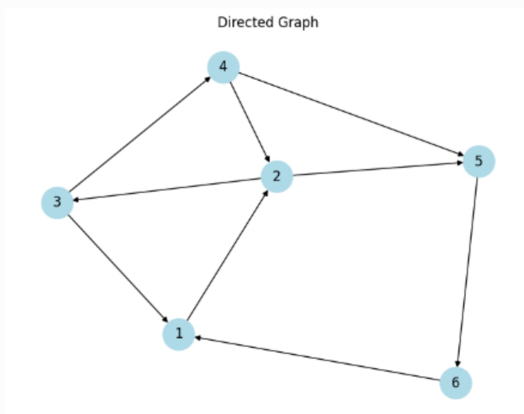


Answer the following questions. Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0, 0.5 → 0.5. Do not add blank spaces!

1. What is the diameter of the above graph? 5.0
2. What is the average path length of the above ? 2.167
3. If edge 6->2 is added to the original graph, what is the new diameter? 4.0
4. Consider the original graph, if edge 1→2 is removed, what is the new diameter of the directed graph? 3
5. What is the density of the original directed graph? 0.3

▾ Solution

You are given the following graph.



Answer the following questions. Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0, 0.5 → 0.5. Do not add blank spaces!

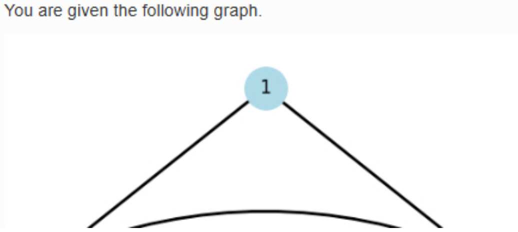
1. What is the diameter of the above graph? 5.0
2. What is the average path length of the above ? 2.167
3. If edge 6->2 is added to the original graph, what is the new diameter? 4.0
4. Consider the original graph, if edge 1→2 is removed, what is the new diameter of the directed graph? undefined
   Undefined, cannot be defined, Infinite, Does not exist, Not applicable, Invalid for this graph, Cannot be computed, invalid
5. What is the density of the original directed graph? 0.3

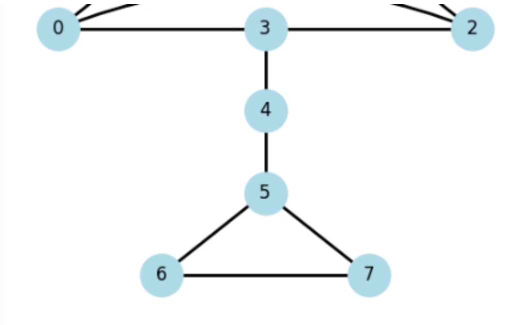
••• 2.2 Undirected Graphs (20 Points)

Status	Answered		
Your score	20 / 20		100%

Response

You are given the following graph.





Answer the following questions. Give the numerical answers rounded to **three** decimal places, e.g., 0.185. Round the last digit as follows: 0.1856 → 0.186, 0.1851 → 0.185, 0.1855 → 0.186, 0→ 0, 0.5 → 0.5. Do not add blank spaces!

1. Calculate the **closeness centrality** for node **5**? 0.467 (3 points)
2. Which node(s) has (have) the highest **betweenness centrality**, and what is the value? If there is more than one node, separate them with a comma. Nodes: 3,4 (3 Points) Betweenness: 0.571 (3 Points)
3. Compute the **clustering coefficien**t for node 0? 0.667 (3 Points)
4. What is the **average clustering coefficient** for the network? 0.625 (3 Points)
5. For node 5, what is the **average degree of its neighbors**? 2.0 (1 Points)
6. Will **friendship paradox** hold for node 5? (yes, no) no (1 Points)
7. If node **3** is removed, what is the **size** of the largest connected component? 4.0 (3 Points)

▸ Solution

go back to overview

3. Programming Tasks (40 Points) 40 of 40 points (100%)

3.1 PageRank (26 Points)

Status	Answered		
Your score	26 / 26		100%

Response

The following code implements the PageRank algorithm. Fill in the gaps with the correct code snippets. Each right answer gives 2 points. No negative points are given for wrong answers.

```
import networkx as nx

def pagerank(G, alpha=0.85, max_iter=100, tol=1.0e-6):
  N = len(G)
  nodes = list( G.nodes() )
  rank = {node: 1.0 / N for node in nodes}

  for _ in range( max_iter ):
    new_rank = {}
    for node in nodes:
      incoming = list(G. predecessors(node) )
      total = 0
      for neighbor in incoming:
        out_deg = G. out_degree(neighbor)
        if out_deg > 0 :
          total += rank[ neighbor ] / out_deg
      new_rank[node] = (1 - alpha ) / N + alpha * total

    diff = sum( abs (new_rank[n] - rank[n]) for n in nodes )
    rank = new_rank
    if diff < tol:
      break

return rank
```

▸ Solution

3.2. Breadth First Search (14 Points)

Status	Answered		
Your score	14 / 14		100%

Response

The following code implements the **Bread First Search (BFS)** algorithm. Fill in the gaps with the correct code snippets. Each right answer gives 2 points. No negative points are given for wrong answers.

```
from collections import deque
```

```
def bfs(graph, start):
    if start not in graph:
        raise ValueError("Start node not in graph")
    visited = {node: False for node in graph}
    distance = {node: -1 for node in graph}
    predecessor = {node: None for node in graph}
    queue = deque()
    visited[start] = [True]
    distance[start] = 0
    queue.append( [start] )

    while queue:
        current = queue. [popleft] ()
        for neighbor in graph[current]:
            if not visited[neighbor]:
                visited[ [neighbor] ] = True
                distance[neighbor] = distance[current] + [1]
                predecessor[neighbor] = [current]
                queue. [append] (neighbor)

    return distance, predecessor

graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

start_node = 'A'
distance, predecessor = bfs(graph, start_node)
```

▸ Solution

< go back to overview

Test execution

Information

Availability: Expired at 6/10/2025, 2:00 PM

Max. attempts: Unlimited

Results of this test are visible to administrators and tutors of this course.

Start test

▸ Change log