DATA STRUCTURE

Chapter 7: Graphs

Model Question Paper with Solutions: Set 5

This set is a complete practice test in the usual university format. Part 1 is the question paper and Part 2 gives the solutions with the marks for each step. Try to finish Part 1 in the given time without looking at Part 2, then mark your own answers.

Time	Maximum marks	Instructions
2 hours	50	Section A and Section B are compulsory. In Section C attempt any TWO questions. Draw neat diagrams wherever necessary.

Part 1: Question Paper

Section A: Answer all questions (10 × 1 = 10 marks)

Q1. State the handshaking property of an undirected graph. **[1]**

Q2. How many edges are there in the complete graph K9? **[1]**

Q3. What is the space complexity of the adjacency list representation of a graph? **[1]**

Q4. Which data structure is used to implement Breadth First Search? **[1]**

Q5. How many edges does a spanning tree of a connected graph with 15 vertices have? **[1]**

Q6. Name the algorithm that finds the shortest paths between all pairs of vertices. **[1]**

Q7. Can Dijkstra's algorithm be used when some edge weights are negative? (Yes or No) **[1]**

Q8. What is a DAG? **[1]**

Q9. How many connected components does a graph with n vertices and no edges have? **[1]**

Q10. The adjacency matrix of an undirected graph is always _____. **[1]**

Section B: Answer all questions (5 × 4 = 20 marks)

Q11. A directed graph has the arcs 1→2, 1→3, 2→3, 3→4, 4→2 and 4→5. (a) Find the in-degree and out-degree of every vertex. (b) Write the adjacency matrix. **[2 + 2]**

Q12. Perform BFS and DFS on the graph of Fig. 1 starting from vertex A. Visit neighbours in alphabetical order. **[2 + 2]**

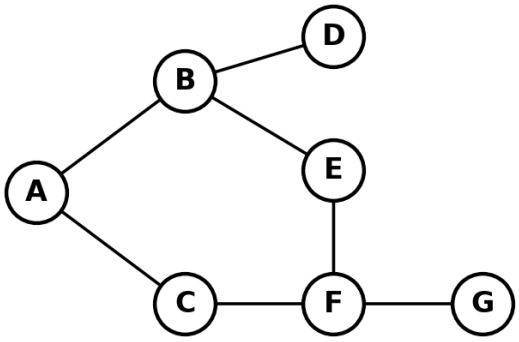

Fig. 1: Graph for Q12

Q13. Find the minimum spanning tree of the graph of Fig. 2 using Kruskal's algorithm and give its total weight. **[4]**

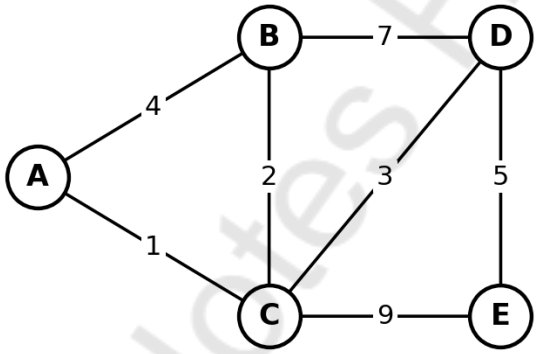

Fig. 2: Graph for Q13 and Q14

Q14. Using Dijkstra's algorithm on the graph of Fig. 2, find the shortest distance from vertex A to every other vertex. **[4]**

Q15. Define a spanning tree. State its properties. How many spanning trees does the complete graph K4 have? **[4]**

Section C: Attempt any TWO questions (2 × 10 = 20 marks)

Q16. For the weighted directed graph of Fig. 3: (a) write the weighted adjacency matrix and the adjacency list; (b) apply Dijkstra's algorithm with source S and show every step; (c) give the shortest path from S to T and its length. **[3 + 5 + 2]**

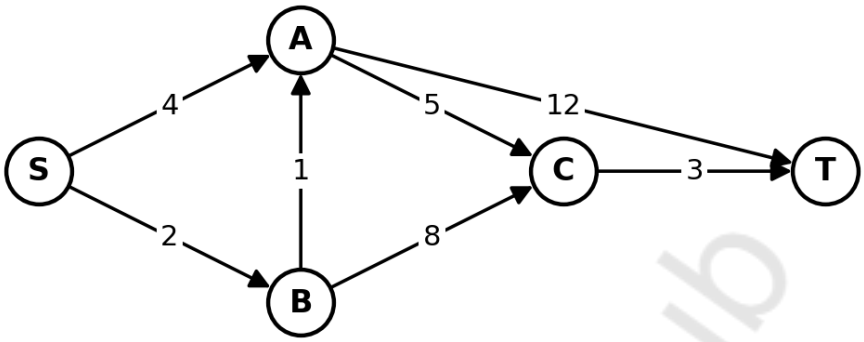

Fig. 3: Graph for Q16

Q17. Write a C program that reads a graph as an adjacency matrix and prints the BFS traversal from a given start vertex. Explain the algorithm, show a sample run and state the time complexity. **[3 + 5 + 1 + 1]**

Q18. For the graph of Fig. 4: (a) find the minimum spanning tree using Kruskal's algorithm; (b) find it again using Prim's algorithm starting from vertex 1; (c) compare the two algorithms and say which is better for a dense graph. **[4 + 4 + 2]**

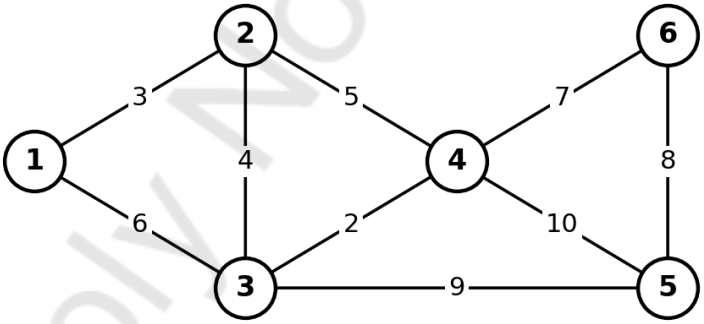

Fig. 4: Graph for Q18

Part 2: Solutions and Marking Scheme

Section A (1 mark each)

Q	Answer
1	In an undirected graph the sum of the degrees of all vertices equals twice the number of edges: Σ deg(v) = 2e.
2	9 × 8 / 2 = 36 edges.
3	O(V + E).
4	A queue.
5	14 edges (n − 1).
6	Floyd–Warshall algorithm.
7	No. It may give wrong results with negative weights; Bellman–Ford should be used.
8	A directed acyclic graph: a directed graph that has no directed cycle.
9	n components (every vertex is a component by itself).
10	Symmetric (A[i][j] = A[j][i]).

Section B (4 marks each)

Q11. (a) In-degree and out-degree [2 marks]

Vertex	In-degree	Out-degree
1	0	2
2	2	1
3	2	1
4	1	2
5	1	0

Total in-degree = 6 = total out-degree = number of arcs. Vertex 1 is the source and vertex 5 is the sink.

(b) Adjacency matrix [2 marks]

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

Q12. BFS [2 marks] and DFS [2 marks]

Adjacency lists (alphabetical): A: B, C; B: A, D, E; C: A, F; D: B; E: B, F; F: C, E, G; G: F.
BFS uses a queue:

Step	Vertex removed	Inserted	Queue after step
1	A	B, C	B, C
2	B	D, E	C, D, E
3	C	F	D, E, F
4	D	none	E, F
5	E	none (F is already queued)	F
6	F	G	G
7	G	none	empty

BFS order: A, B, C, D, E, F, G.

DFS: visit A and go to B. From B go to D; D has no unvisited neighbour, so backtrack to B. From B go to E, then from E go to F. From F go to C (its first unvisited neighbour); C has nothing new, so backtrack to F and go to G. **DFS order: A, B, D, E, F, C, G.** Both traversals take O(V + E) time with adjacency lists.

Q13. Kruskal's algorithm [4 marks]

Sorted edges: A–C (1), B–C (2), C–D (3), A–B (4), D–E (5), B–D (7), C–E (9).

Step	Edge	Weight	Decision
1	A–C	1	Accept
2	B–C	2	Accept
3	C–D	3	Accept
4	A–B	4	Reject (A and B already connected: cycle A–B–C)
5	D–E	5	Accept; 4 = n − 1 edges chosen, stop

MST edges: A–C, B–C, C–D, D–E. Total weight = 1 + 2 + 3 + 5 = 11. (Marks: sorting 1, correct steps 2, final tree and weight 1.)

Q14. Dijkstra's algorithm from A [4 marks]

Vertex selected	dist[B]	dist[C]	dist[D]	dist[E]
Initial (dist[A] = 0)	∞	∞	∞	∞
A (0)	4	1	∞	∞
C (1)	3	1	4	10
B (3)	3	1	4	10
D (4)	3	1	4	9
E (9)	3	1	4	9

Shortest distances from A: B = 3 (A → C → B), C = 1 (A → C), D = 4 (A → C → D), E = 9 (A → C → D → E). The route to E through D (4 + 5 = 9) beats the direct edge C–E (1 + 9 = 10).

Q15. Spanning tree [4 marks]

Definition (1 mark): a spanning tree of a connected graph G is a subgraph that contains all the vertices of G, is connected and has no cycles.

Properties (2 marks):

- A spanning tree of a graph with n vertices has exactly n − 1 edges.
- Adding any edge of G that is not in the tree creates exactly one cycle; removing any tree edge disconnects it.
- A connected graph has at least one spanning tree, and a disconnected graph has none (it has a spanning forest).

Number for K4 (1 mark): by Cayley's formula, Kn has n^(n−2) spanning trees, so K4 has 4^2 = **16**.

Section C (10 marks each)

Q16. (a) Matrix and list [3 marks]

In the matrix the rows and columns are in the order S, A, B, C, T; the diagonal is 0 and missing arcs are ∞.

	S	**A**	**B**	**C**	**T**
S	0	4	2	∞	∞
A	∞	0	∞	5	12
B	∞	1	0	8	∞
C	∞	∞	∞	0	3
T	∞	∞	∞	∞	0

Vertex	**Adjacency list (vertex, weight)**
S	(A, 4) → (B, 2)
A	(C, 5) → (T, 12)
B	(A, 1) → (C, 8)
C	(T, 3)
T	NULL

(b) Dijkstra's algorithm [5 marks]

Vertex selected	dist[A]	dist[B]	dist[C]	dist[T]
Initial (dist[S] = 0)	∞	∞	∞	∞
S (0)	4	2	∞	∞
B (2)	3	2	10	∞
A (3)	3	2	8	15
C (8)	3	2	8	11
T (11)	3	2	8	11

At step B, the arc B→A (2 + 1 = 3) improves dist[A] from 4 and B→C gives 10. At step A, A→C (3 + 5 = 8) improves dist[C] and A→T gives 15. At step C, C→T (8 + 3 = 11) improves dist[T].

(c) Shortest path [2 marks]: the predecessors are A ← B, B ← S, C ← A, T ← C, so the shortest path is **S → B → A → C → T with length 2 + 1 + 5 + 3 = 11**. The direct route S → A → T costs 4 + 12 = 16.

Q17. C program for BFS on an adjacency matrix

Algorithm [3 marks]:

1. Read n and the adjacency matrix a[][]. Read the start vertex and set visited[] to 0 for all vertices.
2. Mark the start vertex visited and insert it into the queue.
3. While the queue is not empty: remove the front vertex v and print it; for every vertex w with a[v][w] = 1 that is not visited, mark it visited and insert it into the queue.

Program [5 marks]:

```
#include <stdio.h>
#define MAX 20

int a[MAX][MAX], visited[MAX], queue[MAX];
int front = 0, rear = -1;

void bfs(int start, int n)
{
    int v, w;
    visited[start] = 1;
    queue[++rear] = start;
    while (front <= rear) {
        v = queue[front++];
        printf("%d ", v);
        for (w = 0; w < n; w++)
            if (a[v][w] == 1 && !visited[w]) {
                visited[w] = 1;
                queue[++rear] = w;
            }
    }
}

int main()
{
    int n, i, j, start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
    printf("Enter start vertex: ");
    scanf("%d", &start);
    printf("BFS order: ");
    bfs(start, n);
    printf("\n");
    return 0;
}
```

Sample run [1 mark]: for the path-like graph with edges 0–1, 0–2, 1–3 the input and output are:

```
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 0
0 1 0 0
Enter start vertex: 0
BFS order: 0 1 2 3
```

Time complexity [1 mark]: every vertex is inserted once and for each removed vertex a full row of n entries is scanned, so the time is **$O(V^2)$** (it would be O(V + E) with an adjacency list). The extra space for the queue and visited array is O(V).

Q18. (a) Kruskal's algorithm [4 marks]

Sorted edges: 3–4 (2), 1–2 (3), 2–3 (4), 2–4 (5), 1–3 (6), 4–6 (7), 5–6 (8), 3–5 (9), 4–5 (10).

Step	Edge	Weight	Decision
1	3–4	2	Accept
2	1–2	3	Accept
3	2–3	4	Accept (joins {1, 2} and {3, 4})
4	2–4	5	Reject (2 and 4 already connected)
5	1–3	6	Reject (1 and 3 already connected)
6	4–6	7	Accept (adds vertex 6)
7	5–6	8	Accept (adds vertex 5); 5 edges chosen, stop

MST edges: 3–4, 1–2, 2–3, 4–6, 5–6; total weight = 2 + 3 + 4 + 7 + 8 = 24. The tree is shown in bold in Fig. 5.

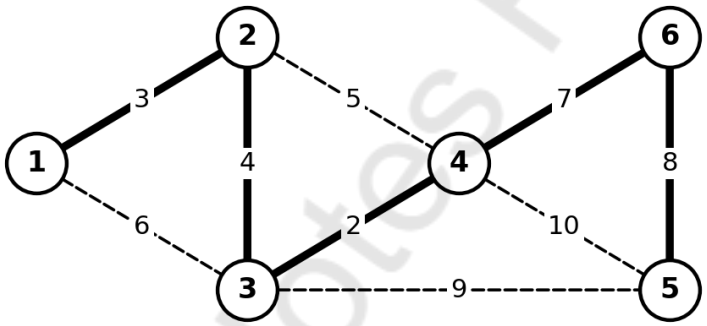

Fig. 5: Minimum spanning tree of Fig. 4 (bold edges), total weight 24

(b) Prim's algorithm from vertex 1 [4 marks]

Step	Tree vertices	Candidate edges	Edge chosen
1	1	1–2 (3), 1–3 (6)	1–2 (3)
2	1, 2	1–3 (6), 2–3 (4), 2–4 (5)	2–3 (4)
3	1, 2, 3	2–4 (5), 3–4 (2), 3–5 (9)	3–4 (2)
4	1, 2, 3, 4	3–5 (9), 4–5 (10), 4–6 (7)	4–6 (7)
5	1, 2, 3, 4, 6	3–5 (9), 4–5 (10), 5–6 (8)	5–6 (8)

Total weight = 3 + 4 + 2 + 7 + 8 = 24, the same tree as in (a).

(c) Comparison [2 marks]:

Point	Kruskal	Prim
Method	Sorts all edges and adds the cheapest edge that forms no cycle	Grows one tree by the cheapest edge leaving it
Time	O(E log E)	$O(V^2)$ with a matrix
Better for	Sparse graphs	Dense graphs

In a dense graph E is close to V^2, so Kruskal's cost O(E log E) becomes about $O(V^2 \log V)$, while Prim's algorithm stays at $O(V^2)$. Hence **Prim's algorithm is better for a dense graph.**

Check Your Score

Add up your marks (maximum 50) and see what to do next.

Marks	Level	What to do
43 – 50	Excellent	You are exam ready. Solve the practice problems of the earlier sets for speed.
33 – 42	Good	Revise the algorithms you lost marks in and redo those solved problems.
23 – 32	Average	Re-read the notes on traversal, spanning trees and shortest path, then attempt this paper again.
Below 23	Needs revision	Start again from the terminology and representation sections, then try Sets 1 and 2.