CS435DE - Lab 11

Saurab Ghimire

417555

Problem 1: Solution

Graph G = (V, E) Problems

Q1(a). G[U] for U = {A, B}

G[U] is the subgraph with vertices A and B and their edges. The edge list shows A-B exists.

Answer:

- Vertices: A, B
- Edge: (A, B)

Q1(b). G[W] for W = {A, C, G, F}

G[W] includes vertices A, C, G, F and all edges between them. From the edge list:

- A-C, A-F, C-F, C-G, F-G

Answer:

- Vertices: A, C, G, F
- Edges: (A, C), (A, F), (C, F), (C, G), (F, G)

Q1(c). G[Y] for Y = {A, B, D, E}

G[Y] has vertices A, B, D, E and their edges. The edge list gives:

- A-B, D-E

Answer:

- Vertices: A, B, D, E
- Edges: (A, B), (D, E)

Q1(d). Subgraph H with Edges B-A, A-F

Can H be G[X] for some vertex set X?

H has edges B-A and A-F, so vertices are A, B, F. For X = {A, B, F}, G[X] includes all edges among them:

- A-B, A-F, B-F (from edge list)

H only has A-B, A-F, missing B-F, which G[X] must include.

Answer: No, H cannot be G[X] since it omits the B-F edge required in G[{A, B, F}].

Q2. Unique MST and Light Edges

Part 1: Unique Light Edge per Cut Implies Unique MST

If every cut in graph G has a unique lightest edge, G has a unique MST.

Suppose G has two MSTs, T1 and T2. If T1 has an edge e not in T2, removing e from T1 makes a cut. T2 must have another edge f crossing this cut. But if e is the unique lightest edge, T2 cannot exclude it and still be an MST. This contradiction means T1 = T2, so the MST is unique.

Part 2: Converse Counterexample

A graph can have a unique MST without every cut having a unique lightest edge.

Take a triangle graph: vertices A, B, C; edges A-B (weight 1), B-C (2), A-C (2).

- MST: A-B (1), A-C (2) (chosen by, say, lexicographic order), total weight 3.
- It's unique with a fixed tie-breaking rule.
- Cut {A, B} vs. C has edges B-C (2), A-C (2)—two lightest edges.

Thus, the converse is false.

Q3. Hamiltonian Cycle

The graph, like a truncated icosahedron with 20 vertices, has a Hamiltonian cycle (visits each vertex once, returns to start).

Label outer vertices v1 to v10 clockwise. A cycle:

- v1 → v2 → v3 → inner a → b → c → d → v4 → v5 → inner e → f → g → v6 → v7 → inner h → i → j → v8 → v9 → v10 → v1

This hits all vertices once and cycles back.

Answer: The graph has a Hamiltonian cycle, weaving through outer and inner vertices.

Q4. Max Spanning Tree with Prim's and Kruskal's

Can Prim's and Kruskal's algorithms find a Maximum Spanning Tree (MaxST) by picking max-weight edges?

Kruskal's: Sort edges by decreasing weight, add them using Union-Find, avoiding cycles, until V-1 edges.

Prim's: Start at a vertex, always add the max-weight edge to a new vertex, using a max-priority queue.

Both work because their greedy approach applies to maximizing weights, ensuring a connected tree with max total weight.

Answer: Yes, both algorithms compute a MaxST by choosing max-weight edges instead of min-weight.