MH1301 Discrete Mathematics

Tutorial 12

Academic Year 2025/2026, Semester 2

Quantitative Research Society @NTU

August 17, 2026

Overview

This tutorial focuses on weighted graphs, minimum spanning trees, shortest paths, and bottleneck spanning trees. The main themes are:

- Prim's algorithm and Kruskal's algorithm for minimum spanning trees;
- edge-order and cut/cycle reasoning in MST construction;
- Dijkstra's algorithm for shortest paths in positively weighted graphs;
- shortest-path subpath properties;
- minimum bottleneck spanning trees versus minimum spanning trees;
- MST cut-property proofs;
- explicit MST computation in a complete graph with structured edge weights.

Question map.

- Questions 1–2: Prim's and Kruskal's algorithms on the same weighted graph.
- Question 3: MSTs for two separate weighted graphs using different algorithms.
- Questions 4–5: Dijkstra's algorithm and shortest-path distances.
- Questions 6–7: minimum bottleneck spanning trees and MSTs.
- Question 8: Kruskal's algorithm on K_n with edge weights $w(\{u, v\}) = u + v$.

For academic reference only. This document is provided for educational purposes and does not constitute an official question paper, answer key, or any formally authorised academic material. Its content is not guaranteed to be complete or accurate.

Contents

Question 1 (Prim's algorithm for a minimum spanning tree) 4
Problem . . . 4
Solution . . . 4
Method 1: Prim's algorithm starting at a . . . 4
Method 2: Cut-property verification of the Prim tree . . . 5

Question 2 (Kruskal's algorithm for a minimum spanning tree) 6
Problem . . . 6
Solution . . . 6
Method 1: Kruskal's algorithm by increasing edge weights . . . 6
Method 2: Cycle-property verification . . . 7

Question 3 (Minimum spanning trees for two weighted graphs) 8
Problem . . . 8
Solution . . . 8
Method 1: Prim on the first graph and Kruskal on the second graph . . . 8
Method 2: Cut and cycle verification . . . 9

Question 4 (Dijkstra's algorithm from a to z) 10
Problem . . . 10
Solution . . . 10
Method 1: Dijkstra table . . . 10
Method 2: Shortest-path tree verification . . . 11

Question 5 (Shortest path lengths in the graph from Question 4) 12
Problem . . . 12
Solution . . . 12
Method 1: Shortest-path tree and local Dijkstra checks . . . 12
Method 2: Subpath property and reverse-distance check . . . 13

Question 6 (Minimum bottleneck spanning trees need not be MSTs) 15
Problem . . . 15
Solution . . . 15
Method 1: Explicit counterexample . . . 15
Method 2: Threshold-connectivity interpretation . . . 16

Question 7 (Every MST is a minimum bottleneck spanning tree) 17
Problem . . . 17
Solution . . . 17
Method 1: Cut-property proof . . . 17
Method 2: Kruskal-prefix proof . . . 17

Question Assistant Solutions

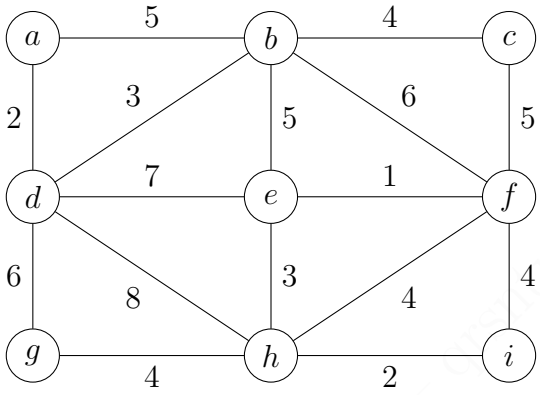
Question 8 (MST of K_n with weights $w(\{u, v\}) = u + v$) **19**
Problem 19
Solution 19
Method 1: Kruskal's algorithm and cycle rejection 19
Method 2: Cut-property proof 20

Unofficial — QRS@NTU — qrsntu.org

Question 1 (Prim's algorithm for a minimum spanning tree)

Problem

(Ex. 11.5.2). Use Prim's algorithm to find a minimum spanning tree for the given weighted graph.



Solution

Method 1: Prim's algorithm starting at a

Algorithm used. Prim's algorithm starts with one vertex and repeatedly adds the minimum-weight edge joining the current tree to a vertex outside it.

Start at vertex a.

Step	Current vertex set	Edge added
0	$\{a\}$	–
1	$\{a, d\}$	$\{a, d\},\ w = 2$
2	$\{a, d, b\}$	$\{b, d\},\ w = 3$
3	$\{a, d, b, c\}$	$\{b, c\},\ w = 4$
4	$\{a, d, b, c, e\}$	$\{b, e\},\ w = 5$
5	$\{a, d, b, c, e, f\}$	$\{e, f\},\ w = 1$
6	$\{a, d, b, c, e, f, h\}$	$\{e, h\},\ w = 3$
7	$\{a, d, b, c, e, f, h, i\}$	$\{h, i\},\ w = 2$
8	$\{a, d, b, c, e, f, h, i, g\}$	$\{g, h\},\ w = 4$

Thus one minimum spanning tree obtained by Prim's algorithm is

$$T = \big\{\{a, d\}, \{b, d\}, \{b, c\}, \{b, e\}, \{e, f\}, \{e, h\}, \{h, i\}, \{g, h\}\big\}.$$

Its total weight is

$$2 + 3 + 4 + 5 + 1 + 3 + 2 + 4 = 24.$$

Hence

$$\boxed{E(T) = \{\{a, d\}, \{b, d\}, \{b, c\}, \{b, e\}, \{e, f\}, \{e, h\}, \{h, i\}, \{g, h\}\}, \quad w(T) = 24.}$$

Method 2: Cut-property verification of the Prim tree

Theorem/criterion used. For any cut $(S, V \setminus S)$ in a weighted graph, if an edge crossing the cut has strictly minimum weight among all edges crossing that cut, then it is safe for some MST. Prim's algorithm is correct because each selected edge is a safe edge for the cut determined by the current tree.

The selected edges are checked by the corresponding cuts:

S	minimum crossing edge chosen	weight
$\{a\}$	$\{a, d\}$	2
$\{a, d\}$	$\{b, d\}$	3
$\{a, d, b\}$	$\{b, c\}$	4
$\{a, d, b, c\}$	$\{b, e\}$ or $\{c, f\}$	5
$\{a, d, b, c, e\}$	$\{e, f\}$	1
$\{a, d, b, c, e, f\}$	$\{e, h\}$	3
$\{a, d, b, c, e, f, h\}$	$\{h, i\}$	2
$\{a, d, b, c, e, f, h, i\}$	$\{g, h\}$	4

At the fourth step there is a tie: $\{b, e\}$ and $\{c, f\}$ both have weight 5. Choosing either gives an MST of the same total weight.

Therefore the tree found in Method 1 is a minimum spanning tree:

$$\boxed{w(T) = 24.}$$

Unofficial — QRS@NTU — Consult Only

Question 2 (Kruskal's algorithm for a minimum spanning tree)

Problem

(Ex. 11.5.4). Use Kruskal's algorithm to find a minimum spanning tree for the weighted graph in Question 1.

Solution

Method 1: Kruskal's algorithm by increasing edge weights

Algorithm used. Kruskal's algorithm sorts all edges in nondecreasing order of weight and adds an edge exactly when it does not create a cycle.

The edges in increasing order of weight are:

Weight	Edges
1	$\{e, f\}$
2	$\{a, d\}, \{h, i\}$
3	$\{b, d\}, \{e, h\}$
4	$\{b, c\}, \{f, i\}, \{g, h\}, \{f, h\}$
5	$\{a, b\}, \{b, e\}, \{c, f\}$
6	$\{d, g\}, \{b, f\}$
7	$\{d, e\}$
8	$\{d, h\}$

Proceeding by Kruskal's algorithm:

Order	Edge considered	Action
1	$\{e, f\}$	add
2	$\{a, d\}$	add
3	$\{h, i\}$	add
4	$\{b, d\}$	add
5	$\{e, h\}$	add
6	$\{b, c\}$	add
7	$\{f, i\}$	reject: cycle
8	$\{g, h\}$	add
9	$\{f, h\}$	reject: cycle
10	$\{a, b\}$	reject: cycle
11	$\{b, e\}$	add

After 8 edges have been added, all 9 vertices are connected, so the MST is complete.

Thus Kruskal's algorithm gives

$$T = \big\{\{e, f\}, \{a, d\}, \{h, i\}, \{b, d\}, \{e, h\}, \{b, c\}, \{g, h\}, \{b, e\}\big\}.$$

The total weight is

$$1 + 2 + 2 + 3 + 3 + 4 + 4 + 5 = 24.$$

Therefore

$$\boxed{E(T) = \{\{e, f\}, \{a, d\}, \{h, i\}, \{b, d\}, \{e, h\}, \{b, c\}, \{g, h\}, \{b, e\}\}, \quad w(T) = 24.}$$

Method 2: Cycle-property verification

Theorem/criterion used. The cycle property says that, in any cycle, an edge of strictly largest weight cannot belong to an MST. Kruskal's algorithm repeatedly avoids forming cycles and therefore keeps only safe edges.

The edges rejected in Method 1 are rejected because they close cycles:

$$\{f, i\} \text{ closes } f - e - h - i - f,$$
$$\{f, h\} \text{ closes } f - e - h - f,$$
$$\{a, b\} \text{ closes } a - d - b - a.$$

The final accepted edge $\{b, e\}$ joins the component containing a, b, c, d to the component containing e, f, g, h, i. It may be replaced by $\{c, f\}$, which has the same weight 5, giving another MST of total weight 24.

Thus the MST need not be unique, but every tree produced here has minimum total weight

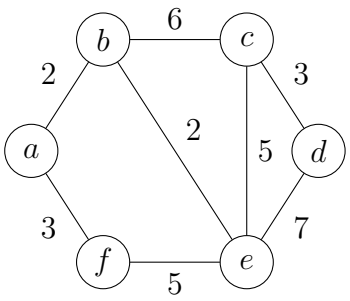
$$\boxed{24.}$$

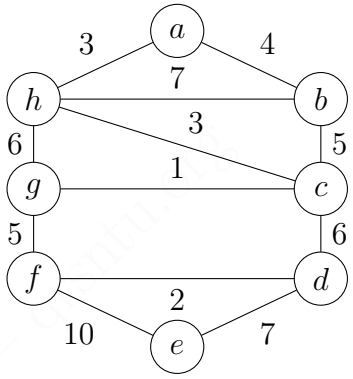
Unofficial — QRS@NTU — qrs.ntu.org

Question 3 (Minimum spanning trees for two weighted graphs)

Problem

Find a minimum spanning tree for each of the following two graphs. Use a different algorithm for each graph. For each graph, state the name of the algorithm used, and list the edges in the order that they are added to the MST.





Solution

Method 1: Prim on the first graph and Kruskal on the second graph

(a) **First graph: Prim's algorithm starting at a.**

Starting at a, add the minimum-weight edge crossing from the current tree to the outside:

Step	Current vertex set	Edge added
0	$\{a\}$	–
1	$\{a, b\}$	$\{a, b\},\ w = 2$
2	$\{a, b, e\}$	$\{b, e\},\ w = 2$
3	$\{a, b, e, f\}$	$\{a, f\},\ w = 3$
4	$\{a, b, e, f, c\}$	$\{e, c\},\ w = 5$
5	$\{a, b, e, f, c, d\}$	$\{c, d\},\ w = 3$

Thus one MST is

$$T_1 = \{\{a, b\}, \{b, e\}, \{a, f\}, \{e, c\}, \{c, d\}\},$$

with total weight

$$2 + 2 + 3 + 5 + 3 = 15.$$

Hence

$$\boxed{E(T_1) = \{\{a, b\}, \{b, e\}, \{a, f\}, \{e, c\}, \{c, d\}\}, \quad w(T_1) = 15.}$$

(b) **Second graph: Kruskal's algorithm.**

Sort the edges by weight:

Weight	Edges
1	$\{g, c\}$
2	$\{f, d\}$
3	$\{a, h\}, \{h, c\}$
4	$\{a, b\}$
5	$\{b, c\}, \{g, f\}$
6	$\{h, g\}, \{c, d\}$
7	$\{h, b\}, \{d, e\}$
10	$\{f, e\}$

Kruskal's algorithm adds:

$$\{g, c\}, \{f, d\}, \{a, h\}, \{h, c\}, \{a, b\}, \{g, f\}, \{d, e\}.$$

The total weight is

$$1 + 2 + 3 + 3 + 4 + 5 + 7 = 25.$$

Thus

$$\boxed{E(T_2) = \{\{g, c\}, \{f, d\}, \{a, h\}, \{h, c\}, \{a, b\}, \{g, f\}, \{d, e\}\}, \quad w(T_2) = 25.}$$

Method 2: Cut and cycle verification

(a) For the first graph, Prim's chosen edges are safe for the successive cuts

$$\{a\}, \quad \{a, b\}, \quad \{a, b, e\}, \quad \{a, b, e, f\}, \quad \{a, b, e, f, c\}.$$

The selected crossing edges have weights

$$2, 2, 3, 5, 3,$$

respectively. By the cut property, these edges form an MST. Hence

$$\boxed{w(T_1) = 15.}$$

(b) For the second graph, after adding

$$\{g, c\}, \{f, d\}, \{a, h\}, \{h, c\}, \{a, b\}, \{g, f\},$$

the only remaining isolated vertex is e. The cheaper incident edge to connect e is

$$\{d, e\}, \qquad w = 7,$$

since

$$w(\{d, e\}) = 7 < 10 = w(\{f, e\}).$$

The edge $\{b, c\}$ is skipped because it closes the cycle

$$b - a - h - c - b.$$

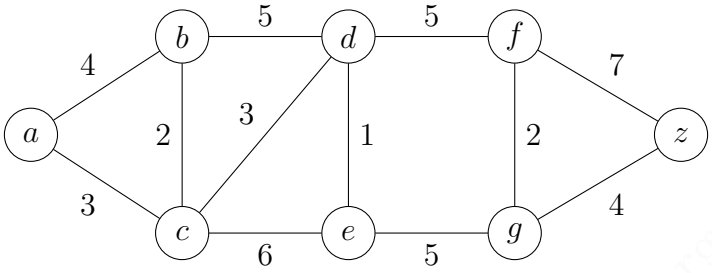
The edges $\{h, g\}$ and $\{c, d\}$ are skipped because their endpoints are already connected. Therefore Kruskal's accepted edges form an MST of total weight

$$\boxed{25.}$$

Question 4 (Dijkstra's algorithm from a to z)

Problem

(Ex. 10.6.2). Find the distance between a and z in the following graph using Dijkstra algorithm. Illustrate how the values of $d[v]$ are updated using a table.



Solution

Method 1: Dijkstra table

Algorithm used. Dijkstra's algorithm repeatedly selects the unvisited vertex with smallest tentative distance and relaxes all edges incident to it. It is valid here because all edge weights are positive.

The distance labels are updated as follows:

Step	$d[a]$	$d[b]$	$d[c]$	$d[d]$	$d[e]$	$d[f]$	$d[g]$	$d[z]$	S
0	0	∞	∞	∞	∞	∞	∞	∞	$\varnothing$
1	0	4	3	∞	∞	∞	∞	∞	$\{a\}$
2	0	4	3	6	9	∞	∞	∞	$\{a, c\}$
3	0	4	3	6	9	∞	∞	∞	$\{a, c, b\}$
4	0	4	3	6	7	11	∞	∞	$\{a, c, b, d\}$
5	0	4	3	6	7	11	12	∞	$\{a, c, b, d, e\}$
6	0	4	3	6	7	11	12	18	$\{a, c, b, d, e, f\}$
7	0	4	3	6	7	11	12	16	$\{a, c, b, d, e, f, g\}$
8	0	4	3	6	7	11	12	16	$\{a, c, b, d, e, f, g, z\}$

Thus

$$d[z] = 16.$$

The shortest path corresponding to the predecessor updates is

$$a - c - d - e - g - z,$$

with total length

$$3 + 3 + 1 + 5 + 4 = 16.$$

Therefore

$$\boxed{\text{dist}(a, z) = 16.}$$

Method 2: Shortest-path tree verification

From Dijkstra's algorithm, the final shortest-path predecessor tree from a is

$$\{ac,\ ab,\ cd,\ de,\ df,\ eg,\ gz\}.$$

The corresponding distances are:

v	a	b	c	d	e	f	g	z
$d[v]$	0	4	3	6	7	11	12	16

Each tree edge realizes the displayed distance:

$$\begin{aligned} d[c] &= d[a] + 3 = 3, \\ d[b] &= d[a] + 4 = 4, \\ d[d] &= d[c] + 3 = 6, \\ d[e] &= d[d] + 1 = 7, \\ d[f] &= d[d] + 5 = 11, \\ d[g] &= d[e] + 5 = 12, \\ d[z] &= d[g] + 4 = 16. \end{aligned}$$

Checking the non-tree edges gives no improvement:

$$\begin{aligned} d[b] + 2 &= 6 \geq d[c] = 3, \\ d[c] + 6 &= 9 \geq d[e] = 7, \\ d[f] + 2 &= 13 \geq d[g] = 12, \\ d[f] + 7 &= 18 \geq d[z] = 16. \end{aligned}$$

Hence the labels are final shortest-path distances. Therefore

$$\boxed{\text{dist}(a, z) = 16.}$$

Unofficial QRS@NTU qrsntu.org

Question 5 (Shortest path lengths in the graph from Question 4)

Problem

(Ex. 10.6.4). Find the length of a shortest path between these pairs of vertices in the weighted graph in Question 4.

(a) a and d;

(b) a and f;

(c) c and f;

(d) b and z.

Solution

Method 1: Shortest-path tree and local Dijkstra checks

(a) From Question 4, Dijkstra's algorithm gives

$$d[d] = 6.$$

A shortest path is

$$a - c - d,$$

with length

$$3 + 3 = 6.$$

Hence

$$\boxed{\text{dist}(a, d) = 6.}$$

(b) From Question 4,

$$d[f] = 11.$$

A shortest path is

$$a - c - d - f,$$

with length

$$3 + 3 + 5 = 11.$$

Hence

$$\boxed{\text{dist}(a, f) = 11.}$$

(c) The path

$$c - d - f$$

has length

$$3 + 5 = 8.$$

Unofficial — QRS@NTU — qrs.ntu.org

To verify minimality, run Dijkstra from c until f is settled:

Step	$d[c]$	$d[b]$	$d[a]$	$d[d]$	$d[e]$	$d[f]$	$d[g]$	$d[z]$	S
0	0	∞	∞	∞	∞	∞	∞	∞	$\varnothing$
1	0	2	3	3	6	∞	∞	∞	$\{c\}$
2	0	2	3	3	6	∞	∞	∞	$\{c, b\}$
3	0	2	3	3	6	∞	∞	∞	$\{c, b, a\}$
4	0	2	3	3	4	8	∞	∞	$\{c, b, a, d\}$
5	0	2	3	3	4	8	9	∞	$\{c, b, a, d, e\}$
6	0	2	3	3	4	8	9	15	$\{c, b, a, d, e, f\}$

Thus

$$\boxed{\text{dist}(c, f) = 8.}$$

(d) A shortest path from b to z is

$$b - d - e - g - z,$$

with length

$$5 + 1 + 5 + 4 = 15.$$

Another shortest path is

$$b - c - d - e - g - z,$$

with length

$$2 + 3 + 1 + 5 + 4 = 15.$$

Running Dijkstra from b gives final label $d[z] = 15$, so

$$\boxed{\text{dist}(b, z) = 15.}$$

Thus the requested lengths are

$$\boxed{\text{(a) } 6, \quad \text{(b) } 11, \quad \text{(c) } 8, \quad \text{(d) } 15.}$$

Method 2: Subpath property and reverse-distance check

Theorem/criterion used. Every subpath of a shortest path is itself a shortest path between its endpoints.

(a) Since

$$a - c - d - e - g - z$$

is a shortest path from a to z, its subpath

$$a - c - d$$

is a shortest path from a to d. Hence

$$\text{dist}(a, d) = 3 + 3 = 6.$$

Therefore

$$\boxed{\text{dist}(a, d) = 6.}$$

(b) The Dijkstra shortest-path tree gives the path

$$a - c - d - f.$$

Its length is

$$3 + 3 + 5 = 11.$$

Therefore

$$\boxed{\text{dist}(a, f) = 11.}$$

(c) The subpath

$$c - d - f$$

of the shortest path

$$a - c - d - f$$

is shortest from c to f. Thus

$$\text{dist}(c, f) = 3 + 5 = 8.$$

Hence

$$\boxed{\text{dist}(c, f) = 8.}$$

(d) Compute the relevant distances to z:

$$\text{dist}(d, z) = 1 + 5 + 4 = 10 \qquad \text{via } d - e - g - z,$$

and

$$\text{dist}(c, z) = 3 + 1 + 5 + 4 = 13 \qquad \text{via } c - d - e - g - z.$$

Since the neighbours of b are a, c, d, any path from b to z must begin with one of

$$b - a, \quad b - c, \quad b - d.$$

Therefore

$$\begin{aligned} \text{dist}(b, z) &= \min\{4 + \text{dist}(a, z),\ 2 + \text{dist}(c, z),\ 5 + \text{dist}(d, z)\} \\ &= \min\{4 + 16,\ 2 + 13,\ 5 + 10\} \\ &= 15. \end{aligned}$$

Thus

$$\boxed{\text{dist}(b, z) = 15.}$$

Unofficial - QRS@NTU - qrs.ntu.org

Question 6 (Minimum bottleneck spanning trees need not be MSTs)

Problem

One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with minimum total cost. Here we explore another type of objective: designing a spanning network for which the most expensive edge is as cheap as possible.

Specifically, let $G = (V, E)$ be a connected graph with n vertices, m edges, and positive edge costs that you may assume are all distinct. Let $T = (V, E')$ be a spanning tree of G; we define the bottleneck edge of T to be the edge of T with the greatest cost.

A spanning tree T of G is a minimum bottleneck spanning tree if there is no spanning tree T' of G with a cheaper bottleneck edge.

Is every minimum-bottleneck tree of G a minimum spanning tree of G? Prove or give a counterexample.

Solution

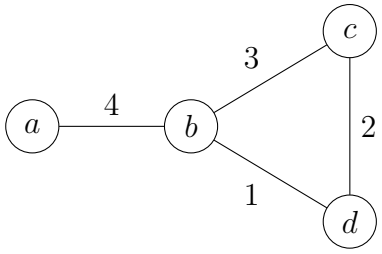
Method 1: Explicit counterexample

The claim is false. Consider the weighted graph

$$V = \{a, b, c, d\},$$

with edge weights

$$w(ab) = 4, \qquad w(bc) = 3, \qquad w(cd) = 2, \qquad w(bd) = 1.$$



The vertex a is incident only with the edge ab. Hence every spanning tree must contain ab, and therefore every spanning tree has bottleneck cost at least

$$4.$$

Now take

$$T = \{\{a, b\}, \{b, c\}, \{c, d\}\}.$$

This is a spanning tree, and its edge weights are

$$4, 3, 2.$$

Thus its bottleneck cost is

$$4.$$

Since no spanning tree can have bottleneck cost below 4, this T is a minimum bottleneck spanning tree.

However, T is not a minimum spanning tree, because

$$w(T) = 4 + 3 + 2 = 9,$$

whereas

$$T^* = \{\{a, b\}, \{b, d\}, \{d, c\}\}$$

has total weight

$$w(T^*) = 4 + 1 + 2 = 7 < 9.$$

Therefore

$$\boxed{\text{not every minimum bottleneck spanning tree is a minimum spanning tree.}}$$

Method 2: Threshold-connectivity interpretation

Criterion used. A spanning tree has bottleneck at most B if and only if the subgraph formed by all edges of weight at most B is connected.

In the graph above, the subgraph using only edges of weight at most 3 has edge set

$$\{bd, dc, bc\}.$$

This subgraph does not contain vertex a as a connected vertex, so it is not connected. Hence no spanning tree has bottleneck at most 3.

The subgraph using edges of weight at most 4 is the whole graph and is connected. Therefore the minimum bottleneck value is exactly

$$4.$$

The tree

$$\{\{a, b\}, \{b, c\}, \{c, d\}\}$$

has bottleneck 4, so it is minimum-bottleneck. But its total cost is 9, while the spanning tree

$$\{\{a, b\}, \{b, d\}, \{d, c\}\}$$

has total cost 7. Thus the minimum-bottleneck tree need not minimize total weight:

$$\boxed{\text{False.}}$$

Question 7 (Every MST is a minimum bottleneck spanning tree)

Problem

Is every minimum spanning tree of G a minimum-bottleneck tree of G? Prove or give a counterexample. *(Hint: use the property of safe edges.)*

Solution

Method 1: Cut-property proof

Theorem/criterion used. Let T be an MST. If $e \in E(T)$ and deleting e partitions T into two components with vertex sets A and B, then no edge crossing the cut (A, B) has weight smaller than $w(e)$. Otherwise, replacing e by that smaller crossing edge would produce a spanning tree of smaller total weight.

Let T be a minimum spanning tree of G. Let e be a bottleneck edge of T, so

$$w(e) = \max_{f \in E(T)} w(f).$$

Delete e from T. This produces two components with vertex sets A and B. Since T is an MST, e is a minimum-weight edge crossing the cut (A, B). Hence every edge crossing the cut has weight at least $w(e)$.

Every spanning tree of G must contain at least one edge crossing the cut (A, B); otherwise it cannot connect vertices in A to vertices in B. Therefore every spanning tree has bottleneck edge of weight at least

$$w(e).$$

But T itself has bottleneck weight exactly $w(e)$. Hence no spanning tree has a cheaper bottleneck edge than T.

Therefore

every minimum spanning tree is a minimum bottleneck spanning tree.

Method 2: Kruskal-prefix proof

Theorem/criterion used. Kruskal's algorithm constructs an MST by adding edges in increasing order of weight whenever they connect two different components.

Let T be an MST, and let B be its bottleneck weight. Suppose, for contradiction, that there is a spanning tree T' whose bottleneck is strictly smaller than B. Then every edge of T' has weight less than B.

Let e be an edge of T with weight B. In Kruskal's algorithm, before edges of weight B are used, the graph formed by all edges of weight less than B is not yet connected; otherwise Kruskal would have completed a spanning tree without needing any edge of weight B.

But if T' has all edges of weight less than B, then the edges of T' form a connected spanning subgraph using only edges of weight less than B. This contradicts the fact that the edges of weight less than B do not connect all vertices.

Hence no spanning tree has bottleneck smaller than B. Therefore every MST is a minimum bottleneck spanning tree:

True.

Unofficial — QRS@NTU — qrsntu.org

Question 8 (MST of K_n with weights $w(\{u, v\}) = u + v$)

Problem

Let the complete graph K_n have vertices

$$\{1, 2, \ldots, n\}$$

and suppose that for each

$$u, v \in \{1, 2, \ldots, n\},$$

the edge $\{u, v\}$ has weight

$$w(\{u, v\}) = u + v.$$

For example, edge $\{1, 2\}$ has weight $w(\{1, 2\}) = 1+2 = 3$, edge $\{2, 4\}$ has weight $w(\{2, 4\}) = 2 + 4 = 6$, etc.

Determine the minimum spanning tree of this graph, and compute the total cost for this minimum spanning tree.

Hint: apply Kruskal's algorithm to this graph, and see what are the edges added to the MST.

Solution

Method 1: Kruskal's algorithm and cycle rejection

Algorithm used. Kruskal's algorithm considers edges in nondecreasing order of weight and adds an edge if and only if it does not create a cycle.

We claim that Kruskal's algorithm selects exactly the edges

$$\{1, 2\}, \{1, 3\}, \ldots, \{1, n\}.$$

These edges form the star centered at 1.

Consider an edge $\{i, j\}$ with

$$2 \le i < j \le n.$$

Its weight is

$$w(\{i, j\}) = i + j.$$

The edges

$$\{1, i\} \qquad \text{and} \qquad \{1, j\}$$

have weights

$$1 + i < i + j, \qquad 1 + j < i + j,$$

because $i, j \ge 2$. Hence Kruskal's algorithm considers and accepts $\{1, i\}$ and $\{1, j\}$ before it considers $\{i, j\}$.

When $\{i, j\}$ is considered, the vertices i and j are already connected through

$$i - 1 - j.$$

Thus adding $\{i, j\}$ would create the cycle

$$1 - i - j - 1,$$

so Kruskal's algorithm rejects it.

Therefore the only accepted edges are

$$\{1, 2\}, \{1, 3\}, \ldots, \{1, n\}.$$

The total cost is

$$\begin{aligned}\sum_{k=2}^{n} w(\{1, k\}) &= \sum_{k=2}^{n}(1 + k) \\ &= (n - 1) + \sum_{k=2}^{n} k \\ &= (n - 1) + \left(\frac{n(n + 1)}{2} - 1\right) \\ &= \frac{n(n + 1)}{2} + n - 2 \\ &= \frac{(n + 1)(n + 2)}{2} - 3.\end{aligned}$$

Hence the MST is

$$\boxed{E(T) = \{\{1, 2\}, \{1, 3\}, \ldots, \{1, n\}\}, \qquad w(T) = \frac{(n + 1)(n + 2)}{2} - 3.}$$

Method 2: Cut-property proof

Theorem/criterion used. If an edge is the unique minimum-weight edge crossing a cut, then it belongs to every MST.

For each

$$k \in \{2, 3, \ldots, n\},$$

consider the cut

$$\big(\{k\}, \{1, 2, \ldots, n\} \setminus \{k\}\big).$$

The edges crossing this cut are exactly

$$\{k, j\} \qquad (j \neq k).$$

Their weights are

$$w(\{k, j\}) = k + j.$$

This is minimized when

$$j = 1.$$

Thus the unique lightest edge crossing the cut is

$$\{1, k\},$$

with weight

$$1 + k.$$

By the cut property, every MST contains

$$\{1, k\} \qquad \text{for each } k = 2, \ldots, n.$$

There are exactly $n - 1$ such edges, and they form a spanning tree. Therefore they are precisely the MST.

Thus

$$T = K_{1,n-1}$$

centered at vertex 1, and its total cost is

$$\sum_{k=2}^{n}(1 + k) = \frac{(n + 1)(n + 2)}{2} - 3.$$

Therefore

the MST is the star centered at 1, $\quad w(T) = \dfrac{(n + 1)(n + 2)}{2} - 3.$

Unofficial — QRS@NTU — qrsntu.org