Problem Set VIII

Huy Quang Lai
132000359

Texas A&M University

16 November 2022

Topological Sort

$s, G, D, A, H, B, E, I, F, C, t$

Shortest Path

A. Shortest unweighted path from A
   A
   $A \rightarrow B$
   $A \rightarrow B \rightarrow C$
   $A \rightarrow B \rightarrow C \rightarrow D$
   $A \rightarrow B \rightarrow C \rightarrow E$
   $A \rightarrow B \rightarrow C \rightarrow E \rightarrow F$
   $A \rightarrow B \rightarrow G$

B. Shortest Path from B
   $B \rightarrow G \rightarrow E \rightarrow D \rightarrow A$ distance: 6
   B distance: 0
   $B \rightarrow C$ distance: 2
   $B \rightarrow G \rightarrow E \rightarrow D$ distance: 4
   $B \rightarrow G \rightarrow E$ distance: 2
   $B \rightarrow G \rightarrow E \rightarrow F$ distance: 3
   $B \rightarrow G$ distance: 1

Dijkstra

Problem 1

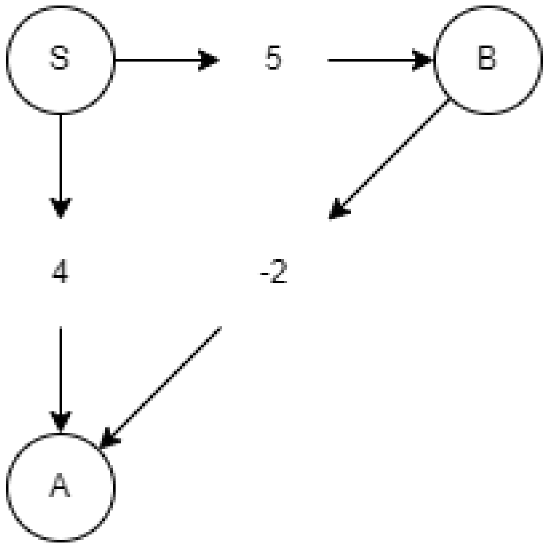

Figure 1: Incorrect Result from Dijkstra

Using S as the source node, Dijkstra's algorithm will visit vertex A since it's closer than vertex B. The distance to A is set to 4 and A is marked as visited. Next Dijkstra's algorithm will traverse to B setting this distance to 5 and marking B as visited. Since A is marked as visited, Dijkstra's algorithm does not traverse down B's connection to A.

However, this result is incorrect, Dijkstra's algorithm determines the shortest path to A is $S \rightarrow A$ with a distance of 4. However the true shortest path is $S \rightarrow B \rightarrow A$ with a distance of 3.

Problem 2

A. Count the number of Different paths.
Using a `map` called `count` such that for any vertex `u`, `count[u]` is the number of distinct paths from `s` to `u` known so far.
Let `v` vertex being marked as known. Let `w` be a vertex on the adjacency list of `v`.
If `distance[v]+cost[u,v]` is equal to `distance[w]`, then increment `count[w]` by `count[v]`.
If `distance[v]+cost[u,v]` is less than `distance[w]`, then `parent[w]` and `distance[w]` get updated.
All previously known shortest paths to `w` are now invalid, but all shortest paths to `v` now lead to shortest paths for `w`.
Set `count[w]` to equal `+count[v]`.

B. Use the least number of edges.
Using a `map` called `numEdges` such that for any vertex `u`, `numEdges[u]` is the number of edges from s to u known so far. Let `v` vertex being marked as known.
Let `w` be a vertex on the adjacency list of `v`.
If `distance[v]+cost[u,v]` is equal to `distance[w]`, then change `parent[w]` to `v`. After this, set `numEdges[w]` to be `min(numEdges[v]+1,numEdges[w])`
If `distance[v]+cost[u,v]` is less than `distance[w]`, then `parent[w]` and `distance[w]` get updated. After this, set `numEdges[w]` to be `numEdges[v]+1`.

Minimum Spanning Tree

Problem 1

Prim's Algorithm

v	known	distance	parent
A	X	0	-
B	X	3	A
C	X	9	G
D	X	4	A
E	X	4	A
F	X	6	B
G	X	8	F
H	X	6	E
I	X	5	E
J	X	7	I

MST:
AB, CG, DA, EA, FB, GF, HE, IE, JI

Kruskal's Algorithm
pq all the edges
EI(1), CG(1), BE(2), EH(2), FG(2), AB(3), BF(3), FI(3), AD(4), AE(4), HI(4), DE(5), CF(6), DH(6), IJ(7), GJ(8), BC(10), EF(11), FJ(11)

MST:
EI, CG, BE, EH, FG, AB, BF, AD, IJ

Since the minimum spanning trees resulting from Prim's and Kruskal's Algorithms are the same, there is only one minimum spanning tree.

Problem 2

A. 1 mst

B. 1 mst

C. 3 mst

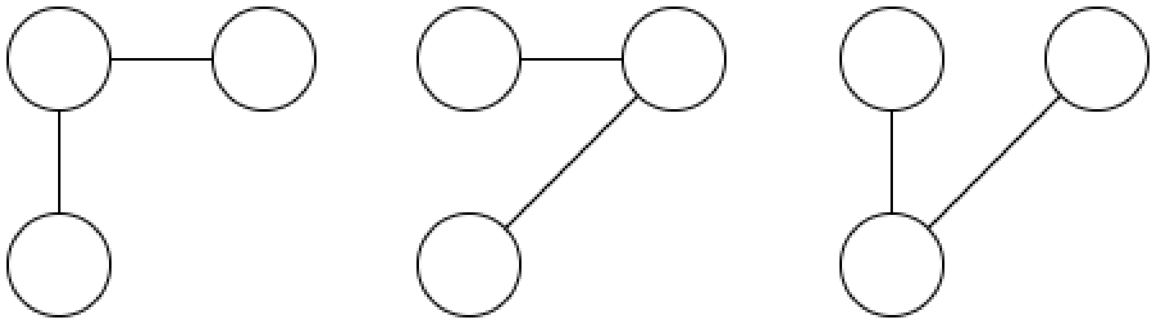

Figure 2: All MST for three nodes

D. 16 mst

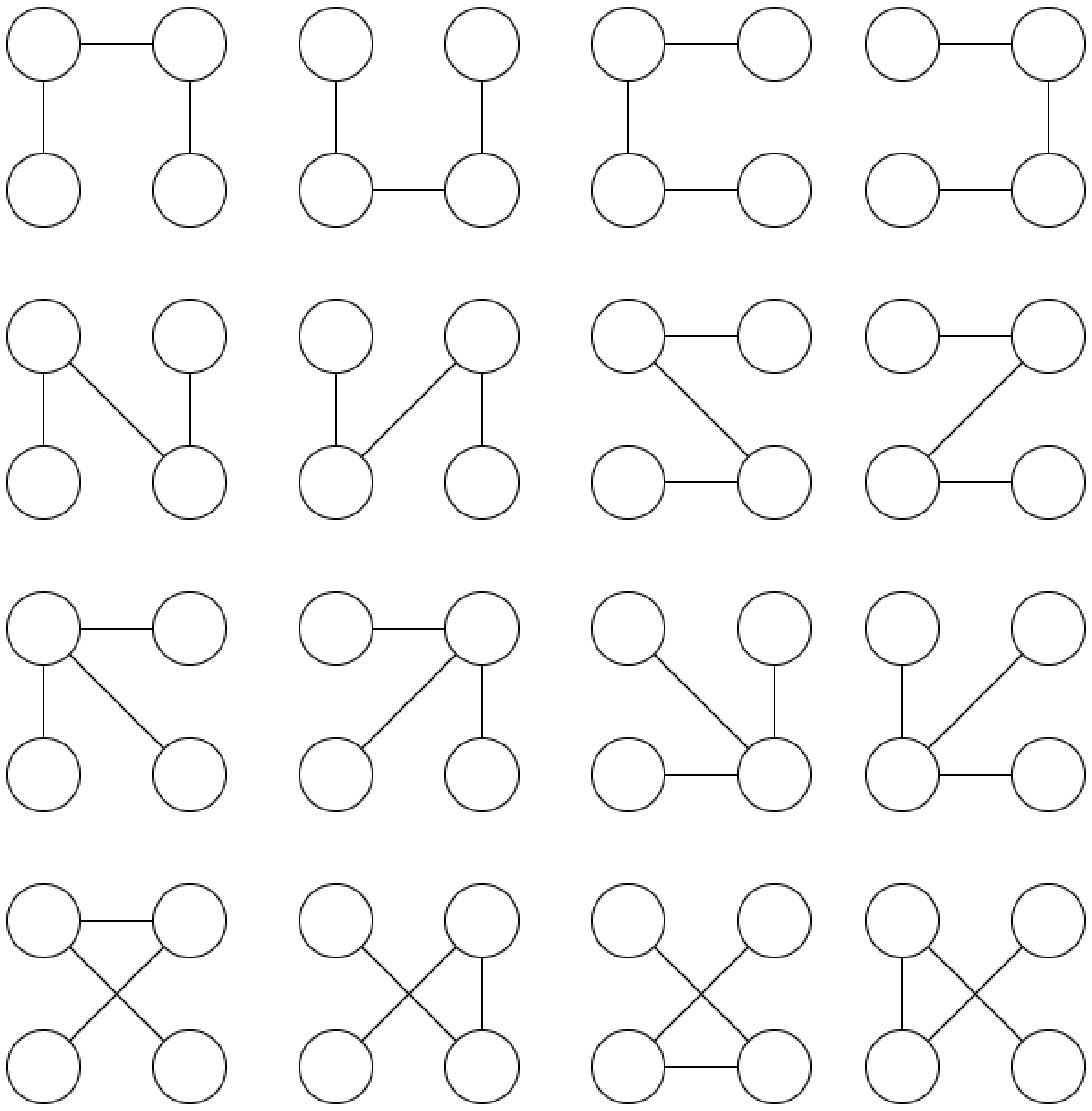

Figure 3: All MST for four nodes

E. V^{V-2}

Let T_n be a minimum spanning tree with n vertices. Let $T \in T_n$ with vertices labeled 1 through n in any manner.
Find the smallest vertex that has only one edge attached to it. For any finite minimum spanning tree with at least two vertices, two such vertices are guaranteed to exist. In a since, these vertices are the “start” and “end” of a path through the minimum spanning tree.
Remove this vertex and record its only neighbor.
Repeat this process until all nodes are recorded.
This process will give us a sequence of $n-1$ vertices. However, since the last term in this sequence is always n, we can remove it without loss of generality.
This results in a sequence of $n-2$ vertices encoding the minimum spanning tree.

Let the recorded vertices be P. We can reconstruct the minimum spanning tree from P through the following process.
Find the smallest vertex from 1 to n that does not exist in P. Link this vertex to the first vertex in P and remove the first vertex from P. Repeat this process until all vertices are removed from P.

Since there exists a bijective relation between P_n and T_n, they must have the same number of elements. It is known that $|P_n| = n^{n-2}$ so as a consequence, $|T_n| = n^{n-2}$

Source:
Prüfer Sequence
Cayley’s formula