Problem Set #7 – Report

Yessengaliyeva Gulnazym

Problem 1. Graph Play

1. Examples of directed graphs and their transposed graphs

For a directed graph $G = (V, E)$, the transpose G^T is obtained by reversing all edges:

$$(u, v) \in E \iff (v, u) \in E^T.$$

Example.

$$V = \{A, B, C, D\}, \quad E = \{(A, B), (B, C), (C, A), (C, D)\}$$

Then

$$E^T = \{(B, A), (C, B), (A, C), (D, C)\}.$$

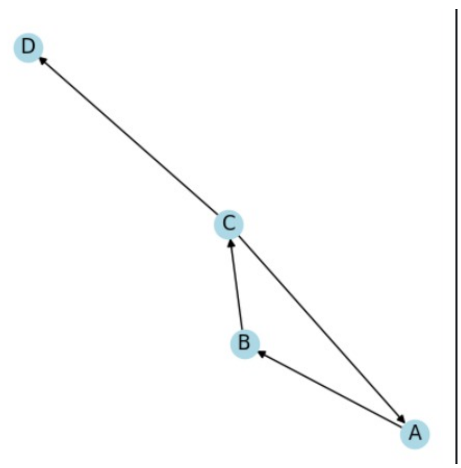

Figure 1: Directed graph G

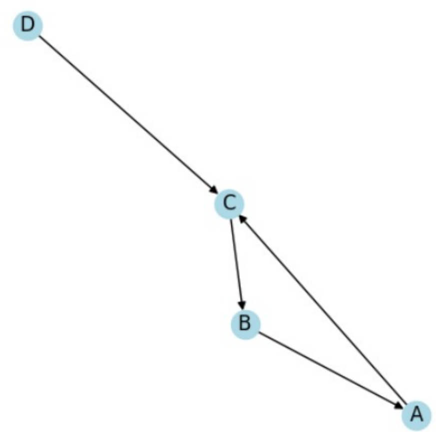

Figure 2: Transposed graph G^T

2. Examples of undirected graphs and their inverse graphs

For an undirected graph $G = (V, E)$, the inverse graph G^{-1} is defined as the graph with the same vertices and with all missing edges added (except self-loops):

$$uv \in E^{-1} \iff uv \notin E.$$

Example.

The figure below shows an undirected graph G_2 with vertices $\{1, 2, 3, 4\}$. All edges that are present in G_2 are removed in the inverse graph G_2^{-1}, and all edges that were missing are added.

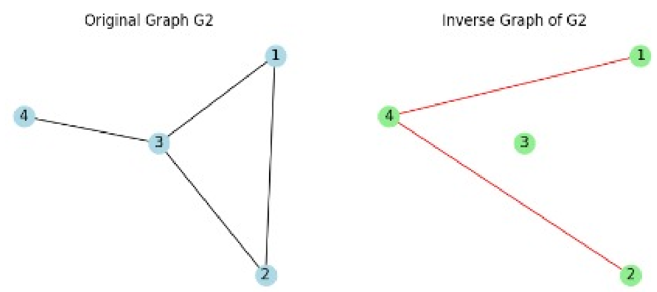

Figure 3: Left: original undirected graph G_2. Right: inverse graph G_2^{-1}.

In G_2, vertices $1, 2, 3$ form several edges, and vertex 4 is connected only to 3. In the inverse graph, every non-existing edge is included, so vertex 4 connects to 1 and 2, while edges from the original graph disappear.

3. What happens if the original graph is dense for the inverse?

For an undirected graph $G = (V, E)$, the inverse graph G^{-1} contains all non-existing edges:

$$uv \in E^{-1} \iff uv \notin E.$$

If G is **dense**, meaning it contains almost all possible edges, then only a few edges are missing. Therefore, the inverse graph G^{-1} becomes **sparse**.

Example.

A complete graph K_4 has:

$$E = \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\},$$

so all edges exist. Thus:

$$E^{-1} = \varnothing,$$

meaning the inverse graph has no edges at all.

This shows that the denser the original graph is, the fewer edges appear in the inverse graph.

4. Examples of undirected graphs and their dual graphs

The dual graph G^* of a planar graph G is defined by converting each face of G into a vertex, and adding an edge between two dual vertices whenever the corresponding faces share a boundary edge.

Example (Triangle).

A triangle has two faces:

$$F = \{\text{inside face}, \text{outside face}\}.$$

Thus the dual graph contains two vertices and three parallel edges, one for each edge of the triangle.

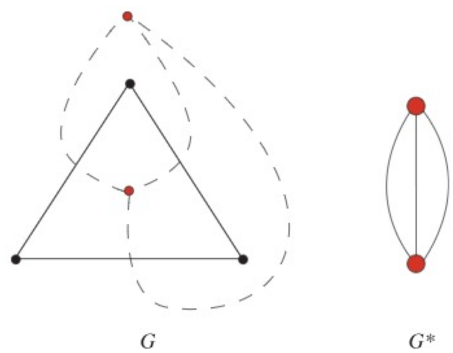

Figure 4: Left: a planar embedding of G. Right: its dual graph G^* with two vertices and three parallel edges.

5. Why is the dual only well-defined for planar graphs?

The definition of the dual graph G^* relies on the faces of a planar embedding of G. A dual vertex is created for each face, and dual edges represent boundaries between faces. Therefore, if a graph is **not planar**, its faces are not well-defined, because the graph cannot be drawn in the plane without edge crossings.

Thus, the dual graph exists **only** for planar graphs.

Example of a non-planar graph: the complete graph K_5.

K_5 has five vertices, each connected to all others:

$$E(K_5) = \{uv \mid u, v \in \{1, 2, 3, 4, 5\},\ u \neq v\}.$$

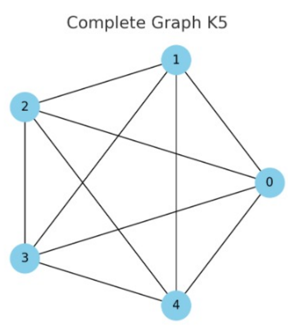

Figure 5: The complete graph K_5, a classical example of a non-planar graph.

It is a classical result that K_5 is non-planar and cannot be embedded in the plane without edge intersections. Since no planar embedding exists, the faces of K_5 are not defined. Therefore, the dual graph K_5^* does not exist.

Problem 2. Maximal Cliques with the Bron–Kerbosch Algorithm

We are given an undirected graph G with vertices $V = \{A, B, C, D\}$ and edges

$$E = \{\{A, B\}, \{A, C\}, \{B, C\}, \{C, D\}\}.$$

The task is to find all *maximal cliques* of G using the Bron–Kerbosch backtracking algorithm.

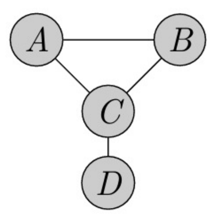

Figure 6: Undirected graph G used in Problem 2.

A **clique** is a set of vertices that are all pairwise adjacent. A clique is **maximal** if we cannot add any other vertex from G without breaking the clique property.

We apply the Bron–Kerbosch algorithm (version without pivot). It works with three sets:

R (current clique), P (potential candidates), X (already processed vertices).

Initially, $R = \emptyset$, $P = \{A, B, C, D\}$, $X = \emptyset$.

From this trace we obtain the following maximal cliques of G:

$$\boxed{\{A, B, C\} \text{ and } \{C, D\}}.$$

Clique $\{A, B, C\}$ is maximal because no other vertex can be added without losing the complete connectivity; similarly, $\{C, D\}$ is maximal.

Step	R	P	X	Comment
0	$\emptyset$	$\{A, B, C, D\}$	$\emptyset$	Initial call $\mathrm{BK}(R, P, X)$.
1	$\{A\}$	$\{B, C\}$	$\emptyset$	Choose $A \in P$. New call with $R = \{A\}$ and $P = N(A) \cap P = \{B, C\}$.
2	$\{A, B\}$	$\{C\}$	$\emptyset$	Choose $B \in P$. Now $P = N(B) \cap \{C\} = \{C\}$.
3	$\{A, B, C\}$	$\emptyset$	$\emptyset$	Choose $C \in P$. Now $P = N(C) \cap \emptyset = \emptyset$, $X = \emptyset$, so we **output clique** $\{A, B, C\}$.
4	$\{C\}$	$\{D\}$	$\{A, B\}$	Back at root, after processing A and B, we choose $C \in P$ and call $\mathrm{BK}(\{C\}, N(C) \cap P, N(C) \cap X) = (\{C\}, \{D\}, \{A, B\})$.
5	$\{C, D\}$	$\emptyset$	$\emptyset$	Choose $D \in P$. Now $P = N(D) \cap \emptyset = \emptyset$, $X = \emptyset$, so we **output clique** $\{C, D\}$.

Table 1: Short trace of the Bron–Kerbosch algorithm for Problem 2 (no pivot).

Maximum Clique

The largest clique is:

$$\boxed{\{A, B, C\}}.$$