TD 10: Chordal Graphs

1 Clique Cover

A **clique cover** of a graph $G = (V, E)$ is a partition of V into disjoint sets $V_1, V_2, \ldots, V_k$ such that for all $i \in \{1, \ldots, k\}$, $G[V_i]$ is a clique. The clique-cover number of G is the minimum number k such that G has a clique cover with k sets.

1. Observe that the clique-cover number of G is equal to $\chi(\overline{G})$.
2. Give a polynomial-time algorithm for computing the clique-cover number of a chordal graph.

2 A Tree Communication Problem

Consider the following problem: we are given a tree $T = (V, E)$ and a collection of pairs of vertices $(x, y) \in V^2$. Informally, T represents a communication network and the pair (x, y) encodes the fact that x wants to communicate with y. Recall that between any two $x, y \in V(T)$ there is a unique path. We want to assign colors to the pairs so that any two pairs that **interfere** with each other receive distinct colors, while using as few colors as possible.

1. Suppose that we say that (x_1, y_1) and (x_2, y_2) interfere when the unique path from $x_1 \to y_1$ and the unique path $x_2 \to y_2$ share a vertex. Show that in this case the minimum number of colors can be computed in polynomial time by constructing a chordal graph with one vertex for each communicating pair, an edge for each interference, and coloring that graph.
2. Show that if we instead define that two paths interfere when they share an **edge** (but sharing a vertex does not count as interference), then the intersection graph we obtain is **not** chordal.

3 Coloring on Chordal Graphs again

We saw in class a polynomial-time algorithm for computing the chromatic number of a chordal graph. We consider here an alternative version which does not rely on perfect elimination orderings but rather uses the fact that minimal separators are cliques.

Consider the following recursive algorithm which takes as input a graph G and an integer k and decides if $\chi(G) \leq k$. If G has at most k vertices, the algorithm replies Yes. Otherwise, if G is a clique, the algorithm replies No. Otherwise, the algorithm finds two vertices x, y which are non-adjacent and computes a minimal xy-separator S. Let $C_1, C_2, \ldots, C_t$ be the connected components of $G - S$. We recursively execute the same algorithm on each $G[S \cup C_i]$ for $i \in \{1, \ldots, t\}$. If the answer is No for any such sub-instance we reply No, otherwise we reply Yes.

Prove that the algorithm sketched above is correct and runs in polynomial time.

4 Tree Intersection Models

Recall that in Exercise 2 we defined a class of intersection graphs and proved that they are chordal. In this exercise we define a richer class of intersection graphs on a tree and prove that this class is actually **exactly** the class of all chordal graphs.

Consider a tree T and let $T_1, T_2, \ldots, T_k$ be a collection of sub-trees, that is, a collection of connected induced subgraphs of T. We define the intersection graph G of this collection as follows: (i) we have a vertex in G for each sub-tree T_i (ii) T_i and T_j are adjacent in G if and only if T_i and T_j have a vertex in common.

(Observe that if all the trees in our collection are paths, the intersection graphs we obtain are the same as those of Exercise 2.)

1. Prove that the intersection graphs that can be formed in this way are chordal.
2. Prove that all chordal graphs can be formed as intersection graphs in this way. (Hint: you will need to use Exercise 5 of TD2 on this regarding the Helly property of trees.)