Practice Problems - P vs NP

1. NP-hardness Result I

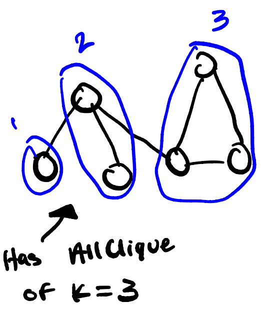
Reminder of what the *CLIQUE* problem is:

Decision Problem

- **Input:** An undirected graph $G = (V, E)$, and a positive integer k.
  Output: TRUE if G contains a set S of size $\geq k$ vertices where every pair of vertices in the set are connected by an edge

Let's define a new problem "ALL CLIQUES" (*ALLCLIQUE*):

- **Input:** An undirected graph $G = (V, E)$ and a positive integer k.
- **Output:** TRUE if there is a set S of $\geq k$ vertices where every pair of vertices in the set are connected by an edge and the graph G without S contains a *ALLCLIQUE* of size $k - 1$. (if $(k-1) == 0$, skip rec call)



(**Hint:** The recursive definition above ensures the graph has a disjoint CLIQUE of size x for all $x \leq k$)

"as hard as"
Polynomial time soln. for A
$\Rightarrow$ polynomial time soln. for B

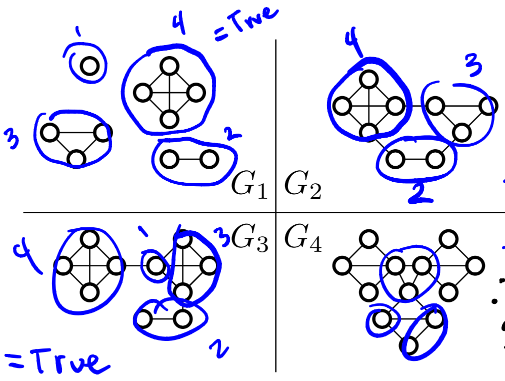
$$- \underset{\text{Clique}}{B} \leq_p \underset{\text{AllClique}}{A}$$

If this is solvable, then Clique is solvable

"polynomial time reduciable to"

∴ reduce first problem to 2nd problem
$\Rightarrow$ reduce B to find pA

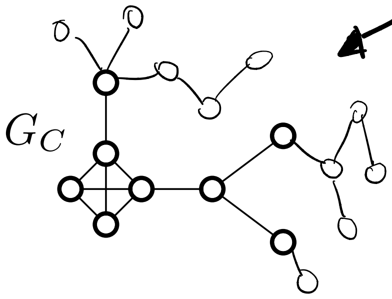
(1) (1.5 points) For which of the following graphs would *ALLCLIQUE* be true for $k = 4$.



∴ Missing a clique of size 1
= False

= False
∴ Missing a Clique of size 4

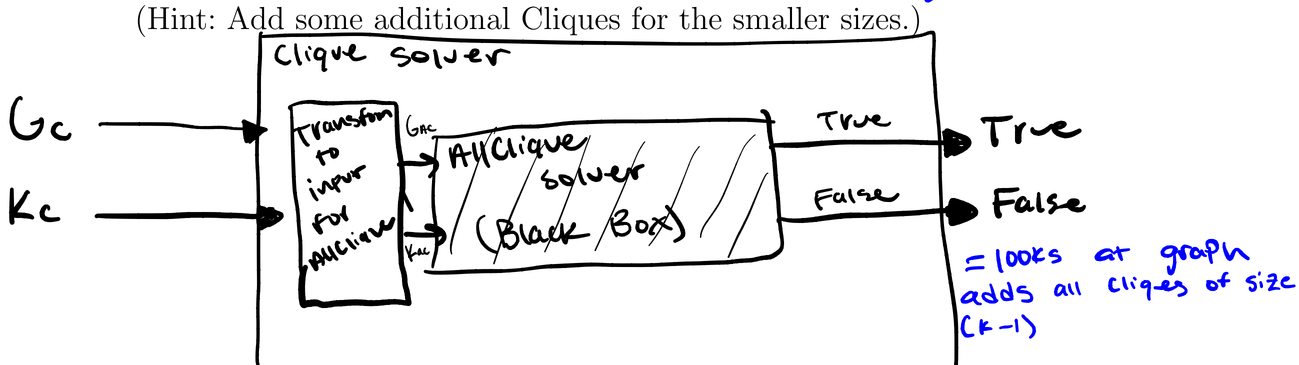
(2) (0.5 points) Below is a graph which contains an CLIQUE of size $k = 4$. What simple changes could you make to the graph to ensure *ALLCLIQUE* would return TRUE for $k = 4$?



Change so AllClique returns True when clique of size k exists in graph
= Add Clique of size 3

"Reduction"
$\Rightarrow$ solving AllClique helps solve Clique

(Hint: Add some additional Cliques for the smaller sizes.)


Clique solver
Gc
Kc
Transform to input for AllClique
G_{AC}
K_{AC}
AllClique solver (Black Box)
True
False
True
False

= looks at graph adds all cliques of size (k−1)

2. NP-hardness Result II

(1) Next we will show that $ALLCLIQUE$ is NP-hard by giving a polynomial time reduction from $CLIQUE$ to $ALLCLIQUE$ (i.e., $CLIQUE \leq ALLCLIQUE$):

Given an input graph $G_C = (V_C, E_C)$ and k_C to $CLIQUE$, we need to produce an input $G_{AC} = (V_{AC}, E_{AC})$ and k_{AC} to $ALLCLIQUE$.

Show how you would produce the following inputs to $ALLCLIQUE$ from your inputs to $CLIQUE$:

- $k_{AC} =$ k_c (same k as w/ Clique)
- $V_{AC} =$ Add to V_c enough vertices to make Cliques of size $k-1, k-2, k-3, \dots, 2, 1$
- $E_{AC} =$ Add to E_c the edges between our new vertices to make the Cliques of size $k-1, k-2, k-3, \dots, 2, 1$

3. NP-Hardness Reduction

Let's define a new problem: The satisfiability problem with at most 5 copies of each variable (SAT5C):

- Input: a boolean formula f with m clauses over n variables. No variable appears in more than 5 clauses.
- Output: TRUE if there is an assignment to the variables that makes the formula true, FALSE otherwise.

(1) (1 point) Is the following boolean formula satisfiable (justify your answer): $(\neg x_3 \vee x_2) \wedge (x_1 \vee x_3 \vee \neg x_2) \wedge (\neg x_3 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge (x_3 \vee \neg x_2)$

(2) (1 point) Is the above an example of the SAT5C problem? Why or why not?

(3) (4 points) Show that $SAT5C$ is NP-hard by giving a polynomial time reduction from SAT to $SAT5C$ (i.e., $SAT \leq SAT5C$):

(**Hint:** you can ensure two variables have the same truth value by adding 2 additional clauses each of which have two variables in them)

4. P vs. NP-hard

Let's imagine a small town has hired you as a software engineer to help them manage their road system. In this case, the town has a graph representing their road network.

Every vertex represents a residence or business. Every edge represents a road between two residences/businesses. Associated with it is both a length of the road as well as a maintenance cost.

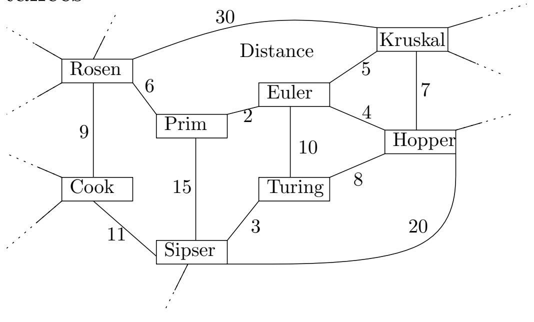


Fig. 1. Small example graph with distances

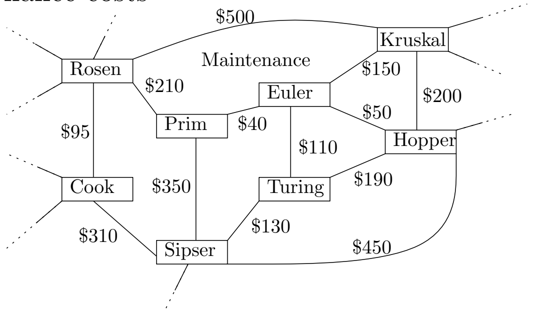


Fig. 2. Small example graph with maintenance costs

On the next page is a list of features the city wants you to look into adding. Specifically, identify whether the given problem can be solved in polynomial time. If it can, give a short description of an algorithm to compute it (feel free to use algorithms and data structures we learned earlier this semester in your justification). Alternatively, if you believe the problem is NP-hard, give a short justification why (this doesn't need to be a proof but try to point out a similar NP-hard problem).

You can assume there are n vertices and m edges in the graph.

(1) **Neighbors Distance I:** Find a set of k vertices which are all within distance 10 of each other.

(2) **Neighbors Distance II:** Find a set of 10 vertices which are all within distance k of each other.

(3) **Unused Roads I:** Suppose everyone always takes the shortest path when driving to someone else's house. Decide if there are at least k roads which no one drives on.

(4) **Unused Roads II:** The city is considering stopping maintenance on some unneeded roads. Find a set of subset roads with a total maintenance $\leq k$ such that every person can still reach every other person using only these roads.