6. $\boxed{\text{10 marks}}$ **(Take Home Question.)** A graph G is a *split graph* if $V(G)$ can be partitioned into set C and I such that C is a clique and I is an independent set in G.[1] In the SPLIT VERTEX DELETION problem, given a graph G and an integer k, the task is to check if one can delete at most k vertices from G to obtain a split graph.

Design a $2^k|V(G)|^{\mathcal{O}(1)}$ algorithm for SPLIT VERTEX DELETION, using iterative compression.

Lemma I : Let I, C be a partition of a split graph G s.t. I is a ind. set & C is a clique. Likewise, let I', C' be another distinct partition of G s.t. I' is ind set & C' is clique. $\exists$ At most one vertex in I not in I'.

Proof : Let $u, v \in I$, $u, v \notin I'$, $u \neq v$

Now, since $u, v \notin I'$, $\Rightarrow u, v \in C'$, $uv \in E(G)$. But this is a contradiction, and thus

Lemma II : The are at most n^2 partitions of G, and there exists a polytime algorithm to enumerate these.

Proof : Consider a partition I, C of G, I is an ind set & C is a clique. In any other partition (I', C') of G s.t. I' is ind. set & C' is clique, there is only one vertex in I not in I', & there are atmost $n-1$ such vertices. Also, there are at most n vertices in I' not in I. Thus there are a total of $(n-1)n + 1 \leq n^2$ partitions of G. It is trivial to find these given an initial partition - we can just enumerate over all possibilities for the "n^2" vertices described previously.

Solving DISJOINT SPLIT VERTEX DELETION in poly time

Input: Graph G, set W s.t. $G \setminus W$ is split graph, $|W| \leq k+1$

Output: Vertex set $S \subseteq V(G) \setminus W$, s.t. $|S| \leq k$, $G \setminus S$ is split graph. or conclude no such set exists.

Algorithm: 1. If $G[W]$ is not split graph, return No.

2. For each partition (C, I) of $G[W]$, C is clique, I is ind set

3. For each partition (C', I') of $G[V(G) \setminus W]$, C' is clique, I' is ind. set

4. Let $X \subseteq C'$ be set of all vertices in C' which are not incident on every vertex on C

5. Let $Y \subseteq I'$ be set of all vertices in I' which are incident on at least one vertex in I

6. If $|X \cup Y| \leq k$

Return $X \cup Y$

7. Return No.

Proof of correctness: Line 1 is correct since if $G[W]$ is not a split graph, $\Rightarrow \nexists$ $S \subseteq V(G) \setminus W$ s.t. $G[W]$ is split graph, so it is a No instance

Let the instance be a YES instance. Now, $\Rightarrow \exists$ set $S \subseteq V(G) \setminus W$ s.t. $G \setminus S$ is a split graph, and in $G \setminus S$, let C'' be the clique component component & I'' be the ind. set component. Now, $C'' \cap W$ and $I'' \cap W$ is a valid partition of $G[W]$, as $C'' \cap W$ is a clique & $I'' \cap W$ is a ind set.

In line 2, we iterate over all partitions of $G[W]$. So this will be considered. Let $C = C'' \cap W$ & $I = I'' \cap W$. Note that $\hat{C} = C'' \cap (V(G) \setminus W)$ is also a clique in $G \setminus (W \cup S)$ & $\hat{I} = I'' \cap (V(G) \setminus W)$ is also a ind. set in $G \setminus (W \cup S)$.

Since $G \setminus W$ itself is a split graph, there would exist a partition of $G \setminus W$, say C' & I', C' clique & I' ind set, such that $\hat{C} \subseteq C'$, $\hat{I} \subseteq I'$. Note that we would have found such a (C', I') in line 3.

For the partition C' in line 4 we remove all vertices which "prevent" $C' \cup C$ from being a clique. Note that we find the minimal such set X s.t. $G[(C' \cup C) \setminus X]$ is a clique.

Likewise in line 5, we find minimal Y s.t. $G[I' \cup I \setminus Y]$ is an ind. set.

Now, $|C' \setminus \hat{C}| + |I' \setminus I| \leq k$ [Since it is Yes instance]

$\Rightarrow$ $|X| + |Y| \leq k$ [Since X & Y are the minimal sets we remove]

Thus, $(X \cup Y)$ is also a valid (Disjoint Split Vertex Deletion) Solution

Runtime

By Lemma II, line 2 takes n^2 iteration & line 3 takes n^2 iterations, while line 4, 5 & 6 can be done in $O(n^2)$ time. Thus, the DISJOINT SPLIT VERTEX DELETION can be solved in $O(n^6) \simeq n^{O(1)}$ time.

SPLIT VERTEX DELETION

We know that using iterative compression, if the DISJOINT VERSION of the problem Π can be solved in $\alpha^k n^{O(1)}$ problem, the problem Π can be solved in $(1+\alpha)^k n^{O(1)}$ time [Taught in class and in book]

$\Rightarrow$ We can solve SPLIT VERTEX DELETION in $2^k n^{O(1)}$ time, using the algorithm for the disjoint version described above.

7. **(Take Home Question.)** A directed graph T with a special vertex $r \in V(T)$ is an *r-broadcast tree* if for each $u \in V(T)$, there is a directed path from r to u in T and $\boxed{|A(T)| = |V(T)| - 1}$. → Arc cardinality requirement.

In the BROADCAST DELETION problem we are given a directed graph D, a special vertex $r \in V(D)$ and an integer k, and the objective is to check if there is $S \subseteq V(D) \setminus \{r\}$ of size at most k, such that $D - S$ is an r-broadcast tree.

Design a $3^k \cdot |V(D)|^{O(1)}$-time algorithm for BROADCAST DELETION.

We will solve using iterative compression.

DISJOINT BROADCAST DELETION

Input: Given directed graph D, $r \in V(D)$, $W \subseteq V(D) \setminus \{r\}$ s.t. $D \setminus W$ is a r-broadcast tree, $|W| \leq k+1$. Let $X = V(D) \setminus (\{r\} \cup W)$

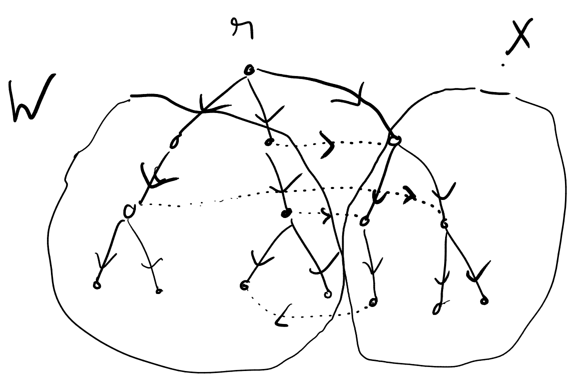
Output: Vertex set $S \subseteq X$, $|S| \leq k$, $D \setminus S$ is a r-broadcast tree, or correctly conclude $\nexists$ exist such a set S.

RR0: If $D[W \cup \{r\}]$ is not a r-broadcast tree, return "No".

Safeness: If $D[W \cup \{r\}]$ is not a r-broadcast tree, $\nexists$ a set $S \subseteq X$ s.t. $D[W \cup \{r\}]$ is a broadcast tree.

Observation:

Property 1 [Note that any r-broadcast tree has a tree as its underlying undirected graph, & the direction of the arcs in the r-broadcast is away from the root r. [This is the only way to maintain connectivity]]



We have 2 separate r-broadcast trees $D[X \cup \{r\}]$ & $D[W \cup \{r\}]$

The "problem" is arcs going across the two trees.

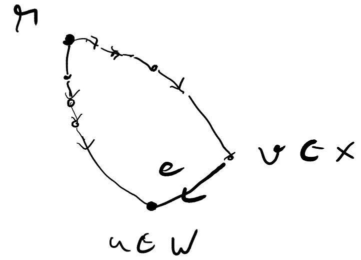
Let (D, W, k) be the DISJOINT BROADCAST DELETION instance.

Note: We are solving the deterministic version of DISJOINT BROADCAST DELETION herein, but it is trivial to return the solution S itself from this.

RR1: If $\exists\, e \in A(T)$ s.t. e is from a vertex $v \in X$ to a vertex in W, return $(D \setminus \{v\}, W, k-1)$.

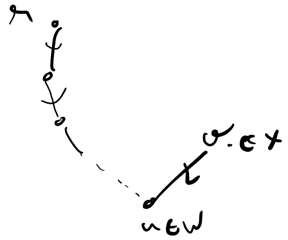
Safeness: Let by contradiction $v \notin S$, i.e. v is not deleted.

Case 1: None of v's ancestors in $D[\{r\} \cup X]$ is deleted



$\Rightarrow$ we have a undirected cycle in the underlying undirected graph of D. However, this is a contradiction since the underlying undirected graph is a tree. This violates property 1.

Case 2

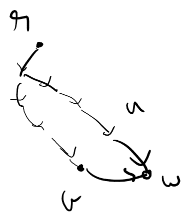


In this case, v is not reachable from r, & this is a contradiction.

Thus in both cases we have a contradiction. Thus, RR1 is safe.

RR3: Assume RR1 cannot be applied anymore. If $\exists\, uw, vw \in A(D)$, s.t. $u, v \in W$, $w \in X$, return $(D \setminus \{w\}, W, k-1)$

Safeness:



By contradiction let w be not deleted, i.e. $w \notin S$
The path from $(r-v)$, vw, wu, and the path from r to u all form a undirected cycle in the underlying undirected graph of the solution. This violates property 1.

Note that $D[X]$ is not a directed tree anymore but a directed forest.

Now, consider the subset $M \subseteq X$, $M := \{ v \in X : v \text{ has a directed edge from a vertex in } \{r\} \cup W \}$

This set M is the set of "marked" vertices

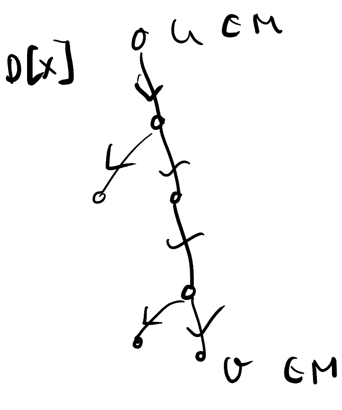
component refers to a component in the underlying undirected graph

RR3: In any component of the $D[X]$, remove all vertices which do not have an ancestor in M.

Safeness: Consider a vertex v s.t. it does not have an ancestor in M. Note that the vertices in M are reachable from "r" in D. Since $D[X]$ is a directed forest, descendants of M are also reachable from "r". However, since v does not have an ancestor in M, it is unreachable from r and hence has to be removed.

Observation 2

After RR0, RR1, RR2, RR3, $\forall$ vertex $v \in M$, we have a unique vertex say $f(v) \in W$ s.t. $f(v) - v$ is an arc in D. Also note that the root of each component of the directed tree is a marked vertex [by RR3].



Branching Rule Assume RR0, RR1, RR2, RR3 are not applicable.
Consider two vertices u & v, $u, v \in M$, u is the youngest ancestor of v. Let the parent of v in $D[X]$ be v'. Branch on $(G\setminus\{v\}, W, k-1)$, $(G'\setminus\{v'\}, W, k-1)$.

Correctness: Note the following undirected cycle: the path from r to $f(u)$, $f(u)-u$, the path from u to v, $v-f(v)$, the path from $f(v)$ to r. Since we have an undirected cycle in the underlying graph of D, at least one vertex in the path from u to v has to be deleted.

If v or v' is deleted, we are done.

Else, say some other vertex $z \in X$, was deleted in S [the soln]
$\Rightarrow$ v' does not have a marked ancestor, and thus v' is deleted anyways. Thus, $v' \in S$.

RR4: If $k < 0$, return "No". Otherwise continue applying the branching rule until we obtain a valid solution $S \subseteq X$ [It can be checked in poly time if $S \subseteq X$ is a valid soln.]

Safeness: If $k < 0$, clearly we removed $(k+1)$ or more vertices, and our solution cannot be a valid solution.

Lemma: RR0, RR1, RR2, RR3, Branching Rule, RR4 all generate a solution "S" of the DISJOINT BROADCAST DELETION, or correctly conclude ∄ exist such a solution.

Proof: Assume we did conclude "Yes" at the end of the algo, but our set 'S' is wrong. By contradiction, let there be a vertex u unreachable from r. If u is marked, u is reachable from r. Thus, u is unmarked.
$\to$ If u has a marked ancestor, it is reachable from r $\to$ a contradiction
$\to$ If u does not have a marked ancestor, we would have deleted it by RR3
— a contradiction.

Now, assume that the arc cardinality requirement is not met. Clearly since every vertex is reachable from r, there are some undirected cycles. Since there are no undirected cycles in $D[\{r\} \cup X]$ & $D[\{r\} \cup W]$, the undirected cycle must be across the two sets $\{r\} \cup W$ and X. There must be at least 2 arcs across $\{r\} \cup W$ and X, and the direction must be from $\{r\} \cup W$ to X for both the arcs [otherwise we would've applied RR1]. But now we can apply RR3 on the vertices of this cycle, clearly a contradiction.

Runtime : RR0, RR1, RR2, RR3, Branching Rule, RR 4 together form a bounded search tree of size at most 2^{k+1} and thus the DISJOINT BROADCAST DELETION can be solved in $2^k n^{O(1)}$ time.

We know that using iterative compression, if the DISJOINT VERSION of the problem Π can be solved in $\alpha^k n^{O(1)}$ problem, the problem Π can be solved in $(1+\alpha)^k n^{O(1)}$ time [Taught in class and in book]

$\Rightarrow$ We can solve BROADCAST DELETION in $3^k n^{O(1)}$ time, using the algorithm for the disjoint version described above.