All problems must be written up and turned in

1. You are given a directed graph \( G = (V, E) \) with unit edge capacities \( (c_e = 1, \text{ for all } e \in E) \). You also given nodes \( s, t \in V \) and a positive integer \( k \). The goal is to delete at most \( k \) edges in order reduce the maximum \( s \)-\( t \) flow in \( G \) by as much as possible. In other words, give an efficient algorithm to find a set of edges \( E' \subseteq E \) where \( |E'| \leq k \) such that the maximum \( s \)-\( t \) flow of \( G' = (V, E \setminus E') \) is minimized.

2. DPV 8.3: STINGY SAT is the following problem: given a set of clauses (each a disjunction of literals) and an integer \( k \), find a satisfying assignment in which at most \( k \) variables are true, if such an assignment exists. Prove that STINGY SAT is NP-complete.

3. DPV 8.13: Determine which of the following problems are NP-complete and which are solvable in polynomial time. In each problem you are given an undirected graph \( G = (V, E) \), along with:

   (a) A set of nodes \( L \subseteq V \), and you must find a spanning tree such that its set of leaves includes the set \( L \).

   (b) A set of nodes \( L \subseteq V \), and you must find a spanning tree such that its set of leaves is precisely the set \( L \).

   (c) A set of nodes \( L \subseteq V \), and you must find a spanning tree such that its set of leaves is included in the set \( L \).

4. For a graph \( G = (V, E) \), a set of nodes \( S \subseteq V \) is independent if no two nodes in \( S \) are joined by an edge. That is, \( \forall u, v \in S, (u, v) \notin E \). Consider the following problems.

   **INDEPENDENT SET**
   
   **Input:** An undirected graph \( G = (V, E) \) and a nonnegative integer \( k \).

   **Decision Problem:** Does \( G \) contain an independent set of size \( k \)?

   **BIGGER INDEPENDENT SET**

   **Input:** An undirected graph \( G = (V, E) \) and nodes \( S \subseteq V \) where \( S \) is an independent set.

   **Decision Problem:** Does \( G \) contain an independent set larger than \( S \)?

   Show that the BIGGER INDEPENDENT SET decision problem is NP-complete if the INDEPENDENT SET decision problem is NP-complete.

5. Suppose you wish to pack a list \( L \) of of \( n \) items each with some weight \( 0 < w_i \leq 1 \). You are packing the items into bins, each of which has as maximum weight capacity of 1. You are allowed to use any number of bins and the goal is to pack all the items using as few bins as possible.

   In the online version of this problem, your task is to pack items into bins one by one, as they arrive. You don’t know what weights are coming later and you are not permitted to change the bin assignment of an object after it has been packed.

   For a particular assignment of items to bins \( B_1, \ldots, B_m \), the sum of the \( w_i \) values packed into a bin \( B_j \) is called the content of \( B_j \), and is denoted by \( c(B_j) \). Note that \( 1 - c(B_j) \) is the empty space that the bin \( B_j \) has left. We call \( 1 - c(B_j) \) the gap of \( B_j \).

   Given a list \( L = (w_1, w_2, \ldots, w_n) \), define the Next-Fit heuristic algorithm as follows. Pack \( w_1, w_2, \ldots, w_i \) into \( B_1 \) until \( w_{i+1} > 1 - c(B_1) \), then \( w_{i+1}, w_{i+2}, \ldots, w_j \) into \( B_2 \) until \( w_{j+1} > 1 - c(B_2) \), etc. In other words, Next-Fit keeps packing items into the current bin until the next item doesn’t fit at which point it moves on to the next bin. Once a bin has no space for the next item, that bin is never used again.

   Let \( NF(L) \) be the number of bins used by the Next-Fit algorithm, and let \( L^* \) be the number of bins needed by an optimal packing. Show that as \( L^* \to \infty \), the ratio \( NF(L)/L^* \) is upper-bounded by 2.
6. DPV 9.7: In the MULTIWAY CUT problem, the input is an undirected graph $G = (V, E)$ and a set of terminal nodes $s_1, s_2, \ldots, s_k \in V$. The goal is to find the minimum set of edges in $E$ whose removal leaves all terminals in different components.

(a) Show that this problem can be solved exactly in polynomial time when $k = 2$.
(b) Give an approximation algorithm with ratio at most 2 for the case $k = 3$. 