COSC 311: ALGORITHMS
MINI 10

Due Wednesday, December 11 in class
(Prof. Rager's section: please drop off at one of our offices)

The *Set Packing* problem is defined as follows:

Input:

- A set U consisting of n elements
- m subsets of U, $S_1, \ldots, S_m$
- A positive integer k

Output: Is there a collection of at least k subsets such that no two subsets intersect?

The Set Packing problem can be seen as a generalization of Independent Set. In the Independent Set problem, we want to find a subset of nodes such that none of the nodes in our set are connected by an edge. Set Packing doesn't make any assumptions that the elements are organized into a graph structure or that conflicts between elements are explicitly encoded as edges. Nonetheless, both problems involve choosing a conflict-free set of items.

1. Give a polynomial-time reduction from Independent Set to Set Packing. Your answer should explain (1) given an instance of Independent Set, how do you turn it into an instance of Set Packing, and (2) how do you know that yes-instances of Independent Set map to yes-instances of Set Packing, and vice versa?

Solution: Our goal is to take an arbitrary instance of Independent Set and take polynomial time to turn it into an instance of Set Packing. We start with our input to Independent Set: an undirected, unweighted graph $G = (V, E)$ and a positive integer k. Our reduction is as follows:

- For every edge $(u, v) \in E$, create one element $x_{u,v}$ and add it to U.
- For every vertex $v \in V$, create one subset $S_v = \{x_{u,v} \in U : (u, v) \in E\}$. That is, we create a subset for vertex v that includes all of the elements in U corresponding to edges incident to v in the original graph.
- Set k in our Set Packing instance to be equal to k in our Independent Set instance.

We need to show that "yes" instances of Independent Set map to "yes" instances of Set Packing, and "yes" instances of Set Packing map to "yes" instances of Independent Set.

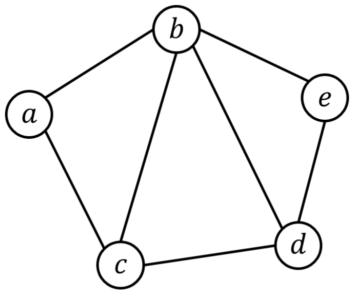
1) Suppose we start with an instance of Independent Set in which it is possible to find a set S of at least k vertices that are all independent. Then for every vertex $v \in S$, choose the corresponding subset S_v in our set packing instance. Consider any two subsets S_u and S_v that were chosen in this

way, and suppose that subsets S_u and S_v have an element in common. Then there must have been an edge between u and v in the original input graph to independent set. But this contradicts the fact that u and v were part of set S, and therefore independent. Hence S_u and S_v have no elements in common, and so the collection of subsets $\{S_v : v \in S\}$ is a collection of at least k subsets such that no two subsets intersect.

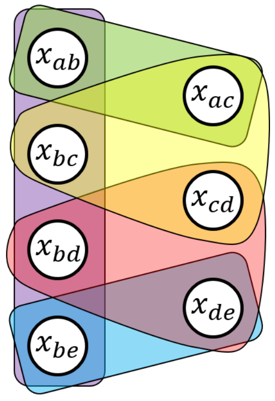
2) Suppose our instance of Set Packing contains a collection C of at least k subsets such that no two subsets intersect. Then consider nodes u, v such that $S_u, S_v \in C$. Suppose there is an edge between u and v in our original graph. Then element $x_{u,v}$ must exist in our Set Packing instance, and is included in both S_u and S_v. But this contradicts the fact that $S_u, S_v \in C$, since their inclusion in the Set Packing solution means that they have no elements in common. Hence edge (u, v) must not exist. This tells us that $\{v : S_v \in C\}$ forms an independent set of at least k nodes in the original graph.

Our reduction involves creating one element for each edge in our graph (of which there are m) and one subset for each node in our graph (of which there are n). We can do this in time $O(n + m)$.

2. Show how your reduction from part (a) turns the following instance of Independent Set, with $k = 2$, into an instance of Set Packing:



Solution: We end up with the following instance of Set Packing, with $k = 2$. There is a set packing of size 2 in this instance: for example, we could choose the yellow set and the blue set. This corresponds to an independent set of size 2 in our original graph: nodes c and e.



3. Fill in the blanks to indicate what the reduction from Independent Set to Set Packing tells us about the relationship between the two problems:

If Independent Set cannot be solved in polynomial time, then neither can Set Packing.

Explain why the above statement is true.

Solution: If Independent Set cannot be solved in polynomial time, then every single possible algorithm to solve Independent Set must take exponential time. One possible algorithm to solve Independent Set is:

1. Reduce Independent Set to Set Packing
2. Solve Set Packing

This algorithm must take exponential time. However since Independent Set $\leq_P$ Set Packing, we know that step 1 takes polynomial time, so step 2 must take exponential time. Hence there cannot exist a polynomial-time algorithm to solve Set Packing.

(Note that on the other hand, if Set Packing cannot be solved in polynomial time we do not learn anything about how long it might take to solve Independent Set. Reducing Independent Set to Set Packing and then solving Set Packing is just one possible algorithm to solve Independent Set; there could be some other algorithm that has nothing to do with Set Packing and that takes polynomial time.)