CS 330 – Summer 2019 – Lab 11

Question 1 (Set Packing)**.**
Given a universe $U = i_1, i_2, \ldots, i_n$ of n items and m subsets $S_j \subseteq U$, find the maximum number of subsets that are independent (that is, they don't have an item in common.). Prove the complexity of this problem.

Question 2 (Seating problem)**.**
Professor X is throwing a big party for his colleagues and is expecting n visitors. Lately some controversial issues have come up that resulted in many fights and some of his guests are not speaking to each other anymore. The professor wants to know whether he can seat his guests at a round table so that each guest is on speaking terms with both of her neighbors. We may assume that Professor X has exact knowledge about the friend-foe relationships among the guests. Prove the complexity of the Professor's task!

Question 3 (Resource reservations)**.**
Suppose you're consulting for a group that manages a high-performance real-time system in which asynchronous processes make use of shared resources. Thus the system has a set of n processes and a set of m resources. At any given point in time, each process specifies a set of resources that it requests to use. Each resource might be requested by many processes at once; but it can only be used by a single process at a time. Your job is to allocate resources to processes that request them. If a process is allocated all the resources it requests, then it is active; otherwise it is blocked. You want to perform the allocation so that as many processes as possible are active. Thus we phrase the Resource Reservation Problem as follows: Given a set of processes and resources, the set of requested resources for each process, and a number k, is it possible to allocate resources to processes so that at least k processes will be active? Consider the following list of problems, and for each problem either give a polynomial-time algorithm or prove that the problem is NP-complete.
(a) The general Resource Reservation Problem defined above.
(b) The special case of the problem when k = 2.