4. 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.

Text Text

(a) The general Resource Reservation Problem defined above.

(b) The special case of the problem when $k = 2$.

(c) The special case of the problem when there are two types of resources—say, people and equipment—and each process requires at most one resource of each type (In other words, each process requires one specific person and one specific piece of equipment.)

(d) The special case of the problem when each resource is requested by at most two processes.

same as independent set.

Solution: a) that is a NP-complete. Rephase the problem: For each process n_i in the the set n, and each resource in set m. For any n can request number of resoure $m_1 \cdots m_k$ in m. each m can only serve one n at a time. ① The issue here is the constract between finding a solution and checking a proposed solution. The checking algorithm needs two input. one is that 's, allocate scources to processes at least k. another input t that contain a "yes" instance of the problem to be active, we say $x.(x \in NP)$ ② for every string t a polynominal-time brueth-force algorithm to check. each k. so that all number of $t \leq p(|s|)$ that is yes. to satisfy "at least k" and it would take linear time to check. at least k processes can will be active. that is k processes need all resources are disjoint. otherwise. if is "no", that is not.

b). when $k = 2$. that can use brueth force algorith in linear time to check the each of n 2 pairs, and loop m. rescours, check whether they share one resources or not. if "yes", then it exists a $k=2$ processes with disjoint resources. otherwise. it dose not. exist.

c). when the process requires one more different resources, person and instruments. from part a. and the resources are independent each other. Thus. this limited case is still NP-complete.

d). same as part c). The limited case is still NP-complete.

10. Your friends at WebExodus have recently been doing some consulting work for companies that maintain large, publicly accessible Web sites—contractual issues prevent them from saying which ones—and they've come across the following *Strategic Advertising Problem*.

A company comes to them with the map of a Web site, which we'll model as a directed graph $G = (V, E)$. The company also provides a set of t *trails* typically followed by users of the site; we'll model these trails as directed paths $P_1, P_2, \ldots, P_t$ in the graph G (i.e., each P_i is a path in G).

The company wants WebExodus to answer the following question for them: Given G, the paths $\{P_i\}$, and a number k, is it possible to place advertisements on at most k of the nodes in G, so that each path P_i includes at least one node containing an advertisement? We'll call this the Strategic Advertising Problem, with input G, $\{P_i : i = 1, \ldots, t\}$, and k.

Your friends figure that a good algorithm for this will make them all rich; unfortunately, things are never quite this simple.

(a) Prove that Strategic Advertising is NP-complete.

(b) Your friends at WebExodus forge ahead and write a pretty fast algorithm S that produces yes/no answers to arbitrary instances of the Strategic Advertising Problem. You may assume that the algorithm S is always correct.

Using the algorithm S as a black box, design an algorithm that takes input G, $\{P_i\}$, and k as in part (a), and does one of the following two things:

- Outputs a set of at most k nodes in G so that each path P_i includes at least one of these nodes, *or*
- Outputs (correctly) that no such set of at most k nodes exists.

Your algorithm should use at most a polynomial number of steps, together with at most a polynomial number of calls to the algorithm S.

Step 1. decision problem. Give graph G. have different V, E. E ≥ V. & path on G. Given a set of nodes. a subset of nodes q. to place adds.

(A) NP. ≤ poly ~~longest~~ = t|E|

(B) reduce

$P_1 \cdots P_t$

t is node.

Solution: (A) Strategic Advertising (SA) is NP-complete. First, SA is NP. First we say a set of advertisement is "valid" if it covers all paths in $\{P_i\}$. ① Given a set of k nodes, we can check the k nodes in linear time and total P_i, say n. in O(kn) time. then we check whether it is a valid set of advertisement in time O(knt)

② we reduce Vertex cover $\leq_p$ SA. for the undirected graph G. we can produce directed $G' = (V, E')$ in Graph G. to make every edge with direction in polynomial time. we can define a path P_i in each edge in E'. we say G' has a valid set at most k advertisements if and only if G has a vertex cover of size at most k. then suppose G' does have such a valid set U; since it meet at least one end of each edge, → it is vertex cover for G. Thus, suppose G has a vertex cover T of size at most k; then the set T meets each path in $\{P_i\}$. and so it is a valid set of advertisements.

(B). ① By Induction. if k=1. we check whether there is any node that lies on all paths. otherwise. we check algorithm S whether there is a set of advertisement of size at most k. if not exist.

return "no". If "yes." we delete each node v and all paths through it. if there is another input that make a valid set of advertisements of size at most k-1. we say there is at least one node. v that make this test succeed. For any valid set u of at most. k advertisements exists. (we proved at least one exist): the test succeed on any $v \in u$, as $u - \{v\}$ is valid set of at most k-1 advertisements on the new input.

So. our algorithm will be done the conduction T by induction. complexity is $O(n+t)$ and call algorithm S need $O(n^2+nt)$ operations.

Question 3 (10 pts).

Suppose that someone gives you a black-box algorithm A that takes an undirected graph $G = (V, E)$, and a number k, and behaves as follows.

- If G is not connected, it simply returns G is not connected.
- If G is connected and contains a simple path of length at least k, it returns yes. if k. : "yes"
- If G is connected and does not contains a simple path of length at least k, it returns no.

BFS

Suppose that the algorithm A runs in time polynomial in the size of G and k. Show how, using calls to A, you could then solve the Longest Path Problem in polynomial time: Given an arbitrary undirected graph G, and a number k, does G contain a simple path of length at least k?

Solution: suppose $G=(V,E)$ undirected graph, after connected the ~~two~~ graph to be a graph $G'=(V,E')$
for G has components, $G_2, G_3 \dots G_K$, $K>1$, so that $G=G'$ any component i, check all the path use the Black-box Algorithm if it exist a path k in a component, return "yes" after check all the component in linear time, if it is at least one of them in length k, return "yes" otherwise return "no"

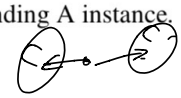
②. suppose a graph $G=(V,E)$ is undirected. we connected graph $G'=(V,E)$.
let's say G does not have any component at least k, that is not possible has k simple path that connected G. it return "no"

Complexity: use linear time to check every components by black-box. is polynominal time.

Question 4 (10 pts).

A company has two trucks, and must deliver a number of packages to a number of addresses. They want both drivers to be home at the end of the day. This gives the following decision problem: You are given a set of locations L, with for each pair of locations $v, w \in L$, a distance $d(v, w)$, a starting location $s \in L$, and an integer k. Find two cycles, that both start in s, such that every location in L is on at least one of the two cycles, and both cycles have length at most k? Show that this problem is NP-complete.

When you reduce problem A **to** problem B (does prove $A \leq_p B$), for full credit you need to first identify what constitutes an instances of A, then define the corresponding instance of B, finally you need to prove that there is a one-to-one correspondence between the solution to the instance of A and the solution to the instance of B. Note, that for all instances of A, there needs to be a corresponding instance of B, but this does not apply the other way around, there may B instances of B that do not have a corresponding A instance.



Solution: It can be reduced by Hamiltonian cycle (HC) constitutes of HC. Saleman, s, visits cities, $v_1\ v_2 \cdots v_n$, start from v_1, and visit all the cities, then return to home. for each order pair of cities, By the given set of distances. : order the cities into a tour $v_{i_1}, v_{i_2}, \ldots v_{i_n}$, with $i_1 = 1$. minimize the total distance $\sum_j d(v_{i_j}, v_{i_{j+1}}) + d(v_{i_n}, v_{i_1})$, the tour starts at home city, and the terms in sum simply give the distance from each city on the tour to the next one. ① TSP is NP with the certificate is a permutation of the cities, and a certifier checks that the length of the corresponding tour is at most the given bound.

② Hamiltonian cycle $\leq_p$ TSP. Given a direct graph $G=(V,E)$. we define two city $d(V_i', V_j')$ with distance of edge 1. for n cities, by Hamilton cycle, the tour of length at most n. that corresponding the cities. in TSP. Thus. the TSP must form Hamiltonian cycle. Conversely suppose the tour length at most n. Thus. The expression for the length of this tour is a sum of n terms, each of term is at least 1. so all the terms are equal to 1. Hence each pair of nodes in G that correspond to consecutive cities on the tour must be connected by edge: the orders of these corresponding nodes must form a Hamiltonian cycle. Thus. TSP is np-complete.

For the problem of 2 trucks and all the locations (cities), as we proved in TSP. problem.

Hamiltonian cycle $\geq_p$ 2-TSP. Suppose. two trucks A, B travel a set of locations L. We add one city that distance is

$\frac{k}{2}$ from home. then the distance of truck B is k. no other city will assign to truck B. ② for truck A, suppose A travel all cities. with TSP method. thus. drive A will solve all the problem. we can set all the path from home to cities and pair of city $d(v, w)$ is very big, so the sum of length in every pair of cities can be infinite. thus. the running time is $O(n)$. both cycle have length equal or less than k.

By the reduce of Hamilton cycle. to the 2 trucks. is polynomial time, and also one to one corresponce. from Hamilton cycle to 2 truck problems.