INFO 6205 – Program Structures and Algorithms
Assignment 4

Student Name: Vidhi Bharat Patel
Professor: Nik Bear Brown

Q1 (25 Points) Given a directed graph G = (V,E), a cycle-cover is a set of vertex-disjoint cycles so that each vertex v € *V* belongs to a cycle. On other words, a cycle cover of a graph G is a set of cycles which are sub-graphs of G and contain all vertices of G. If the cycles of the cover have no vertices in common, the cover is a called vertex-disjoint cycle cover or simply a disjoint cycle cover.

The *vertex-disjoint cycle-cover problem* asks whether a given directed graph has a vertex-disjoint cycle cover.

A. (10 points) Is the vertex-disjoint cycle-cover problem in P? If so, prove it.

- No, the vertex-disjoint cycle-cover problem is not known to be in P. In fact, it is a well-known NP-hard problem.
- To see why this is the case, consider the following reduction from the Hamiltonian cycle problem to the vertex-disjoint cycle-cover problem:
- Given an instance of the Hamiltonian cycle problem, which asks whether a given undirected graph has a Hamiltonian cycle (a cycle that visits every vertex exactly once), we construct a directed graph by replacing each edge in the undirected graph with two directed edges in opposite directions. This directed graph has a Hamiltonian cycle if and only if the original undirected graph had a Hamiltonian cycle.
- Now suppose we could solve the vertex-disjoint cycle-cover problem in polynomial time. Then we could use this algorithm to solve the Hamiltonian cycle problem as follows:
    1. Construct the directed graph as described above.
    2. Use the polynomial-time algorithm for vertex-disjoint cycle-cover to determine whether the graph has a vertex-disjoint cycle cover.
    3. If the graph has a vertex-disjoint cycle cover, then the original undirected graph had a Hamiltonian cycle. Otherwise, it did not.
- Since the Hamiltonian cycle problem is known to be NP-hard, this reduction implies that the vertex-disjoint cycle-cover problem is also NP-hard, and therefore not known to be in P.

B. (5 points) Suppose we require each cycle to have at most three edges. We call this the *3-cycle-cover problem.* Is the 3-cycle-cover problem in NP? If so, prove it.

- Yes, the 3-cycle-cover problem is in NP.
- To prove that the 3-cycle-cover problem is in NP, we need to show that given a candidate solution, we can verify in polynomial time whether it is a valid solution or not.
- Given a directed graph G = (V, E), and a candidate solution which is a set of vertex-disjoint cycles, we need to verify if each cycle in the solution has at most three edges and if the cycles together cover all vertices of G.
- To verify if each cycle has at most three edges, we can simply check the number of edges in each cycle and compare it with three. This can be done in O(E) time by iterating over all the edges in the graph.
- To verify if the cycles together cover all vertices of G, we can simply check if the set of vertices covered by the cycles is equal to V. This can be done in O(V) time by iterating over all the vertices in the graph.

- Therefore, the 3-cycle-cover problem is in NP, as we can verify in polynomial time whether a given candidate solution is valid or not.

C. (10 points) Is the *3-cycle-cover problem* in NP-complete? If so, prove it.

- To show that the 3-cycle-cover problem is NP-complete, we need to show that it is in NP and that it is NP-hard. A problem is in NP if there is a polynomial time algorithm to verify a "yes" answer given a solution, and the 3-cycle-cover problem satisfies this property.
- To prove that the 3-cycle-cover problem is NP-complete, we need to reduce another known NP-complete problem to it in polynomial time.
- One such problem is the vertex-cover problem. We can reduce the vertex-cover problem to the 3-cycle-cover problem by constructing a new graph G' from G, where every vertex in G' represents a triangle in G, and every edge in G' represents a pair of adjacent triangles in G.
- Then, we can show that G has a vertex cover of size k if and only if G' has a 3-cycle-cover of size k. If G has a vertex cover of size k, then we can construct a 3-cycle-cover of size k in G' by selecting a triangle for each vertex in the vertex cover and taking the edges in G' that correspond to adjacent pairs of triangles.
- Conversely, if G' has a 3-cycle-cover of size k, then we can construct a vertex cover of size k in G by selecting one vertex from each triangle in the 3-cycle-cover.
- Since the reduction can be done in polynomial time and the vertex-cover problem is known to be NP-complete, it follows that the 3-cycle-cover problem is also NP-complete.

Q2 (20 Points) The *Directed Disjoint Paths Problem* is defined as follows. We are given a directed graph G and k pairs of nodes $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. The problem is to decide whether there exist node-disjoint paths $P_1, P_2, \ldots, P_k$ so that P_i goes from s_i to t_i.
Show that *Directed Disjoint Paths* is NP-complete.

- To show that the Directed Disjoint Paths Problem is NP-complete, we can use a reduction from the well-known NP-complete problem of Hamiltonian Path, which is defined as follows: given a graph G and a starting node s, does G contain a simple path that visits every node exactly once and ends at s?
- We can reduce the Hamiltonian Path problem to the Directed Disjoint Paths Problem as follows. Given a graph G and a starting node s for the Hamiltonian Path problem, we can construct a new graph G' as follows:
  1. For each node v in G, create two new nodes v1 and v2 in G'.
  2. For each edge (u,v) in G, create an edge (u2,v1) in G'.
  3. For each node v in G, create an edge (v1,v2) in G'.
  4. For each node v in G, create an edge (v2,s1) in G' if v is not the starting node s, or an edge (v2,s2) in G' if v is the starting node s.
- Now, consider the k pairs of nodes (s1,t1), (s2,t2), ..., (sk,tk) in G', where si and ti are the corresponding nodes in G' for the ith node in G. We can see that G contains a Hamiltonian path from s to s if and only if G' contains k node-disjoint paths from si to ti for all i.
- Since the Hamiltonian Path problem is NP-complete, and we have shown that it can be reduced to the Directed Disjoint Paths Problem in polynomial time, it follows that the Directed Disjoint Paths Problem is also NP-complete.

Q3 (20 Points) You are organizing a game hack-a-thon and want to make sure there is at least one instructor who is skilled at each of the n skills required to build a game (e.g. programming, art, animation, modeling, artificial intelligence, analytics, etc.) You have received job applications from m potential instructors. For each of n skills, there is some subset of potential instructors qualified to teach it. The question is: For a given number $k \leq m$, is it possible to hire at most k instructors that can teach all

of the n skills. We'll call this the *Cheapest Teacher Set*.
Show that *Cheapest Teacher Set* is NP-complete.

- To show that Cheapest Teacher Set is NP-complete, we need to show that it is both in NP and NP-hard. First, let's show that Cheapest Teacher Set is in NP.
- Given a set of potential instructors and their qualifications, and a number k, we can easily verify if a given set of k instructors can teach all n skills by checking if each skill is covered by at least one instructor in the set. This verification can be done in polynomial time, therefore Cheapest Teacher Set is in NP.
- Next, we need to show that Cheapest Teacher Set is NP-hard. We will do this by reducing the Set Cover problem to Cheapest Teacher Set. Given an instance of the Set Cover problem with a universe U and a collection S of subsets of U, we can construct an instance of Cheapest Teacher Set as follows:
  1. Let n = |U| be the number of elements in the universe.
  2. Let there be n skills, one for each element in U.
  3. For each subset Si in S, create an instructor with skills corresponding to the elements in Si.
  4. Set k = |S|.
- Now, the Cheapest Teacher Set problem asks if we can hire at most k instructors such that they collectively cover all n skills, which is equivalent to asking if we can select at most k subsets from S such that their union covers U.
- Therefore, an algorithm that solves Cheapest Teacher Set can be used to solve Set Cover, and since Set Cover is known to be NP-hard, Cheapest Teacher Set is also NP-hard. In conclusion, since Cheapest Teacher Set is in NP and NP-hard, it is NP-complete.
- For example: we have 3 skills required to build a game - programming, art, and animation. We have received job applications from 4 potential instructors: Alice, Bob, Charlie, and Dave. The table below shows which skills each instructor is qualified to teach: -

Instructor	Programming	Art	Animation
Alice	Yes	Yes	No
Bob	No	Yes	Yes
Charlie	Yes	No	Yes
Dave	No	Yes	Yes

- We need to find out if we can hire at most k instructors, who can teach all three skills. Let's say k=2. One possible solution would be to hire Alice and Bob. Together, they can teach all three skills. However, if k=1, then we cannot find a solution, because no single instructor is qualified to teach all three skills.
- To prove that this problem is NP-complete, we can reduce the 3-SAT problem to this problem. In the 3-SAT problem, we are given a Boolean formula in conjunctive normal form (CNF) with 3 literals per clause, and the question is whether there exists a truth assignment to the variables that makes the formula true.
- We can construct an instance of the Cheapest Teacher Set problem as follows: for each variable in the 3-SAT formula, we create a skill. For each clause in the formula, we create an instructor who can teach the three literals in the clause.
- We can then set k to be equal to the number of clauses in the formula. If we can find a set of at most k instructors that can teach all the skills, then we have found a satisfying truth assignment for the 3-SAT formula.
- Conversely, if we cannot find such a set, then there is no satisfying truth assignment. Therefore, the Cheapest Teacher Set problem is NP-complete.

Q4 (20 Points) Suppose you're helping to organize a summer sports camp, and the following problem comes up. The camp is supposed to have at least one counselor who's skilled at each of the *n* sports

covered by the camp (baseball, volleyball, and so on). They have received job applications from *m* potential counselors. For each of the *n* sports, there is some subset of the *m* applicants qualified in that sport. The question is: For a given number $k < m$, is it possible to hire at most *k* of the counselors and have at least one counselor qualified in each of the *n* sports? We'll call this the *Efficient Recruiting Problem*.
Show that Efficient Recruiting is NP-complete.

- To explain the Efficient Recruiting problem, we can take up the example of organizing a summer sports camp which will cover three sports: basketball, soccer, and tennis. And you have received job applications from six potential counselors, with their qualifications for each sport shown below:

Counselor	Basketball	Soccer	Tennis
A	Yes	No	Yes
B	Yes	Yes	No
C	No	Yes	Yes
D	Yes	No	No
E	No	Yes	No
F	No	No	Yes

- For each of the three sports, there is a subset of the counselors who are qualified to teach it. Specifically, for basketball, counselors A, B, and D are qualified. For soccer, counselors B, C, and E are qualified; and for tennis, counselors A, C and F are qualified.
- Now suppose you want to hire at most two counselors and ensure that each of the three sports has at least one qualified counselor. The question is whether this is possible.
- One possible solution is to hire counselors B and C. This way, you we have a qualified counselor for each of the three sports: B for basketball and soccer, and C for tennis.
- To see that this is indeed the optimal solution, note that any solution with only one counselor cannot cover all three sports, since no single counselor is qualified for all three. Therefore, we need at least two counselors.
- Moreover, we cannot hire counselors A, D, E, or F, since they are not qualified for at least one of the three sports. This leaves us with counselors B and C as the only viable candidates, and they indeed provide a solution to the problem.
- This example illustrates how the Efficient Recruiting problem involves finding a subset of qualified counselors to hire, subject to certain constraints, and how it can be solved through careful consideration of the qualifications and constraints.

Q5 (15 Points) Suppose you live with $n - 1$ other people, at a popular off-campus cooperative apartment, the *Ice-Cream and Rainbows Collective*. Over the next *n* nights, each of you is supposed to cook dinner for the co-op exactly once, so that someone cooks on each of the nights.

Of course, everyone has scheduling conflicts with some of the nights (e.g., algorithms exams, Miley concerts, etc.), so deciding who should cook on which night becomes a tricky task. For concreteness, let's label the people, $P \in \{p_1, \ldots, p_n\}$, the nights, $N \in \{n_1, \ldots, n_n\}$ and for person p_i, there's a set ofnights $S_i \subset \{n_1, \ldots, n_n\}$ when they are not able to cook. A person cannot leave S_i empty.

If a person isn't doesn't get scheduled to cook in any of the n nights they must pay $200 to hire a cook.

A (10 Points) Express this problem as a maximum flow problem that schedules themaximum number of matches between the people and the nights.

- We can express this problem as a maximum flow problem in the following way:

1. Create a source node s and a sink node t.
2. For each person pi, create a node pi' and connect s to pi' with an edge of capacity 1.
3. For each night nj, create a node nj' and connect nj' to t with an edge of capacity 1.
4. For each person pi and each night nj not in Si, create an edge between pi' and nj' with capacity 1.
5. For each person pi not scheduled to cook on any night, create an edge between pi' and t with capacity n-1.

- The idea behind this construction is that each edge between a person and a night represents a possible assignment of that person to that night, and has capacity 1 to ensure that each person is assigned to at most one night.
- The edges from s to the people nodes and from the night nodes to t ensure that each person is assigned to exactly one night, and each night is assigned to exactly one person.
- The edges from a person not scheduled to cook on any night to t ensure that they pay the penalty if they are not assigned to any night.
- By finding the maximum flow in this network, we can determine the maximum number of matches between the people and the nights, which corresponds to the maximum number of people who can be assigned to cook on different nights.

B (5 Points)Can all n people always be matched with one of the n nights? Prove that it can or cannot.

- It is not always possible to match all n people with one of the n nights.
- To see why, consider the case where each person has exactly one night when they are unable to cook, and each person has a different night that they cannot cook on. In this case, there is no way to schedule everyone to cook, as each person is unavailable on a different night, and there are no extra nights left to assign to them.
- For example, suppose we have 3 people (p1, p2, p3) and 3 nights (n1, n2, n3), and each person is unable to cook on one particular night as follows:
  1. P1 cannot cook on night n1
  2. P2 cannot cook on night n2
  3. P3 cannot cook on night n3
- In this case, there is no way to schedule all three people to cook, as each person is unavailable on a different night. One possible schedule is:
  1. P1 cooks on night n2
  2. P2 cooks on night n3
  3. P3 cooks on night n1
- However, this schedule leaves no cook available on night n2, n2, and n1, respectively. Thus, at least one person must pay $200 to hire a cook.
- Therefore, it is not always possible to match all n people with one of the n nights.