Algorithms for Big Data
Polynomial-time Reductions

Professor: Nik Bear Brown

Q1

WebFlix maintains customer data in a 2D-array called WF. Where the rows correspond to the customers and the columns correspond to films that it rents. An entry WF[i,j] indicates the number of times a customer has rented a film.

WebFlix wants to find subsets of customers who have never rented the same film. (i.e. they share no entries WF[i,j] $\geq$ 1). WebFlix calls these Distinct Customer Subsets. We define the Distinct Customer Subset problem as follows: Given a *c* by *f* (customers by films) array of customers and films and a number k $\leq$ c, is there a subset of at least k customers that is *distinct?*

A. Is the Distinct Customer Subset problem NP? Why or why not?
B. Is the Distinct Customer Subset problem NP-complete? If NP-complete show a polynomial-time reduction.

Solution:

Part A (5 Points)

Is the Distinct Customer Subset problem NP? Why or why not?

Solution:

Yes. Because it can be verified by a nested loop. For every film count the number of customers who rented if the number is greater than 1 return "not a solution". After the loop is complete the solution has been verified. The nested loop is polynomial because it runs in O(C F).

Part B (15 Points)

Is the Distinct Customer Subset problem NP-complete? If NP-complete show a polynomial-time reduction.

Solution:

Independent set can be reduced to this problem. Where every node represents a different customer and every edge represents a different film. Any node incident to an edge means that that customer has seen that film. That implies that any 2 nodes connected by the same edge have seen the same film and therefore they can't be in the same subset. A solution to this problem for at least k customers would solve the independent set problem for at least k nodes.

Q2

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 ≤ m, is is possible to hire at most k instructors that can teach all of the k skills. We'll call this the Cheapest Teacher Set.

Is the Cheapest Teacher Set NP-complete? If NP-complete show a polynomial-time reduction.

Solution:

The problem is in NP. We can easily find a poly-time certifier. That is, IF given a solution for an NP problem you could verify it yes or no in polynomial time.

Poly-time certifier

Given a set of k instructors we can easily check if they cover the n skills just by putting the n skills in a list and then looping thru the k instructors and removing any skills they have from the list. If the list is empty then the k instructors have covered the list and we return yes. If we loop thru all k instructors and any skills remain in the list then we return no.

Recipe to establish NP-completeness of problem Y.
Step 1. Show that Y is in NP.
Step 2. Choose an NP-complete problem X.
Step 3. Prove that X ≤p Y.

Poly-time reduction

For this problem both Set Cover and Vertex Cover make straightforward reductions to Cheapest Teacher Set. In Vertex Cover we can define an edge as a skill and a node as an instructor. If k nodes (instructors) cover all n edges (skills) then those k ≤ m instructors form a Cheapest Teacher Set as the solution to the Vertex Cover gives us a minimal k.

http://en.wikipedia.org/wiki/Vertex_Cover

Likewise Set Cover is just as straight-forward. Here the sets are the skills of each instructor. The minimal set cover k ≤ m that covers all n skills form a Cheapest Teacher Set.

http://en.wikipedia.org/wiki/Set_cover_problem

Q3

Brainiac.com is creating a paywall (a system that prevents Internet users from accessing certain webpage content without a paid subscription). The site is modeled as a directed graph G = (V,E) in which nodes are webpages and edges are hyperlinks. The homepage is accessible to everyone and the rest of the sites is split into k zones (Z_1,Z_2 … Z_k) in which each page (node) belongs to a single zone. A path taken by a non-subscriber is restricted to include at most one node (page) from each zone.

You've been hired to develop an algorithm to analyze the reachability of pages on Brainiac.com. Specifically the question: Is it possible for a non-subscriber to visit a given page, p? More precisely, we define the *Non-payer Path* problem as follows: Given G, Z_1,Z_2 … Z_k (k zones), a start page, s € V and a target page, t € V; is there an s-t path that includes at most one page from each zone.

Is the *Non-payer Path* problem NP-complete? If so, prove it. If not, come up with a polynomial-time algorithm to find an s-t path.

Solution:

Note: I gave up to 17 points for a well thought out polynomial-time algorithm to find an s-t path. But we are not checking for a single path.

The *Non-payer Path* is in NP since we may check, for an s-t path P that includes at most one page from each zone simply by using a dictionary with the zone as a key and a count as a value. If we have a count >1 or do not reach t then we return no, otherwise yes.

The *Non-payer Path* is NP-complete as we can reduce it to 3-SAT.

3-SAT reduction of *Non-payer Path*:

Now let us suppose that we have an instance of 3-SAT with n variables x_1, …. x_n and t clauses. We create the following directed graph G = (V, E). The vertices will be partitioned into layers, with each node in one layer having edges to each node in the next. Layer 0 will consist only of the node s. Layer i will have two nodes for i = 1, …., n, three nodes for i = n+1, …. n+t, and only the node t in layer n+t+1. Each node in layers 1, 2, …... n+t will also be assigned a label, equal to one of the 2n possible literals. In layer i, for 1 ≤ i ≤ n, we label one node with the literal x_i and one with the literal not x_i. In layer n + i, for 1≤ i ≤ t, we label the three nodes with l_{i1},l_{i2}, l_{i3}, where {l) are the three literals appearing in clause i. Finally, for every pair of nodes whose labels corresponded to a variable and its negation, we define a distinct zone Z.

Now, if there is a satisfying assignment for the 3-SAT instance, we can define a non-payer s-t path P that passes only through nodes with the labels of literals set to *true*; P is thus a *Non-payer* path. Conversely, consider any *Non-payer* path P; we define variable x_i to be true if P passes through the vertex in layer i with label x_i and false if P passes through the vertex in layer i with label not x_i. Since P does not visit any zone a second time, it must therefore visit only nodes whose labels correspond to literals set to *true*, and hence the 3-SAT instance is satisfiable.

Q4

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

Solution:

a. Assume graph $G = (V, E)$ has vertex set V and edge set E. Create a bipartite graph with two sides both corresponding to V, that is, for each node $v \in V$ we add two modes v_{in} and v_{out} to the bipartite graph. For each undirected edge $e = (v, w) \in G$ we add the edge (v_{out}, v_{in}) to the bipartite graph G'. We claim that cycle covers in G are in one-to-one correspondence with perfect matching's in G'. This is true as a set of edges forms a cycle cover if and only if it contains exactly one edge entering and exactly one edge entering and exactly one edge leaving each vertex v. Hence we can find a cycle cover in $O(mn)$ time by finding perfect matching in G'.
b. Clearly the Cycle Cover Problem is in NP as given the set of edges that form a cycle cover, it is easy to check if they form a cycle cover of G. We just need to check if a given solution contains all vertices of G. If the cycles of the cover have no vertices in common, and if each cycle has at most three edges.
c. We will prove that Cycle Cover with at most 3 edges in each cycle is NP-complete by a reduction from a *3-Dimmensional Matching*. Consider an instance of the *3-Dimmensional Matching* Problem, given by disjoint sets, X, Y, and Z, each of size n; and a set $T \subseteq X \times Y \times Z$ of ordered triples. We create an instance of the cycle cover problem as follows. Let $S = X \cup Y \cup Z$ denote the set of nodes. First add 3 nodes with a triangle connecting them for each triple $t \in T$. There will be the only triangles used in our construction. Let the 3 nodes in the triangle correspond to the 3 elements of the set t, but node that if a node $s \in S$ is covered by multiple triples, than we add separate nodes corresponding to v for each triple. For a node $s \in S$ let A, be the set of these nodes corresponding to node v. We will also add a set of B_v additional nodes corresponding to node v with $|B_v| = |A_v| - 1$, and add edges between all nodes in the sets A_v and B_v in both directions. This finishes the construction of graph G.

   Now we claim that there is a perfect matching in the *3-Dimmensional Matching* Problem if and only if G has a Cycle Cover with at most 3 edges in each cycle. First assume we have perfect 3D matching. The matching corresponds to a cycle cover as follows. For each triple *t* in the 3D matching, select the corresponding triangle. This covers exactly one node in each set A_s for each $s \in S$. Now cover the remaining nodes by cycles of length 2 going between the sets A_s and B_s.

Finally, we need to show that if *G* has a Cycle Cover with at most 3 edges in each cycle, than there is a perfect 3D matching in the *3-Dimmensional Matching* Problem. So see this note that the elements of the sets B_s can only be covered by the 2 cycles, as no 3 cycle passes through these nodes. Any set of 2 cycles covering the node $\cup_s B_s$ leaves one node uncovered from each set A_s. The union of set $\cup_s A_s$ consists of node-disjoint triangles corresponding to the triples in *T,* so these remaining nodes can only be covered if they correspond to triples in *T.*