CSE 40113: Algorithms
Homework 8

You may complete this homework in groups of 3 or less students. Note that the integrity policy applies: your group should write up your own work, and cite any sources used (including other students). If you have any questions, please re-read both the homework guidelines and the academic integrity policy carefully, and then come discuss any questions or concerns with me.

Required Problems

1. A website is trying to analyze the behavior of its customers from recent sales. They store prior purchase information in a 2-dimensional array X, where the columns correspond to products and the rows to customers; the entry $P[i][j]$ then specifies the number of items of product i that customer j has bought in the past year.

   For example:

	Alice	Bob	Carol	David
stickers	0	6	0	3
tshirts	2	3	0	0
hats	5	0	1	3
mugs	0	3	2	7
cards	3	0	1	0

   Now the store wishes to identify a set of customers that respectively purchase different things, to identify a good subset of customers to poll for feedback on as many distinct products as possible. In other words, we want to find a large set of the customers C such that no two customers in C have purchased the same product.

   This leads us to the *customer poll* problem: given the matrix P and a target value k, is there a subset of at least k customers such that no two have purchased the same product? Show that this problem is NP-Complete.

   Solution: First, show the problem is in NP: Given a set of k customers as a certificate, we can loop through each product and check how many of the customers in the set have purchased that product in $O(k)$ time; as long as there is at most one, the set is a solution. If there are p products, this takes a total of $O(kp)$ time.

   Reduction: We reduce independent set to this problem. The input to independent set is a graph $G = (V, E)$ and a number k. We convert this to customer poll by created a customer for each vertex in G, and then for each edge, creating a single product that is purchased by the customers corresponding to its endpoints. More formally, the matrix P is then:

   $$P[i][j] = \begin{cases} 1 & \text{if customer } j \text{ corresponds to a vertex incident to edge } i, \\ 0 & \text{otherwise.} \end{cases}$$

   Finally, k is unchanged, so k in the customer poll is set to be the same number k as we got as input to independent set.

   The reduction takes time $O(VE)$ total time.

   Proof (if and only if):

- If you have an independent set in G, that immediately gives a set of corresponding customers. Since the set is independent, we know they have no edges between any two vertices included; therefore, there are no products where both customers purchase, since these were only created for edges.
- For the backwards direction: suppose you have a good subset of customers. This means they bought no common products. If we consider the corresponding set of vertices in the graph, this means no edges can be present between them, since that would have given a common product.

■

2. Let's take a look at a more complex resource scheduling problem. Consider a group of n asynchronous processes. However, these processes might need access to some subset of the m shared resources. At any given timestep, each process has a set of resources it would like to use. Each resource might be requested by many of the processes, but it can only actually be used by one at a time. If a process can gain access to all of its requested resources, it is *active*; otherwise, it is *blocked*.

   Your job is to allocate the resources to processes as effectively as possible; here, this means we would like to allocate so that as many processes as possible can be active. This leads to a decision version of the question: given a set of n processes, m resources, and a list of requested resources for each of the processes (so a set of n lists, each of length $\leq m$), and a value k, is it possible to allocate resources to processes so that at least k are active?

   (a) Show that this problem is NP-Hard.

   Solution: We can again reduce Independent set: for this one, create a process for each vertex in the graph, and a resource for each edge, where a process requests each resource that is associated with its edges. In other words, process i requests resource j if vertex i is incident to edge j in the original graph. Finally, k for the process selection problem is the same as k for the independent set problem input.

   The reduction takes $O(V+E)$ time, since we create V process and E resources, then then add each of the resources to two lists.

   Proof:

   - Suppose you have a solution to independent set. Those k vertices directly correspond to k processes, which I claim cannot have any resource requests in common. To see this, note that any such resource would correspond to a shared edge in the graph, so since the set of vertices is independent, no such resource can exist.
   - For the other direction: consider a set of k process that share no resources. We can consider the associated vertices to these, and note that the set of vertices is independent, since any edge between them would mean a shared resource would have been created, which is impossible.

   ■

(b) Suppose now that there are only two types of resources, and each process requires exactly one resource of each type. (Example: say every resource is either a person or a piece of equipment, and each process requires one specific person and one specific piece of equipment from the list.) Show that this version of the problem can be solved in polynomial time (by giving an algorithm).

Solution: This solution is also a reduction, but this time from process section to either bipartite matching or flows. Create a vertex for every resource; since there are two types, this naturally corresponds to two disjoint subsets of vertices. Then, create an edge for every process, and make it incident to the two resources needed (one for each type). Since edges only go between distinct types, we know this graph is bipartite.

Claim: There is a matching of size k in the graph if and only if there is a set of k process that we can activate.

Proof: Suppose we have a set of k processes that can all be active. Since they cannot share resources, this corresponds to a set of edges in the graph where each vertex is touched by at most one edge, which is by definition a matching of size k. For the other way, assume we have a matching of size k. These correspond to a set of k processes, and since in a matching we cannot have a vertex incident to two edges, this means the resources will not be requested by any two processes in the set.

Runtime: If there are p processes and r resources, we create a graph with r vertices and p edges. I accepted either $O(VE) = O(rp)$ or $O(\sqrt{V}E) = O(\sqrt{r}p$ for full credit, since the book mentions both running times.

Note: you also got full credit for reducing this directly to a flow network.

■

3. In my excitement over the planned Mandalorian movie, I have been hunting for algorithms problems while re-watching far too much Star Wars. To my delight, I have found several!

Consider the faced by the Rebel Alliance as they fly from the Death Star back to the secret base on Degobah. We can view the galaxy as an undirected graph $G = (V, E)$, where each node is a start system and each edge $\{u, v\}$ indicates you can travel between u and v. The Death Star is represented by a node s, the hidden Rebel base by a node t. Certain edges have longer distances than others; thus we will give each edge an integer length $l_e \geq 0$. Also, certain edges represent routes that are more heavily patrolled by evil Imperial spacecraft; so each edge e also gets an integer risk $r_e \geq 0$, indicating the expended amount of damage incurred from the special-effects-intensive space battles if you use this edge.

There is a tradeoff here: it would be safest to travel through the outer rim of the galaxy, from one quiet far away star system to another, but then the ship would likely run out of fuel long before getting to its destination. (After all, they are on the run, so stopping to refuel should be avoided!) Alternatively, it would be fasted to dive through the cosmopolitan core of the galaxy, but then there would be far too many Imperial spacecraft to deal with. In general, for any path from s to t, we get both a length (the sum of all the lengths of its edges), and a total risk (the sum of all the risks of its edges).

So Luke, Leia, and company are looking at a complex shortest path type problem in this graph: they want to get from s to t along a path whose total length and risk are *both* reasonably small. In concrete terms, we will phrase this as the *Galactic Shortest-Path Problem*

as follows: Given a setup as above and integer bounds L and R, is there a path from s to t whose total length is at most L and whose total risk is at most R?

Show that Galactic Shortest Paths is NP-Complete.

Solution: First, we show it is in NP: A certificate for this problem is simply a path $P = (v_0 = s, v_1, \ldots, v_k = t)$. We then loop over i from 0 to k and check:

- that the path is valid (adjacent nodes are connected by edges),
- compute the total length $\sum_{i=0}^{k-1} l_{\{v_i, v_{i+1}\}}$,
- compute the total risk $\sum_{i=0}^{k-1} r_{\{v_i, v_{i+1}\}}$,
- and check whether these sums are $\leq L$ and $\leq R$.

Since each path has $O(V)$ edges and does $O(1)$ work per pair $i, i+1$, this in total takes $O(V)$ time.

Reduction: We reduce from the NP-hard problem PARTITION, which given a set of positive integers $X = \{x_1, \ldots x_n\}$, asks if there is a subset $S \subseteq X$ such that

$$\sum_{s_i \in S} x_i = \sum_{x_i \notin S} x_i = \frac{1}{2} \sum_{i=1}^{n} x_i.$$

First, set $T = \sum_{i=1}^{n} x_i$, and let $L = R = T/2$. Now, construct a graph $G = (V, E)$ as follows:

- Create nodes $v_0, v_1, \ldots, v_n$, where $v_0 = s$, and $v_n = t$.
- For each $i = 1, \ldots, n$, add *two edges* between v_{i-1} and v_i:
  - a *top edge* with length $l = x_i$ and risk $r = 0$,
  - a *bottom edge* with length $l = 0$ and risk $r = x_i$.

This construction takes $O(n)$ time, since we need to calculate T and then we create $n+1$ vertices and $2n$ edges.

The proof of reduction is the following:

There exists a path from s to t with total length $\leq L$ and total risk $\leq R$ if and only if there exists a subset $S \subseteq X$ with $\sum_{x_i \in S} x_i = T/2$.

Proof: *($\Rightarrow$)* Suppose such a path exists. Let S be the set of indices i where the top edge was chosen. (We know both cannot be chosen, since such a walk would visit a vertex twice, and in any case could be shortened by removing the cycle.) Then:

$$\sum_{i \in S} x_i \leq \frac{T}{2}, \quad \sum_{i \notin S} x_i \geq \frac{T}{2}.$$

Since $\sum_{i \in S} x_i + \sum_{i \notin S} x_i = T$, equality must hold: if one of the values is $< T/2$, the other must necessarily be larger, in which case it would not solve the galactic shortest paths.

(⇐) Conversely, suppose such a subset S exists. Construct a path taking the top edge if $i \in S'$, and the bottom edge otherwise. Then:

$$\text{total length } = \sum_{i \in S'} a_i = \frac{T}{2} \leq L,$$

$$\text{total risk } = -\sum_{i \notin S'} a_i = -\frac{T}{2} \leq R.$$

And we know that the path is a valid solution to galactic shortest paths.

■