CS 170 Homework 10

Due **4/12/2023, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

Solution: I worked on this homework with the following collaborators:

- Lakshya Nagal, SID: 3037935253

2 Flow vs LP

You play a middleman in a market of m suppliers and n purchasers. The i-th supplier can supply up to $s[i]$ products, and the j-th purchaser would like to buy up to $b[j]$ products.

However, due to legislation, supplier i can only sell to a purchaser j if they are situated at most 1000 miles apart. Assume that you're given a list L of all the pairs (i, j) such that supplier i is within 1000 miles of purchaser j. Given m, n, $s[1..m]$, $b[1..n]$, and L as input, your job is to compute the maximum number of products that can be sold. The runtime of your algorithm must be polynomial in m and n.

For parts (a) and (b), assume the product is divisible—that is, it's OK to sell a fraction of a product.

(a) Show how to solve this problem, using a network flow algorithm as a subroutine. Describe the graph and explain why the output from the network flow algorithm gives a valid solution to this problem.

Solution:

We begin by setting a up a graph with $2 + m + n$ nodes:

- One source node.
- One sink node.
- m nodes, one for each supplier.
- n nodes, one for each purchaser.

For the edges:

- $\forall (i, j) \in L$ draw an edge from the i^{th} m node to the j^{th} n node with capacity $s[i]$.
- Draw an edge with capacity $s[i]$ from the source to the i^{th} supplier.
- Draw an edge wih capacity $b[j]$ from the j^{th} purchaser to the sink node.

This gives a valid solution because for every supplier it well send flow to the corresponding purchaser equal to the max amount of products that the purchaser can buy. Additionally, the max-flow algorithm will saturate the all possible edges, and guarantee the max-flow is outputted, which corresponds to the max amount of products that can be sold.

(b) Formulate this as a linear program. Explain why this correctly solves the problem, and the LP can be solved in polynomial time.
Solution: Define $p(i,j)$ be the number of products bought by purchaser i from the j^{th} supplier.

$$\textbf{maximize: } \sum_{i=1}^{m}\sum_{j=1}^{n} p(i,j)$$

$$\textbf{Subject to: } \sum_{i=1}^{m} p(i,j) \leq b[j], \text{ where } j \in [1,\ldots,n]$$

$$\sum_{j=1}^{n} p(i,j) \leq s[i], \text{ where } i \in [1,\ldots,m]$$

$$\forall (i,j) \in L, p(i,j) \geq 0$$

$$\forall (i,j) \notin L, p(i,j) = 0$$

This essentially model the graph described in part (a), where we want to maximize the amount of products sold to purchaser, under the constraints that we don't exceed the amount of items the purchasers can buy. There are mn variables and at most mn constraints, and the runtime is polynomial in terms of n and m.

(c) Now let's assume you *cannot* sell a fraction of a product. In other words, the number of products sold by each supplier to each purchaser must be an integer. Which formulation would be better, network flow or linear programming? Explain your answer.
Solution: The linear program above is vulnerable to finding non integer solutions, so the best approach would be to run a network flow algorithm like Ford-Fulkerson. Since our capacities are all integers, a network flow algorithm will find an integer solution.

3 Reduction to 3-Coloring

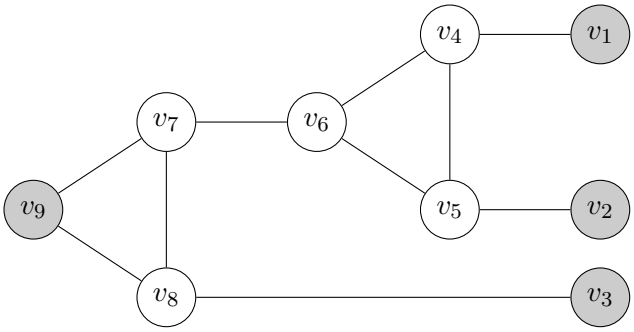
Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from {red, green, blue} such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 3-SAT to the 3-coloring problem. Since we know that 3-SAT is NP-Hard (there is a reduction to 3-SAT from every NP problem), this will show that 3-coloring is NP-Hard (there is a reduction to 3-coloring from every NP problem).
In our reduction, the graph will start with three special vertices, labelled v_{TRUE}, v_{FALSE}, and v_{BASE}, as well as the edges $(v_{\mathsf{TRUE}}, v_{\mathsf{FALSE}})$, $(v_{\mathsf{TRUE}}, v_{\mathsf{BASE}})$, and $(v_{\mathsf{FALSE}}, v_{\mathsf{BASE}})$.

(a) For each variable x_i in a 3-SAT formula, we will create a pair of vertices labeled x_i and $\neg x_i$. How should we add edges to the graph such that in any valid 3-coloring, one of $x_i, \neg x_i$ is assigned the same color as v_{TRUE} and the other is assigned the same color as v_{FALSE}?

Hint: any vertex adjacent to v_{BASE} must have the same color as either v_{TRUE} or v_{FALSE}. Why is this?

Solution: We create an edge from x_i to $\neg x_i$, in this way the two vertices are adjacent (must be colored differently), and we assign one with v_{TRUE} and one with v_{FALSE}. We can then proceed to connect all other vertices to these vertices and assign them to v_{BASE}.

(b) Consider the following graph, which we will call a "gadget":



Consider any valid 3-coloring of this graph that does *not* assign the color red to any of the gray vertices (v_1, v_2, v_3, v_9). Show that if v_9 is assigned the color blue, then at least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Hint: it's easier to prove the contrapositive!

Solution:

Contrapositive: If $\neg\{v_1, v_2, v_3\}$ is assigned blue $\Rightarrow$ v_9 is not assigned blue.
Proof by Contrapositive: If none of v_1, v_2, v_3 are assigned color blue and red is not assigned to any of the gray vertices, then v_1, v_2, v_3 must be green. Consequently v_4, v_5 must be assigned to combination of either blue or red, forcing v_6 to be green. v_8 and v_7 must also be assigned a combination of blue or red, forcing v_9 to be green. Therefore if none of $\{v_1, v_2, v_3\}$ are assigned blue, then v_9 is not assigned blue. Which proves the contrapositive, so our original claim must be true.

(c) We have now observed the following about the graph we are creating in the reduction:

(i) For any vertex, if we have the edges (u, v_{FALSE}) and (u, v_{BASE}) in the graph, then in any valid 3-coloring u will be assigned the same color as v_{TRUE}.

(ii) Through brute force one can also show that in a gadget, if all the following hold:

(1) All gray vertices are assigned the color green or blue.

(2) v_9 is assigned the color blue.

(3) At least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Then there is a valid coloring for the white vertices in the gadget.

Using these observations and your answers to the previous parts, **give a reduction from 3-SAT to 3-coloring. Prove that your reduction is correct (you do not need to prove any of the observations above).**

Hint: create a new gadget per clause!

Solution:

Reduction Algorithm 3-SAT to 3-COLORING:

1. $\forall x_i$ in a 3-SAT formula, draw an edge from x_i to $\neg x_i$ if it exist. Assign x_i and $\neg x_i$ to one of v_{TRUE} or v_{FALSE} such that x_i and $\neg x_i$ are different.
2. Connect x_i and $\neg x_i$ to other variables assigned to v_{BASE}

Proof $\rightarrow$: If $\exists$ a 3-SAT solution $\Rightarrow$ $\exists$ a 3-COLORING of the vertices.
If there exist a 3-SAT solution, then there exists an assignment such that all clauses are satisfied. This assignment is analogous to a coloring scheme. Since each variable and its negation are connected in the graph, they must be assigned different colors in accordance to their 3-SAT assignment. Additionally, since each variable is connected to either v_{TRUE} or v_{FALSE}, the colors of the vertices must satisfy the 3-COLORING constraints.

Proof $\leftarrow$: If $\exists$ a 3-COLORING solution $\Rightarrow$ $\exists$ a 3-SAT solution.
If there exist a valid 3-COLORING of the graph, we can use the colorings to derive a satisfying assignment to the 3-SAT problem. Since each each vertex that is connected has has a different color assignment, we can assign $True$ to the variables with same color as v_{TRUE}, $False$ to the ones with same color as v_{FALSE}.