Homework 1
Due: End of Day Sept 10th (Wednesday), 2025

1. Consider the network as shown in Figure 13. Use linear program to formulate the minimum cost flow problem where the goal is send 1 unit of goods from source s to sink t. Solving your linear program using Matlab or Python or whatever solver that you are comfortable with and interpret the result.

   Hint: Your solution should include the linear program (i.e., objective function and constraints) and also the source code of your routine calling the solver.

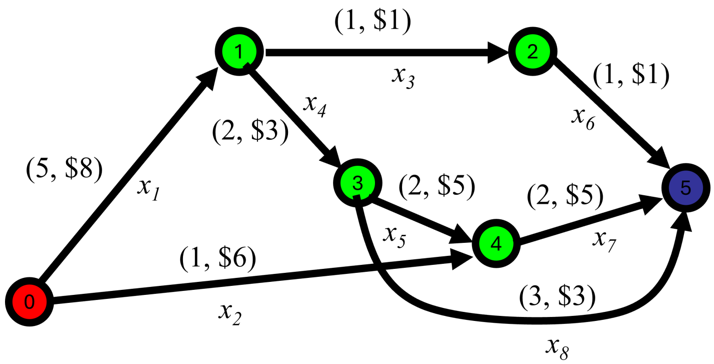


Figure 13: An instance of the minimum cost flow problem.

2. UNM CS Lab orders computers from two vendors Apple and Dell. Each semester, at least 10 computers must be ordered. A computer from Dell costs 2300 USD each, and a computer from Apple costs 600 USD each. (Yep, Apple is cheaper because it is using its own silicon. Dell is more expensive becasue it must equip with a discrete graphics card.) Costs must be kept to less than 10,000 USD. Moreover, UNM requires that the number of computers from each Vendor can't exceed twice the number from another. How many computers UNM CS must order from each vendor to minimize the total cost subject to the above constraints? If we are going to view the problem as a minimum cost flow problem, what does the network looks like? Show that how this problem can be modeled using linear programming. Solve your linear program. Did you notice any issues? Can you explain what causes the issue?

   (Hint: in this problem, you have two sources for the minimum cost flow problem.)

3. You walk into a room, and see a row of n cards. Each one has a number x_i written on it, where i ranges from 1 to n. However, initially all the cards are face down. Your goal is to find a local minimum: that is, a card i whose number is less than or equal to those of its neighbors, $x_{i-1} \geq x_i \leq x_{i+1}$.

   The first and last cards can also be local minima, and they only have one neighbor to compare to. Clearly there can be many local minima, but you are only responsible for finding one of them. Obviously you can solve this problem by turning over all n cards, and scanning through them. However, show that you can find such a minimum by turning over only $o(n)$ cards.

   Answer the following questions:

(a) What does small o of n, i.e., $o(n)$ mean? Can you give a two increasing functions of n that are small o of n?

(b) Develop a strategy that you can find such a minimum by turning over only $o(n)$ cards.

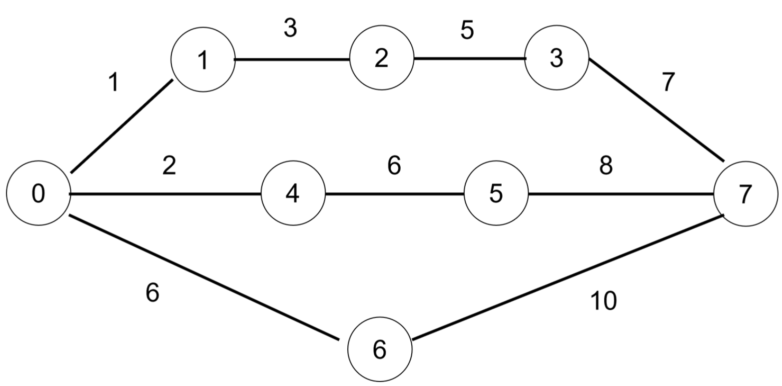
Hint: you may want to consider the search strategy of **binary search**.

4. Show that how you can solve the shortest path problem using linear programming. Specifically, you may assume that you are given a weighted directed graph $G(V, E)$, and two vertices $s, t \in V$. The goal is to find the shortest $s - t$ path.

5. In the lectures, we discussed three shortest path algorithms Bellman-Ford, Floyd-Warshall, and Dijkstra's. All these algorithms are designed based on inductions. Answer the following questions regarding these three shortest paths algorithms.

   - Describe the main induction hypothesis for these three shortest paths algorithms.
   - Simulate the execution of these three algorithms for the graph as shown in Figure 5. For Bellman-Ford and Dijikstra's algorithms, please use vertex 0 as the source vertex.

     Hint: For Bellman-Ford and Floyd-Warshall algorithms, it may be easier to simply implement them using one of your favorite programming languages with the adjacency matrix representation because the algorithm are simply loops.
   - What are the shortest paths from vertex 0 to vertex 7 found by the algorithms? Note that there are obviously three different shortest paths from 0 to 7 and one of them will be discovered first. Explain why.



6. Suppose you are given an undirected simple graph $G(V, E)$. Design an efficient algorithm to determine if the given graph is a tree or not. Note that you should also discuss how you want the input graph to be represented, e.g., an adjacency list or an adjacency matrix.

7. Several families are planning a shared car trip on scenic drives in beautiful Navajo Nation. To minimize the possibilities of any quarrels, they want to assign individuals to cars so that no two members of a family are in the same car. Can you formulate the problem as a network flow problem?