CSCI 570 Homework 4

Due Date: Oct. 29, 2023 at 11:59 P.M.

1. (**Dynamic Programming**) Given n balloons, indexed from 0 to $n-1$. Each balloon is painted with a number on it represented by array $nums$. You are asked to burst all the balloons. If you burst the balloon i you will get $nums[left] \cdot nums[i] \cdot nums[right]$ coins, where left and right are adjacent indices of i. After the bursting the balloon, the left and right then becomes adjacent. Assume, $nums[-1] = nums[n] = 1$ and they are not real therefore you can not burst them. Design a dynamic programming algorithm to find the maximum coins you can collect by bursting the balloons wisely. Analyze the running time of your algorithm.

2. (**Dynamic Programming**) Suppose you are in Casino with your friend, and you are interested in playing a game against your friend by alternating turns. The game contains a row of n coins of values v_i, where n is even. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money you can definitely win if you move first. Analyze the running time of your algorithm.

3. (**Dynamic Programming**) Jack has gotten himself involved in a very dangerous game called the octopus game where he needs to pass a bridge which has some unreliable sections. The bridge consists of $3n$ tiles as shown below. Some tiles are strong and can withstand Jack's weight, but some tiles are weak and will break if Jack lands on them. Jack has no clue which tiles are strong or weak but we have been given that information in an array called **BadTile(3,n)** where **BadTile(j, i) = 1 if the tile is weak and 0 if the tile is strong**. At any step Jack can move either to the tile right in front of him (i.e. from tile (j, i) to $(j, i+1)$), or diagonally to the left or right (if they exist). (No sideways or backward moves are allowed and one cannot go from tile $(1, i)$ to $(3, i+1)$ or from $(3, i)$ to $(1, i + 1)$). Using dynamic programming find out how many ways (if any) there are for Jack to pass this deadly bridge. Analyze the running time of your algorithm.

   Figure below shows bad tiles in gray and one of the possible ways for Jack to safely cross the bridge alive (See Fig. 1).

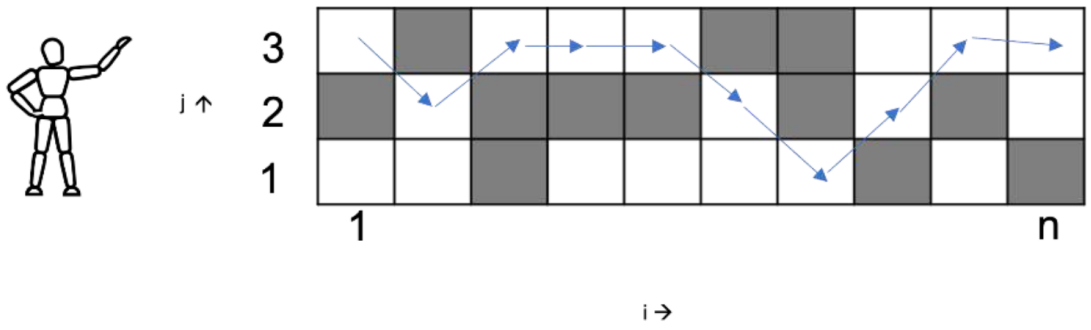

Figure 1

4. Given a flow network with the source s and the sink t, and positive integer edge capacities c. Prove or disprove the following statement: if deleting edge e reduces the original maximum flow more than deleting any other edge does, then edge e must be part of a minimum $s-t$ cut in the original graph.

5. Given a flow network with the source s and the sink t, and positive integer edge capacities c. Let $s - t$ be a minimum cut. Prove or disprove the following statement: If we increase the capacity of every edge by 1, then $s - t$ still be a minimum cut.

6. (**Network Flow**)

(a) For the given graph G_1 (see Fig. 2), find the value of the max flow. Edge capacities are mentioned on the edges. (You don't have to show each and every step of your algorithm).

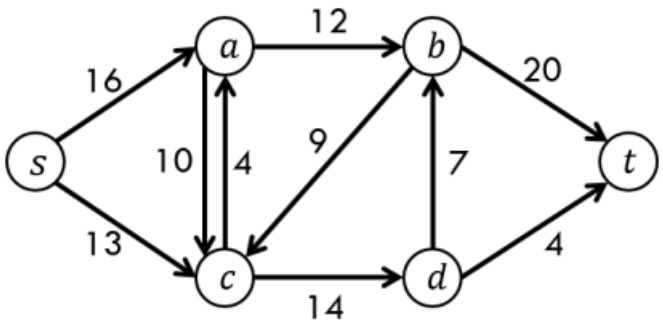

Figure 2: G1

(b) For the given graph, find the value of the min-cut. (You don't have to show each and every step of your algorithm).

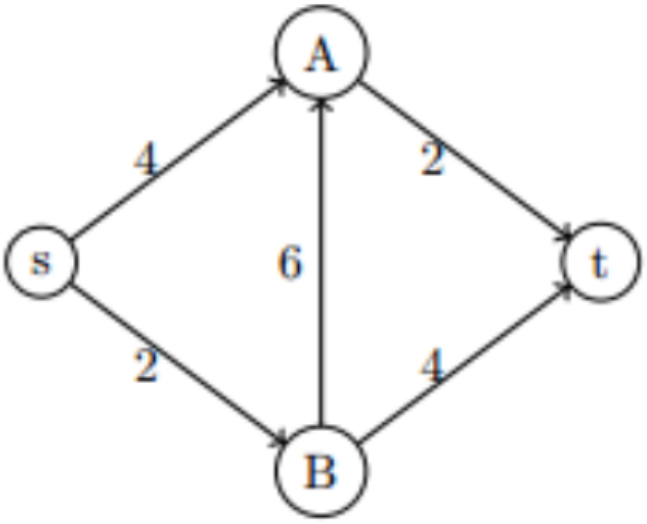

Figure 3

7. (**Network Flow**) Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. We'll suppose there are n clients, $c_1, c_2, \ldots, c_n$, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations, $b_1, b_2, \ldots, b_k$; the position of each of these is specified by (x, y) coordinates as well. For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a range parameter R which means that a client can only be connected to a base station that is within distance R. There is also a load parameter$L$ which means that no more than$L$ clients can be connected to any single base station. Given the positions of a set of clients and a set of base stations, as well as the range and load parameters, decide whether every client can be connected simultaneously to a base station. Prove the correctness of the algorithm.

8. (**Network Flow**) You are given a flow network with unit-capacity edges: it consists of a directed graph $G = (V, E)$ with source s and sink t, and $u_e = 1$ for every edge e. You are also given a positive integer parameter k. The goal is delete k edges so as to reduce the maximum $s - t$ flow in G by as much as possible. Give a polynomial-time algorithm to solve this problem. In other words, you should find a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum $s - t$ flow in the graph $G' = (V, E - F)$ is as small as possible. Give a polynomial-time algorithm to solve this problem.

9. (**Network Flow**) Counter Espionage Academy instructors have designed the following problem to see how well trainees can detect SPY's in an $n \times n$ grid of letters S, P, and Y. Trainees are instructed to detect as many disjoint copies of the word SPY as possible in the given grid. To form the word SPY in the grid they can start at any S, move to a neighboring P, then move to a neighboring Y. (They can move north, east, south or west to get to a neighbor.) The following figure shows one such problem on the left, along with two possible optimal solutions with three SPY's each on the right (See Fig. 4). Give an efficient network flow-based algorithm to find the largest number of SPY's.
Note: We are only looking for the largest **number** of SPY's, not the actual location of the words. No proof is necessary.

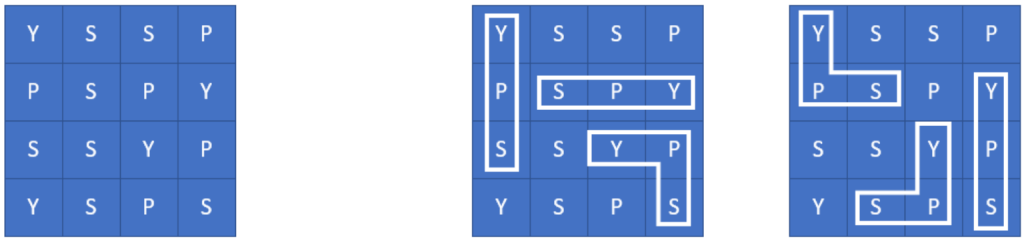

Figure 4

10. (**Network Flow**) USC students return to in person classes after a year long interval. There are k in-person classes happening this semester, $c_1, c_2, ..., c_k$. Also there are n students, $s_1, s_2, ..., s_n$ attending these k classes. A student can be enrolled in more than one in-person class and each in-person class consists of several students.

    (a) Each student s_j wants to sign up for a subset p_j of the k classes. Also, a student needs to sign up for at least m classes to be considered as a full time student. (Given: $p_j \geq m$) Each class c_i has capacity for at most q_i students. We as school administration want to find out if this is possible. Design an algorithm to determine whether or not all students can be enrolled as full time students. Prove the correctness of the algorithm.

    (b) If there exists a feasible solution to part (a) and all students register in exactly m classes, the student body needs a student representative from each class. But a given student cannot be a class representative for more than r (where $r < m$) classes which s/he is enrolled in. Design an algorithm to determine whether or not such a selection exists. Prove the correctness of the algorithm. (Hint: Use part (a) solution as starting point).