Homework 5

Course: Algorithm Design and Analysis **Semester**: Spring 2024

Instructor: Shi Li **Due Date**: **2024/5/19**

Student Name: _________ Student ID: _________

Problems	1	2	3	4	5	Total
Max. Score	20	20	20	20	20	100
Your Score						

Problem 1. Consider the network flow instance $(G = (V, E), c, s, t)$ in Figure 1. The current s-t flow f is also indicated in the figure. (The first number on each edge e is $f(e)$ and the second number is c_e.)

(1a) Draw the residual graph G_f, with residual capacities on edges.

(1b) Using the f in the figure as the initial flow, run Ford-Fulkerson's method to find the maximum flow from s to t. In each iteration, you need to draw the flow f and the residual graph G_f; you can choose the augmentation path arbitrarily.

(1c) Give an s-t cut with value equaling to the flow value.

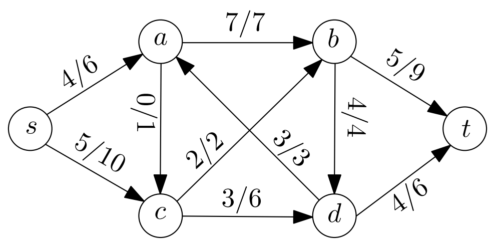

Figure 1: The Network Flow Instance.

Problem 2. Given a graph $G = (V, E)$, an independent set of G is a subset $S \subseteq V$ such that for every two distinct vertices $u, v \in S$, we have $(u, v) \notin E$. The maximum independent set (MIS) problem asks for the independent set S of a given graph $G = (V, E)$ with maximum $|S|$. The problem is NP-hard for general graphs.

Give a *bipartite* graph $G = (L \uplus R, E)$ where all edges are between L and R, design a polynomial time algorithm to find the MIS of G. You need to prove the correctness of your algorithm, that is, the set S returned by the algorithm is indeed the MIS.

Problem 3. Recall problem 3 from homework 2, where you need to design a greedy algorithm to check if n jobs with arrival times, deadlines, and processing times can be processed preemptively on a single machine. This is the problem description:

You are given n jobs which needs to be processed on one machine. Each job j has an arrive time r_j, a deadline d_j and a processing time p_j, where $r_j \in \mathbb{Z}_{\geq 0}$ and $p_j, d_j \in \mathbb{Z}_{>0}$. A job j can only be processed during the time interval $(r_j, d_j]$. The processing of jobs can be interrupted and resumed later, and a job j is completed if it is processed for p_j units of time during the interval $(r_j, d_j]$. Your goal is to check if all the jobs can be completed. Design an $O(n \log n)$ time algorithm to solve the problem.

Formally, the time is slotted into unit time slots $1, 2, 3, \cdots$. During any time slot t, you can choose to process a job j on the time slot, if $r_j < t \leq d_j$. You can process at most one job in a time slot. The jobs can be completed if every job j can be processed during p_j time slots.

For example, consider the instance with 3 jobs given by Table 1. The 3 jobs can all be completed: A feasible solution is given in Figure 2. So, for the instance you should output "yes". However, if p_{c} is 2 instead of 1, then you should output "no".

job indices	a	b	c
r_j	0	3	7
d_j	5	10	9
p_j	4	5	1

Table 1: Instance for Problem 3.

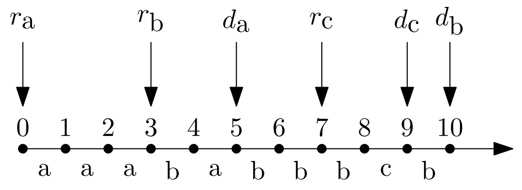

Figure 2: A feasible schedule of the instance.

In this problem, you need to prove the following statement using the maximum-flow-minimum-cut theorem: the input instance is feasible (i.e, there is a schedule that completes all jobs) if and only if for every two integers $0 \leq r < d$ we have

$$\sum_{j \in [n]: r \leq r_j < d_j \leq d} p_j \leq d - r.$$

Problem 4. Given a directed acyclic graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the goal of the problem is to find the minimum number of robots needed to serve all vertices of the graph. Each robot can start and end at any vertex in G and can move only along the edges of the graph. Once a robot visits a vertex v, it can serve v. We allow multiple robots to visit a same vertex v.

For example, consider the example in Figure 3. We need 3 robots to serve all the vertices. Design a polynomial-time algorithm to solve the problem; you need to prove the correctness of the algorithm.

Problem 5. The densest subgraph problem seeks to find the induced subgraph within a given graph that has the highest density, where density is defined as the ratio of the number of edges to the number of vertices in the subgraph.

Formally, given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, the densest subgraph problem aims to find an induced subgraph

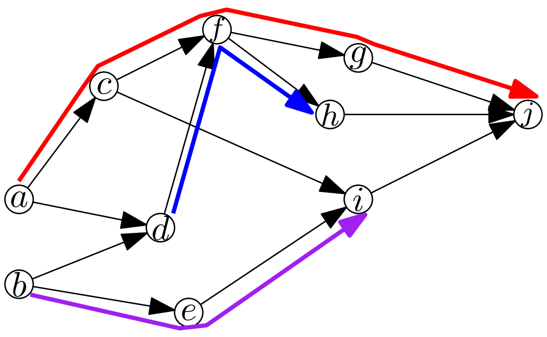

Figure 3: Using 3 robots to serve all the vertices in the graph. The paths for the three robots are $a \to c \to f \to g \to j$, $d \to f \to h$ and $b \to e \to i$.

$H = (V_H \subseteq V, E_H \subseteq E)$ that maximizes the ratio of edges to vertices, i.e.,

$$\text{density}(H) = \frac{|E(H)|}{|V(H)|}.$$

Design a polynomial-time algorithm to solve the problem; you need to prove the correctness of the algorithm.