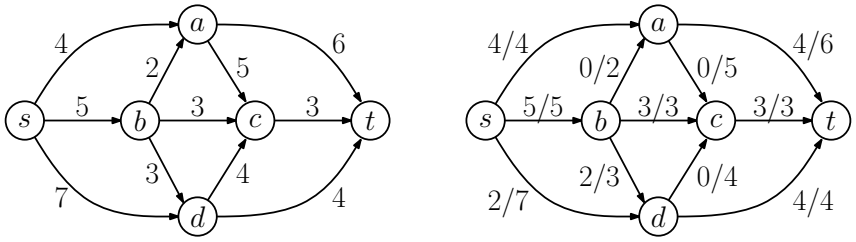
**Homework 4: Network Flow and Applications**

Handed out Thu, Nov 9. Due Mon, Nov 20, 11:59pm (electronic submission through ELMS.) See the syllabus for the late policy.

**Note:** When asked to present an "efficient algorithm," it suffices to provide a reduction to network flow. (Unless the problem asks specifically for this information, you do not need to explain which network-flow algorithm will be used to solve the problem.)

**Problem 1.** In this step, we will trace the partial execution of the Ford-Fulkerson algorithm on a sample network.

(a) Consider the $s$-$t$ network $G$ shown in Fig. 1(a), and consider the initial flow $f$ in Fig. 1(b). Show the residual network $G_f$ for this flow.




(a) Initial network $G$ (b): Initial flow $f$

Figure 1: Problem 1: Ford-Fulkerson.

(b) Find any $s$-$t$ path in $G_f$. How much flow can you push along this path? Show the updated flow (in the same manner as Fig. 1(b)).

(c) Show the residual network that results for your flow from (b).

(d) Is this the final maximum flow in this network? (If not, keep running Ford-Fulkerson until you get the maximum flow, and show the final flow.) What is the value of the maximum flow?

(e) Show the residual network for your maximum flow from (d). (If the flow from (c) was already maximum, then state this.)

(f) Show the cut that results by partitioning the network into two subsets of vertices, the vertices $X$ that are reachable from $s$ and the remaining vertices $Y = V \setminus X$. What is the capacity of this cut? (It should match your flow value, if you did everything correctly.)

**Problem 2.** The computer science department at a major university has a tutoring program. There are $m$ tutors, $\{t_1, \ldots, t_m\}$ and $n$ students who have requested the tutoring service $\{s_1, \ldots, s_n\}$. Each tutor $t_i$ has a set $T_i$ of topics that he/she knows, and each student $s_j$ has a set of topics $S_j$ that he/she wants help with. We say that tutor $t_i$ is *suitable* to work with student $s_j$ if $S_j \subseteq T_i$. (That is, the tutor $t_i$ knows all the topics of interest to student $s_j$.) Finally, each tutor $t_i$ has a range $[a_i, b_i]$, indicating that this tutor would like to work with at least $a_i$ students and at most $b_i$ students.

Given a list of students, a list of tutors, the ranges $[a_i, b_i]$ for the tutors, and a list of suitable tutors for each student, present an efficient algorithm that determines whether it is possible to generate a pairing of tutors to students such that:

- Each student is paired with *exactly* one tutor.
- Each tutor $t_i$ is paired with *at least* $a_i$ and *at most* $b_i$ students.
- Each student is paired only with a *suitable* tutor.

**Problem 3.** An edge of a flow network is called *critical* if decreasing the capacity of this edge results in a decrease in the maximum flow value. Present an efficient algorithm that, given an $s$-$t$ network $G$ finds any critical edge in a network (assuming one exists).

**Problem 4.** In class we showed that the maximum matching in a bipartite graph could be computed by reduction to network flow. In this problem, we will consider a different approach.

Let $G = (V, E)$ be a bipartite graph, meaning that $V$ can be partitioned into two sets $X$ and $Y$ such that each edge has one endpoint in $X$ and the other in $Y$. Let $M \subset E$ be a *(partial) matching*, which means that it consists of a set of edges (but not necessarily maximal) such that each vertex of $V$ is incident to at most one edge of $M$. If a vertex is incident to an edge of $M$ it is said to be *matched*, and otherwise it is *unmatched*. (We may assume that each vertex $u$ is associated with a boolean matched[$u$], which is True if $u$ is matched and False otherwise.)

For example, Fig. 2(b) shows a (partial) matching $M$ is given for the bipartite graph $G$. The unmatched vertices are shaded.

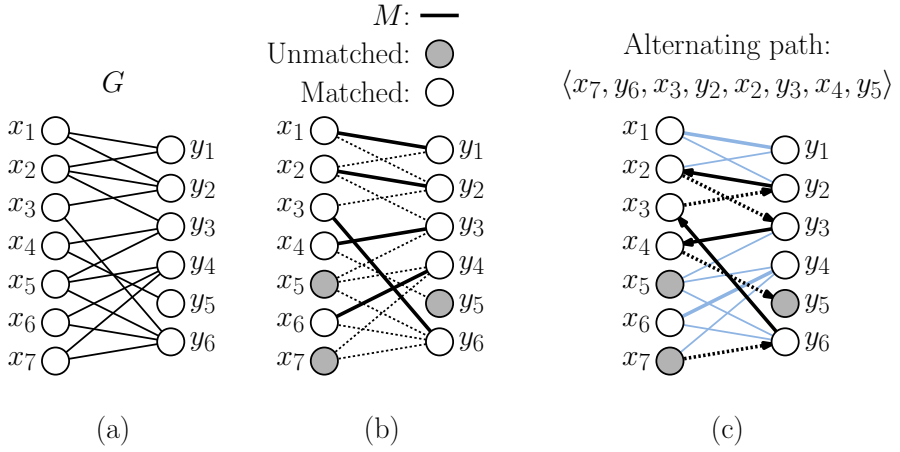



Figure 2: Problem 4: Bipartite matching and alternating paths.

Given $G$ and a matching $M$, define an *alternating path* for $G$ with respect to $M$ to be any path in $G$ that:

- starts at an unmatched vertex of $X$,
- ends at an unmatched vertex of $Y$, and
- its edges alternate between being in $E \setminus M$ and $M$.

(See Fig. 2(c).) Since the first and last vertices are unmatched, the first and last edges are not in $M$. Observe that such a path consists of an odd number of edges, and it has one fewer edge in $M$ than it has edges that are not in $M$. (A single edge $(x, y)$ where both $x$ and $y$ are unmatched is a trivial example of an alternating path.)

(a) Prove that $M$ is a maximum matching if and only if $G$ has no alternating path with respect to $M$. (**Hint:** One side of the implication is straightforward. For the other side, suppose that $M$ is not maximum. Let $M'$ be the true maximum matching. Show that the union of the two edge sets $M$ and $M'$ contains an alternating path.)

(b) Present an $O(n+m)$ time algorithm which given $M$ and $G$ either computes an alternating path, or determines that no such path exists. (Note: You are not told where the path starts.) **Hint:** This can be done by modifying either DFS or BFS to an appropriate graph.

(c) Using this approach, describe an $O(nm)$ time algorithm for computing a maximum matching in a bipartite graph.

**Problem 5.** The river Zod runs through the along the border of the great kingdom of Zod. There are $m$ bridges that cross this river, and great king Zod has asked you to construct a guardhouse at one of the two ends of each bridge to protect the bridge from the barbarians living the neighboring kingdom of Zilch. Each guardhouse costs $\$10,000$ Zod dollars to build, the king has given you $\$10,000 \cdot m$ dollars to construct all the guardhouses. (Fig. 3(a) shows a possible layout of the bridges across the river Zod.)

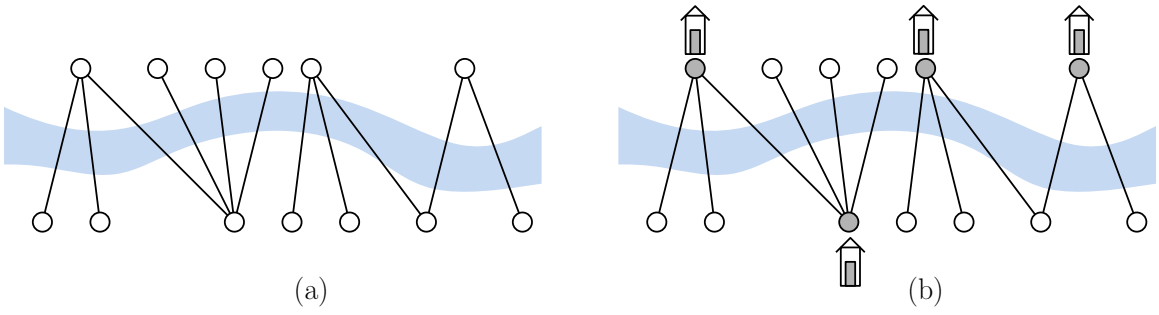



Figure 3: Problem 4: Guarding the bridges of Zod.

You observe that many of the bridges share a common endpoint. Thus, you can save money by building just one guardhouse to protect all the bridges that share this common endpoint. The fewer guardhouses you build, the more of the king's money you get to keep. (Hopefully the king will never find out, or it's off with your head!)

You model the bridges as the edges of a bipartite graph. Two bridges that share the same endpoint are incident to the same vertex (on one side of the river or the other). Two bridges can share the same endpoint, but no bridge can cross over another bridge.

Given this graph, your objective is to identify the smallest number of vertices (from either side) to place guardhouses so that every bridge is guarded. (For example, in Fig. 3(b) we show that four guardhouses suffice.) Let $m$ be the number of bridges (edges) and $n$ be the total

number of endpoints (vertices). Devise an efficient algorithm to solve this problem. (**Hint:** This can be solved by Dynamic Programming, but the non-overlapping nature of the bridges is critical. It suffices to give just the formulation for computing the maximum number of bridges. I don't need pseudocode or the locations of the bridges.)

Hey, I thought this was a homework on Network Flows! . . . See Challenge Problem 2.

Challenge problems count for extra credit points. These additional points are factored in only after the final cutoffs have been set, and can only increase your final grade.

**Challenge Problem 1.** What relationship is there (if any) between the alternating-path algorithm of Problem 3(c) and applying Ford-Fulkerson to the reduction given in class (Lecture 16) for the bipartite matching problem.

**Challenge Problem 2.** Solve Problem 4 (bridges over the river Zod) without making any assumptions about the layout of the bridges. (An example of the input and output is shown in Fig. 4(a) and (b)). In particular, two bridges may overlap each other. Prove that your algorithm is correct. (**Hint:** This problem is closely related to maximum matching, and can be reduced to computing a minimum cut in an appropriate $s$-$t$ network. The selection of guardhouses depends on which side of the cut the endpoint lies.)

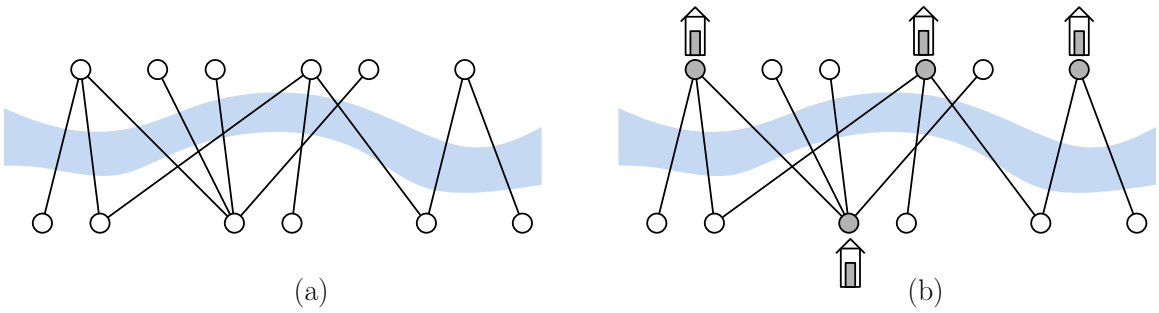



Figure 4: Challenge Problem 2: Guarding the bridges of Zod, general case.