CS 473: Algorithms, Fall 2016
HW 7 (due Tuesday, October 25th at 8pm)

This homework contains three problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this home work, each student can work in a group with up to three members. Only one solution for each group needs to be submitted. Follow the submission instructions carefully.

For problems that use maximum flows as a black box, a full-credit solution requires the following.

- A complete description of the relevant flow network, specifying the set of vertices, the set of edges (being careful about direction), the source and target vertices s and t, and the capacity of every edge. (If the flow network is part of the original input, just say that.)
- A description of the algorithm to construct this flow network from the stated input. This could be as simple as "We can construct the flow network in $O(n^3)$ time by brute force."
- A description of the algorithm to extract the answer to the stated problem from the maximum flow. This could be as simple as "Return TRUE if the maximum flow value is at least 42 and False otherwise."
- A proof that your reduction is correct. This proof will almost always have two components. For example, if your algorithm returns a boolean, you should prove that its TRUE answers are correct and that its FALSE answers are correct. If your algorithm returns a number, you should prove that number is neither too large nor too small.
- The running time of the overall algorithm, expressed as a function of the original input parameters, not just the number of vertices and edges in your flow network.
- You may assume that maximum flows can be computed in $O(VE)$ time. Do *not* regurgitate the maximum flow algorithm itself.

Reductions to other flow-based algorithms described in class or in the notes (for example: edge-disjoint paths, maximum bipartite matching, minimum-cost circulation) or to other standard graph problems (for example: reachability, minimum spanning tree, shortest paths) have similar requirements.

---

1. Maxflow-mincut theorem and Menger's theorem(s) allow one to move between flows and cuts and prove useful and interesting facts that would otherwise be difficult to show directly. Prove the following.

   - Let $G = (V, E)$ be a directed graph and let u, v, w be distinct vertices. Suppose there are k edge disjoint paths from u to v in G, and k edge disjoint paths from v to w in G. Note that the paths from u to v can share edges with the paths from v to w. Prove that there are k edge disjoint paths from u to w in G.
   - **Not to submit:** Suppose G is an Eulerian directed graph. That is, the in-degree of each vertex is the same as the out-degree of each vertex. Prove that if there are k edge-disjoint paths from u to v then there are k edge-disjoint paths from v to u.
   - Suppose G is a connected simple *undirected* graph. A vertex v is a cut-vertex if $G - v$ has at least two distinct connected components. If G has no cut-vertices we say it is a *block* or 2-node-connected. If G has exactly two vertices then it is an edge and is a block. Blocks are more interesting when there are at least 3 vertices. Prove that in a block with at least 3 vertices, every two vertices u, v are in some cycle. In fact you can use this to prove an even a stronger statement that every two edges e_1 and e_2 are in a cycle. Do you see how?

2. The Computer Science Department at UIUC has n professors. They handle department duties by taking part in various committees. There are m committees and the j'th committee requires k_j professors. The head of the department asked each professor to volunteer for a set of committees. Let $S_i \subseteq \{1, 2, \ldots, m\}$ be the set of committees that professor i has volunteered for. A committee assignment consists of sets $S'_1, S'_2, \ldots, S'_n$ where $S'_i \subseteq \{1, 2, \ldots, m\}$ is the set of committees that professor i will participate in. A *valid* committee assignment has to satisfy two constraints: (i) for each professor i, $S'_i \subseteq S_i$, that is each professor is only given committees that he/she has volunteered for, and (ii) each committee j has k_j professors assigned to it, or in other words j occurs in at least k_j of the sets $S'_1, S'_2, \ldots, S'_n$.

   (a) **Not to submit:** Describe a polynomial time algorithm that the head of the department can employ to check if there is a valid committee assignment given m, $k_1, k_2, \ldots, k_m$ the requirements for the committees, and the lists $S_1, S_2, \ldots, S_n$. The algorithm should output a valid assignment if there is one.

   (b) The head of the department notices that often there is no valid committee assignment because professors naturally are inclined to volunteer for as few committees as possible. To overcome this, the definition of a valid assignment is relaxed as follows. Let ℓ be some integer. An assignment $S'_1, S'_2, \ldots, S'_n$ is now said to be valid if (i) $|S'_i - S_i| \leq \ell$ and (ii) each committee j has k_j professors assigned to it. The new condition (i) means that a professor i may be assigned up to ℓ committees not on the list S_i that he/she volunteered for. Describe an algorithm to check if there is a valid committee assignment with the relaxed definition.

3. Let $G = (V, E)$ be a *directed* graph and let $\mathcal{C} = \{C_1, C_2, \ldots, C_h\}$ be a collection of cycles in G. We say that $\mathcal{C}$ is a *cycle partition* of G if each vertex of V is in exactly one of the cycles. In other words the cycles of $\mathcal{C}$ are vertex disjoint and together contain all vertices. Describe an algorithm that given G decides whether G contains a cycle partition. Follow the two steps below.

- Argue that a set of edges $E' \subseteq E$ forms a cycle partition if and only if each vertex v has exactly one incoming edge and one outgoing edge in E'.
- Use bipartite matching to check if there is an $E' \subseteq E$ satisfying the propety in the previous part.

The remaining problems are for self study. Do *NOT* submit for grading.

- See Problem 3 in HW 6 and all problems in HW 7 from Jeff's home work last spring. `https://courses.engr.illinois.edu/cs473/sp2016/hw/hw7.pdf`
- Klenberg-Tardos Chapter 7 is an excellent source on network flow and has many nice problems starting with basic ones to advanced ones. There are several nice problems on reductions to network flow.
- Jeff's notes on network flow applications also has a good collection or problems.