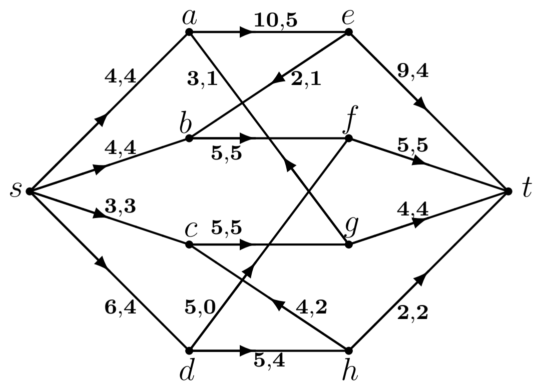
APM 4663/5663—Fall 2024
Assignment #3
First due date: Sunday, November 10, 2024

100% for graduate students: 42 pts. 100% for undergraduate students: 38 pts.

1. (6 pts.) Let n, k be positive integers. Prove that if G is a simple graph on n vertices such that $\deg_G(v) \geq \lceil \frac{n+k-1}{2} \rceil$ for every vertex $v \in V(G)$, then deleting any k vertices of G results in a connected graph.

2. (6 pts.) Let $k \geq 3$ and $\delta \geq 2$. Prove that if G is a simple graph such that $\delta_{\min}(G) \geq \delta$ and every cycle of G has length at least k, then G contains a cycle of length at least $(\delta - 1)(k - 2) + 2$.

3. (12 pts.) A flow F is given in the network below (capacities and flow values are indicated on the edges). Check that F satisfies the conservation constraints (be explicit), then find an F-augmenting s-t path starting from the given flow (indicate the queue and the parents of the vertices in the queue). Augment F along this path. Repeat this process until you find a maximum flow and a minimum s-t cut to certify that your solution is optimal. You may use the picture in Moodle to indicate the searches, the new flow(s), and the minimum cut.



4. (6 pts.) For $n \geq 1$ let G_n be the simple graph with vertex set $V(G_n) = \{1, 2, \ldots, n\}$ in which two different vertices i and j are adjacent whenever j is a multiple of i or i is a multiple of j. For what n is G_n planar?

5. (6 pts.) Prove that every d-regular ($d \geq 1$) bipartite graph has a perfect matching.

6. (6 pts.) Show that every 5-regular, 4-edge-connected graph has a perfect matching. *(4-edge-connected means that if we delete up to 3 edges, the graph remains connected.)*

Students choosing the Computer Science Option should do problems 1–4, and

1. (12 pts.) Implement the Ford–Fulkerson algorithm as we discussed it in class to find a maximum flow and a corresponding minimum s-t-cut in a weighted directed graph. Specifications of the program:

   Input: The input graph is given in a file in the following format: the first line contains just the number of vertices (which are assumed to be labeled with the numbers 1 up to the number of vertices), the second line contains two vertices, the first one is the source s, the second one is the sink t, and then the edges are listed by specifying the tail followed by the head and the corresponding weight of the edge (each line contains info about one edge, you may assume that the graph has no multiple edges). Sample files will be found on Moodle.

   Output: The program should print into a file the value of a maximum flow, and list all edges with the flow value on them, then give the vertices on the left side of a minimum cut (containing the source), and all edges in the cut with their capacities (which should add up to the value of the flow).

   E-mail me the source code of the program as a simple text file and the resulting outputs for the given sample inputs. In addition, e-mail me documentation of the program (this may partially be in the source code), explaining the data structures, main variables, and the algorithm you used.