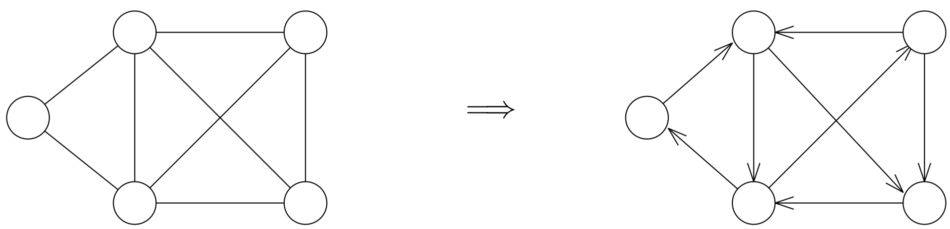
CS 473, Spring 2023
Homework 7 (due Mar 29 Wed 9pm)

Instructions: As in Homework 1.

Problem 7.1: Given an undirected graph $G = (V, E)$ and an integer d, we want determine whether it is possible to direct the edges so that the resulting directed graph has maximum **out-degree** at most d. Describe how to solve this problem by reduction to maximum flow. Prove correctness of your method.

Example: for the undirected graph below (left) and $d = 2$, the answer is yes, and one solution is shown on the right (there are many other solutions).



(Hint: start with a bipartite graph where the vertices on the left side are the edges in G and the vertices on the right side are the vertices in G. Then add a source and a sink, set capacity of each edge appropriately. . .)

Problem 7.2: Given a bipartite graph $G = (V, E)$ with n vertices and m edges, we want to find the largest *independent set* I, i.e., a subset $I \subseteq V$ such that no two vertices in I are adjacent in G.

One way to solve the problem is to construct a flow network (a directed graph) $G' = (V \cup \{s, t\}, E')$, where s is the source and t is the sink. Let V_L and V_R denote the left and right side of V in G. For each $u \in V_L$, we add the directed edge (s, u) to E'. For each $v \in V_R$, we add the directed edge (v, t) to E'. For each $uv \in E$ with $u \in V_L$ and $v \in V_R$, we add the directed edge (u, v) to G'. All edges have capacity 1. (This is the same flow network we have used to reduce maximum bipartite matching to maximum flow.)

(a) (45 pts) Prove that if there is an independent set in G of size k, then there is an (s, t)-cut in G' of capacity $n - k$.

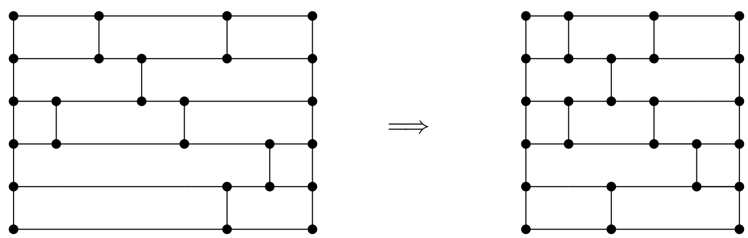
(b) (45 pts) Conversely, prove that if there is an (s, t)-cut in G' of capacity $n - k$, then there is an independent set in G of size k.

(c) (10 pts) Conclude that there is a polynomial-time algorithm to compute a largest independent set in G.

Problem 7.3: We are given a set of n axis-aligned rectangles $R = \{r_1, \ldots, r_n\}$. The rectangles r_i all have height 1 and have integer x- and y-coordinates, and no two rectangles intersect except along the boundaries, and their union is a rectangle $U(R)$. We want to redraw the rectangles as $R' = \{r'_1, \ldots, r'_n\}$, whose union is a new rectangle $U(R')$, so that the rectangles r'_i still have height 1 and have integer x- and y-coordinates, and the adjacency structure is unchanged, i.e., if a side of the rectangle r_i touches a side of rectangle r_j, then the corresponding side of r'_i touches the corresponding side of r'_j.

(a) (25 pts) Describe an efficient algorithm to compute a redrawing R' that minimizes the overall width of $U(R')$.

(Hint: reduce this to a shortest path problem. No need for flows!)



a redrawing with overall width 5

(b) (75 pts) Suppose that each rectangle r_i is given a cost a_i. Describe a polynomial-time algorithm to compute a redrawing S' that minimizes $\sum_{i=1}^{n} a_i \cdot \text{width}(r'_i)$, where $\text{width}(r'_i)$ denotes the width of r'_i.

You may assume that there is a polynomial-time algorithm for the following version of the *minimum-cost flow* problem: given a directed graph $G = (V, E)$ with source s and sink t and a value d, where each edge $e \in E$ is given numbers $\ell(e), c(e), \text{cost}(e)$, compute a flow f of value d, minimizing $\sum_{e \in E} \text{cost}(e) \cdot f(e)$, such that for every $e \in E$, we have $\ell(e) \le f(e) \le c(e)$, and we have conservation of flow at every vertex in $V - \{s, t\}$.

(Hint: create a vertex for each rectangle. . .)