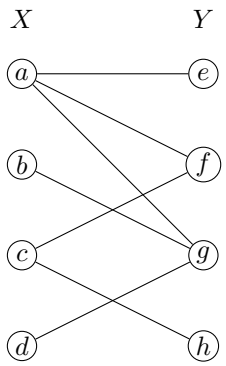
CMPSC 465 Data Structures & Algorithms
Fall 2024 Mingfu Shao and Debarati Das

HW 8

Due October 25th (Friday), 11:59 pm

Formatting: Each problem should begin on a new page. When submitting in Gradescope, try to assign pages to problems from the rubric as much as you can. Make sure you write all your group members' names. For the full policy on assignments, consult the syllabus.

1. (2 × 6 pts.) Find a maximum matching and minimum vertex cover of the following bipartite graph $G = (X \cup Y, E)$ by reducing to the maximum flow problem:



   (a) Draw the corresponding network $G' = (V', E')$ together with capacity $c(e)$ for any $e \in E'$.

   (b) Give a maximum integral flow f^* of G'.

   (c) Give the maximum matching of G constructed from f^*.

   (d) Draw the residual graph G_{f^*}.

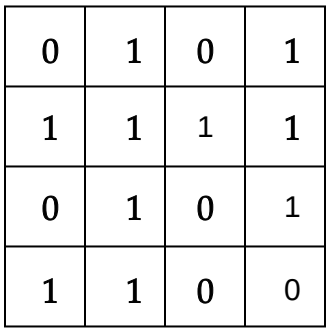
   (e) Give a minimum s-t cut (S^*, T^*) of G' (based on above residual graph).

   (f) Give the minimum vertex cover of G constructed from G_{f^*}.

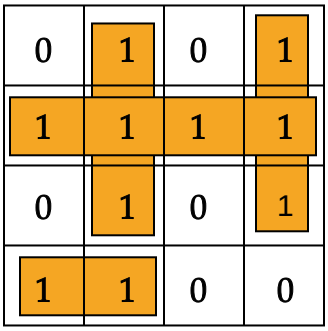
2. (10 + 4 bonus pts.) You are given an $n \times n$ binary matrix M, unlimited number of $k \times 1$ column vectors and $1 \times k$ rows vectors for all $1 \leq k \leq n$. Each length k column/row vector can be used to cover continuous k elements of one column/row of M.

   (a) Devise a polynomial-time algorithm to compute the minimum number of vectors such that each 1 is covered by at least one vector and that each 0 is not covered by any vector. See an example below. Hint: transform this problem into a minimum vertex cover problem on a bipartite graph.

   (b) Prove that your algorithm is correct.



input: matrix M



output: 4 vectors can cover all 1s.

3. (12 pts.) There are m mice and n holes on the ground of a warehouse; each is represented with a distinct 2D (x, y) coordinates. A cat is coming and all mice are running to try to hide in a hole. Each hole can hide at most k mice. All mice run at the same velocity v. A mouse is eventually safe if it reaches a hole (with at most k mice including itself) in s seconds. Design a polynomial-time algorithm (in m and n) to find the maximum number of mice that can be safe. Hint: reduce this problem to a network flow problem.