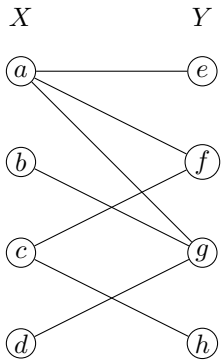
CMPSC 465 Data Structures & Algorithms
Fall 2024 Mingfu Shao and Debarati Das

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.

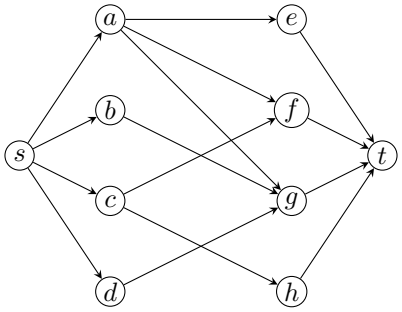
1. (2 × 6 pts.) Find a maximum matching and minimum vertex cover of the following bipartite graph $G = (X \cup Y, E)$ by reducing to the maximum flow problem:



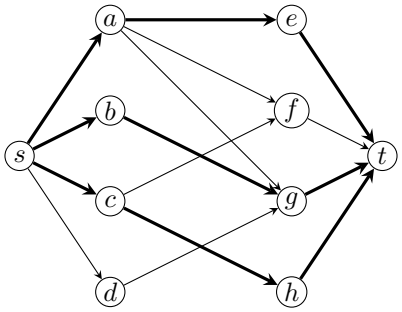
   (a) Draw the corresponding network $G' = (V', E')$ together with capacity $c(e)$ for any $e \in E'$.
   (b) Give a maximum integral flow f^* of G'.
   (c) Give the maximum matching of G constructed from f^*.
   (d) Draw the residual graph G_{f^*}.
   (e) Give a minimum s-t cut (S^*, T^*) of G' (based on above residual graph).
   (f) Give the minimum vertex cover of G constructed from G_{f^*}.

   Solution.

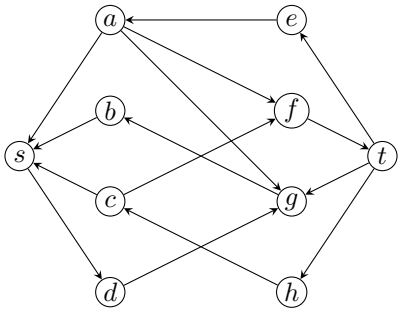
   1. Network $G' = (V', E')$ is given below, with capacity $c(e) = 1$ for any $e \in E'$:



2. A maximum flow f^* is represented below, with thick edges representing edges with $f^*(\cdot) = 1$ and thin edges representing edges with $f^*(\cdot) = 0$:



3. The maximum-matching for G is $\{(a, e), (b, g), (c, h)\}$.
4. The residual graph G_{f^*} is given below, with capacity $c(e) = 1$ for all edges:



5. $S = \{s, b, d, g\}$, $T = V \setminus S = \{a, c, e, f, h, t\}$.
6. $X \cap T = \{a, c\}$, and $Y \cap S = \{g\}$. The minimum vertex cover for G is $(X \cap T) \cup (Y \cap S) = \{a, c, g\}$.

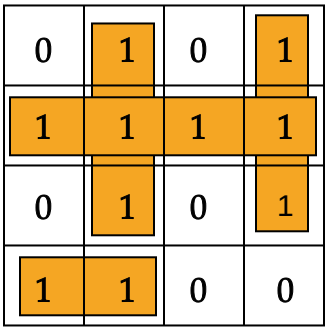
2. (10 + 4 bonus pts.) You are given an $n \times n$ binary matrix M, unlimited number of $k \times 1$ column vectors and $1 \times k$ rows vectors for all $1 \leq k \leq n$. Each length k column/row vector can be used to cover continuous k elements of one column/row of M.

   (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.

0	1	0	1
1	1	1	1
0	1	0	1
1	1	0	0

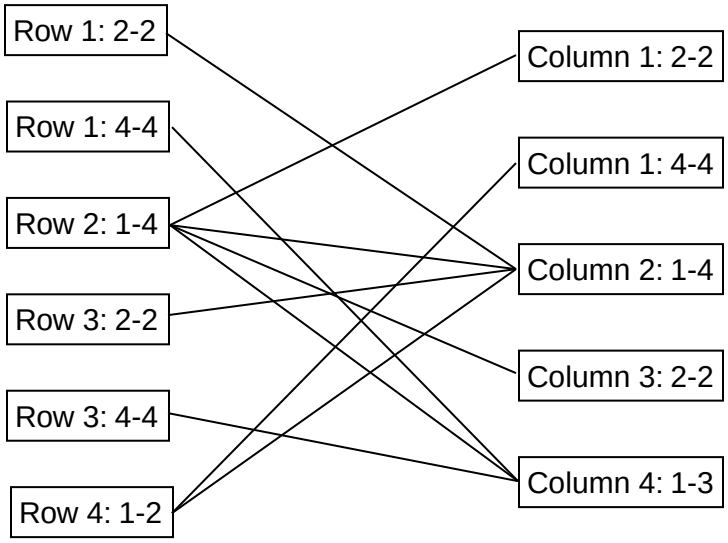
input: matrix M



output: 4 vectors can cover all 1s.

Solution:

(a) First, we find all possible locations of those vectors, which are continuous 1s in M, since 0 is not allowed to be covered. Clearly, we only need to consider those longest continuous 1s, since each 1 can be covered many times. Now we build a bipartite graph $G = (X \cup Y, E)$. Each vertex in X corresponds to one vertical continuous longest 1s, and each vertex in Y corresponds to one horizontal longest 1s. Two vertices are connected if the corresponding column segment and row segment share one element. See the graph for the example below.



The minimum vertex cover of the bipartite graph gives the minimum vectors needed. The running time of above algorithm is dominated by finding a minimum vertex cover of G, which can be done in polynomial-time.

(b) Note that each edge in G corresponds to a single 1 in the given matrix M. Hence, there is a one-to-one correspondence between a vertex-cover of the constructed bipartite graph G and a placement of vectors to cover all 1s in M. Thus, the problem of finding the minimum number of covering vectors is equivalent to determining the minimum vertex cover for graph G, i.e., the minimum vertex-cover of G also gives the minimum vectors required.

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.

   Solution:

   We build a network $G = (V, E)$, where $V = \{M_1, M_2, \cdots, M_m\} \cup \{H_1, H_2, \cdots, H_n\} \cup \{s, t\}$. $\{M_i\}$ correspond to the m mice and $\{H_j\}$ correspond to the n holes. In the network we add edge (M_i, H_j) if mouse-i can reach hole-j in s seconds, i.e., if the distance between mouse-i and hole-j is within $s \cdot v$; the capacities of such edges are always 1. We add edge (s, M_m) with capacity 1, and add edge (H_j, t) with capacity of k to guarantee that each hole can hide up to k mice. We then calculate an (integral) flow f of this network; $|f|$ gives the maximum number of mice that can be safe.

   The running time of above algorithm consists of building the network which is $O(mn)$ and calculating the max-flow which is $O(|v|^2|E|) = O((m+n)^2mn)$. The total running time is $O((m+n)^2mn)$.

   The correctness of above is straightforward. There is a one-to-one correspondence between "hiding" plan and integral flow of the network. Therefore, a plan in which maximized number of mice can be safe correspond to the flow with maximized value.