Algorithm Design and Analysis

Assignment 5

Deadline: Jan 2, 2025

1. (25 points) **[Invertible Matrices]** Given an $n \times n$ matrix A whose entries are either 0 or 1, you are allowed to choose a set of entries with value 1 and modify their value to 0. Design a polynomial-time algorithm to decide if we can make A invertible by the above-mentioned modification. Prove the correctness of your algorithm and analyze its time complexity. (Hint: Prove that this is possible if and only if there exist n entries with value 1 such that no two entries are in the same row and no two entires are in the same column.)

2. (25 points) **[Common System of Distinct Representatives]** Given a ground set $U = \{1, \ldots, n\}$ and a collection of k subsets $\mathcal{A} = \{A_1, \ldots, A_k\}$, a *system of distinct representatives* of $\mathcal{A}$ is a "representative" collection T of distinct elements from the sets in $\mathcal{A}$. Specifically, we have $|T| = k$, and the k *distinct* elements in T can be ordered as $u_1, \ldots, u_k$ such that $u_i \in A_i$ for each $i = 1, \ldots, k$. For example, $\{A_1 = \{2, 8\}, A_2 = \{8\}, A_3 = \{4, 5\}, A_4 = \{2, 4, 8\}\}$ has a system of distinct representatives $\{2, 4, 5, 8\}$ where $2 \in A_1, 4 \in A_4, 5 \in A_3, 8 \in A_2$, while $\{A_1 = \{2, 8\}, A_2 = \{8\}, A_3 = \{4, 8\}, A_4 = \{2, 4, 8\}\}$ does not have a system of distinct representatives.

   (a) (10 points) Design a polynomial time algorithm to decide if $\mathcal{A}$ has a system of distinct representatives.

   (b) (15 points) Given a ground set $U = \{1, \ldots, n\}$ and two collections of k subsets $\mathcal{A} = \{A_1, \ldots, A_k\}$ and $\mathcal{B} = \{B_1, \ldots, B_k\}$, a *common system of distinct representatives* is a collection T of k elements that is a system of distinct representatives of both $\mathcal{A}$ and $\mathcal{B}$. Design a polynomial time algorithm to decide if $\mathcal{A}$ and $\mathcal{B}$ have a common system of distinct representatives.

   For each part, prove the correctness of your algorithm, and analyze its time complexity.

3. (25 points) **[Perfect Matching on Bipartite Graph]** A graph is *regular* if every vertex has the same degree. Let $G = (A, B, E)$ be a regular bipartite graph with $|E| > 0$.

   (a) (10 points) Prove that $|A| = |B|$.

   (b) (15 points) Let $n = |A| = |B|$. Prove that G must contain a matching of size n.

4. (25 points) **[LP-Duality and Total Unimodularity]** In this question, we will prove König-Egerváry Theorem, which states that, in any bipartite graph, the size of the maximum matching equals to the size of the minimum vertex cover. Let $G = (V, E)$ be a bipartite graph.

   (a) (4 points) Explain that the following linear program describes *the fractional version of* the maximum matching problem (i.e., replacing $x_e \geq 0$ to $x_e \in \{0, 1\}$ gives you the exact description of the maximum matching problem).

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{e : e=(u,v)} x_e \leq 1 && (\forall v \in V) \\ & x_e \geq 0 && (\forall e \in E) \end{aligned}$$

   (b) (4 points) Write down the dual of the above linear program, and justify that the dual program describes the fractional version of the minimum vertex cover problem.

   (c) (7 points) Show by induction that the *incident matrix* of a bipartite graph is totally unimodular. (Given an undirected graph $G = (V, E)$, the incident matrix A is a $|V| \times |E|$ zero-one matrix where $a_{ij} = 1$ if and only if the i-th vertex and the j-th edge are incident.)

   (d) (6 points) Use results in (a), (b) and (c) to prove König-Egerváry Theorem.

   (e) (4 points) Give a counterexample to show that the claim in König-Egerváry Theorem fails if the graph is not bipartite.

5. (Bonus 60 points) **[Dinic's Algorithm]** In this question, we are going to work out *Dinic's algorithm* for computing a maximum flow. Similar to Edmonds-Karp algorithm, in each iteration of Dinic's algorithm, we update the flow f by increasing its value and then update the residual network G^f. However, in Dinic's algorithm, we push flow along *multiple* s-t paths in the residual network instead of a *single* s-t path as it is in Edmonds-Karp algorithm.

   In each iteration of the algorithm, we find the *level graph* of the residual network G^f. Given a graph G with a source vertex s, its level graph $\overline{G}$ is defined by removing edges from G such that only edges pointing from level i to level $i+1$ are kept, where vertices at level i are those vertices at distance i from the source s. An example of level graph is shown in Fig. 1.

   Next, the algorithm finds a *blocking flow* on the level graph $\overline{G}^f$ of the residual network G^f. A blocking flow f in G is a flow such that each s-t path contains at least one *critical edge*. Recall that an edge e is *critical* if the amount of flow on it reaches its capacity: $f(e) = c(e)$. Fig. 2 gives examples for blocking flow.

   Dinic's algorithm is then described as follows.

   1. Initialize f to be the empty flow, and $G^f = G$.
   2. Do the following until there is no s-t path in G^f:
      - construct the level graph $\overline{G}^f$ of G^f.
      - find a blocking flow on $\overline{G}^f$.
      - Update f by adding the blocking flow to it, and update G^f.

   Complete the analysis of Dinic's algorithm by solving the following questions.

   (a) (15 points) Prove that, after each iteration of Dinic's algorithm, the distance from s to t in G^f is increased by at least 1.

   (b) (15 points) Design an $O(|V| \cdot |E|)$ time algorithm to compute a blocking flow on a level graph.

   (c) (10 points) Show that the overall time complexity for Dinic's algorithm is $O(|V|^2 \cdot |E|)$.

   (d) (20 points) **(challenging)** We have seen in the class that the problem of finding a maximum matching on a bipartite graph can be converted to the maximum flow problem. Show that Dinic's algorithm applied to finding a maximum matching on a bipartite graph only requires time complexity $O(|E| \cdot \sqrt{|V|})$.

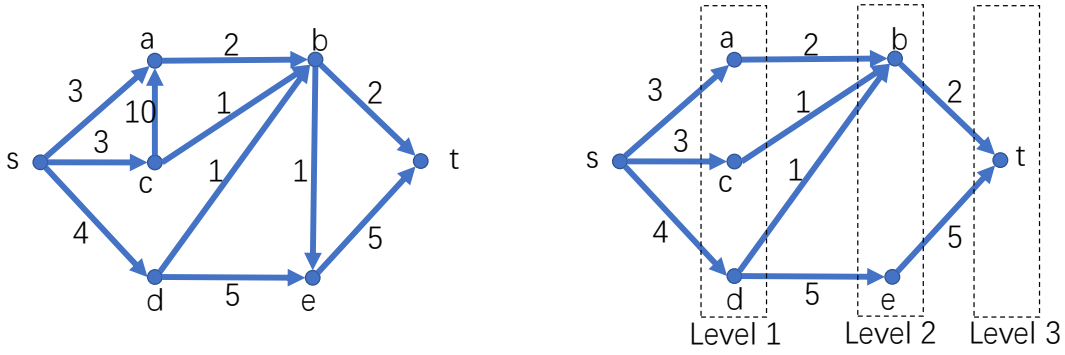

Figure 1: The graph shown on the right-hand side is the level graph of the graph on the left-hand side. Only edges pointing to the next levels are kept. For example, the edges (c, a) and (b, e) are removed, as they point at vertices at the same level. If there were edges pointing at previous levels, they should also be removed.

6. How long does it take you to finish the assignment (including thinking and discussion)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Please write down their names here.

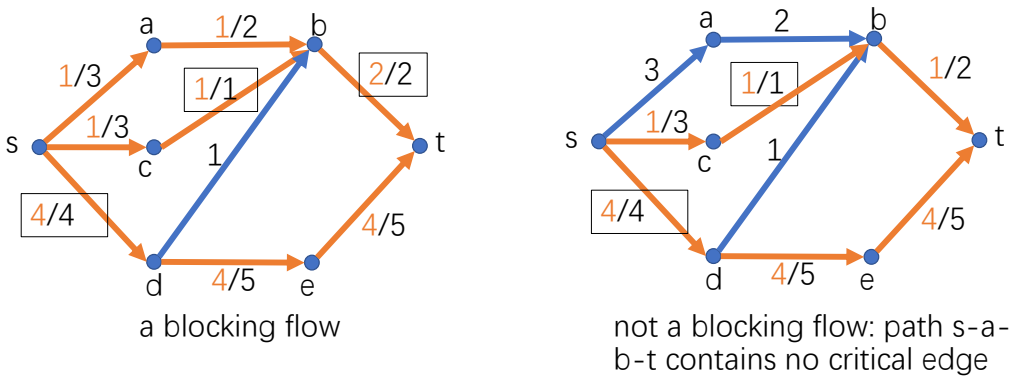

Figure 2: Blocking flow examples