CMPSC 465 Fall 2024 | Data Structures & Algorithms Mingfu Shao and Debarati Das | **Recitation 7**

Monday, Oct 21, 2024

1. Let $G = (V, E)$ be an directed graph with $s, t \in V$. Design an algorithm (by reducing to the max-flow problem) to find the maximum number of mutually edge-disjoint s-t paths in G. We define a set of s-t paths are mutually edge-disjoint if any two of them do not share any edge (they may share vertices).

   Solution: We construct a max-flow instance from the given directed graph $G = (V, E)$. We first remove all in-edges of s and out-edges of t, as they will not appear in any s-t paths; denote the new graph as $G' = (V, E')$. The network will be exactly G' and the source and sink vertices are the given s and t respectively. We set $c(e) = 1$, for any edge $e \in E'$.

   We then run any max-flow algorithm to find an integral maximum flow f^* of G'. Then we can construct $|f^*|$ mutually edge-disjoint s-t paths in G by using the edges in G' with $f^*(e) = 1$.

   This algorithm is optimal. In fact, G contains k mutually edge-disjoint s-t paths if and only if the value of maximum-flow in G' equals k. Above we proved one side, i.e., if the value of maximum-flow of G' equals k, then we can construct k mutually edge-disjoint s-t paths in G. On the other side, if there are k mutually edge-disjoint s-t in G, then we can construct a flow f with $|f| = k$ by setting $f(e) = 1$ for each edge in these k paths.

2. Given a directed graph $G = (V, E)$, with a source vertex $s \in V$ and a sink vertex $t \in V$, we define two paths from s to t are vertex-disjoint, if these two paths share no vertex except s and t. Describe a polynomial-time algorithm that finds the maximum number of vertex-disjoint paths from s to t.

   Solution: We transform this problem into the max-flow problem. We first build network $G' = (V', E', s', t', c')$ from the given graph $G = (V, E)$ as follows. For each original vertex $v \in V$ we add two new vertices v_1 and v_2 to V', and then connect them with a new edge (v_1, v_2) added to E'. For each original edge $e = (u, v) \in E$ we add edge (u_2, v_1) to E'. The source s' of G' will be s_1 and the sink t' of G' will be t_2. All edges in G' has a capacity of 1, i.e., $c'(e) = 1$ for any $e \in E'$.

   We now run any max-flow algorithm on G' to find a max-flow f^*. Since all capacities of G' are integers (i.e., 1), we can assume that $f^*(e) \in \{0, 1\}$ for any $e \in E'$. Hence, $|f^*|$ must be integer as well.

   We now prove that $|f^*|$ equals to the maximum number of vertex-disjoint paths from s to t in the original graph G. In fact, edges in G' with $f^*(e) = 1$ form $|f^*|$ vertex-disjoint paths in G' from s' to t', because each vertex in G' is either have in-degree of 1 or out-degree of 1; these $|f^*|$ paths therefore correspond to $|f^*|$ vertex-disjoint paths in G, by simply concatenating v_1 and v_2 into v. On the other side, if there exists k vertex-disjoint s-t paths in G, then they can be transformed into k vertex-disjoint s'-t' paths in G'. Combined, we conclude that $|f^*|$ equals to the maximum number of vertex-disjoint s-t paths in G (and these paths can be fetched following edges with $f^*(e) = 1$).

The running time of above algorithm consists of building network G', which takes $O(|V|+|E|)$ time, and running one max-flow algorithm, which takes polynomial-time.