Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

Summer semester 2000
Solutions 8
July 18, 2000

Efficient Algorithms and Data Structures II

Problem 1 (15 credits)

Let G be an undirected, bipartite graph with real edge weights.

a. Devise an efficient algorithm that determines a maximum cardinality maximum weight matching.

b. Devise an efficient algorithm that determines a maximum weight matching.

Solution.

Ad a. Assume all edge weights to be non-negative (otherwise add some sufficiently large number to all weights). We construct the auxiliary directed graph exactly like in the algorithm that determines a maximum cardinality matching with minimum weight, just that we now negate the weights of the unmachted edges. Then, a shortest path from s to t corresponds to a heaviest augmenting path (w.r.t. the alternating sum), and vice versa. Further, there can be no directed cycles with negative weight in the auxiliary graph, if the construction is perfomed with a maximum k-matching.
If G is a graph with n nodes and m edges, then a matching of maximum cardinality can be found in $O(n)$ phases. If for the determination of a shortest path the Bellman-Ford algorithm is used, then we have a total running time of $O(n^2m)$.

Ad b. Just like under **a**, for $i = 0, 1, 2, \ldots, M_G$, we compute a series of maximum i-matchings, where M_G denotes the maximum cardinality of any matching in G. If all edge weights are non-negative, we obtain a matching of maximum weight if we choose a heaviest matching from this series. Otherwise, add $C = -\min_{e \in E} c(e)$ to all weights, compute the series of matchings, and then choose a heaviest one, where the term iC has to be subtracted from the weight of an i-matching to obtain its true weight.

Problem 2 (15 credits)

a. For a graph G let $\mathcal{C}_G$ denote the set of connected components, and let $q(G)$ denote the number of components with odd cardinality.

   Theorem (Tutte, 1947): A graph $G = (V, E)$ has a perfect matching if and only if $q(G - S) \leq |S|$ for all $S \subseteq V$.

   Show that the condition of the theorem is necessary for the existence of a perfect matching.

b. A graph $G = (V, E)$ is called *factor critical* if $V \neq \emptyset$ and $G - \{v\}$ has a perfect matching for any $v \in V$. A vertex set $S \subseteq V$ is called *pairable with* $G - S$, if the graph H_S, which is obtained from G by contraction of all components in $\mathcal{C}_{G-S}$ and deletion of all edges within S, contains a matching of cardinality $|S|$.

Here, under contraction of a subgraph $S \subseteq G$, we understand the consecutive elimination of all edges within S and amalgamation of the endpoints of any eliminated edge, until just one vertex remains (which then is adjacent to all nodes $V - S$ that have been adjacent to any node of S before).

> **Theorem**: Each graph $G = (V, E)$ contains a vertex set S with the following properties.
>
> 1. S is pairable with $G - S$.
> 2. Each component of $G - S$ is factor critical.
>
> For each such S, G contains a perfect matching if and only if $|S| = |\mathcal{C}_{G-S}|$.

Show that Tutte's theorem follows from this theorem.

Solution.

Ad a. Assume that $q(G - S) > |S|$ for some $s \subseteq V$. Lee M be an arbitrary matching in G, and let C be an odd component of $G - S$ (by assumption there *is* at least one odd component). The there exists at least one node v_C in C that is not endpoint of an edge of $M \cap E(C)$ (the matching M restricted to $E(C)$). If v_C is matched in M, the other endpoint of the corresponding edge must be in S. Since by assumption there are more odd components than nodes in S, not all nodes can be matched in M. Hence, there is no perfect matching in G.

Ad b. With (1) and (2) of the theorem it directly follows that $|S| \leq |\mathcal{C}_{G-S}|$, since factor critical graphs have odd cardinality. From the Tutte condition $q(G - S) \leq |S|$ we can conclude that $|S| = |\mathcal{C}_{G-S}|$, and hence by the last statement of the theorem the existence of a perfect matching.

Problem 3 (10 credits)

Design a linear time algorithm to determine if a string T is a cyclic rotation of another string T'. For example, *particle* and *ticlepar* are cyclic rotations of each other. (Hint: Use the KMP algorithm.)

Solution.

Concatenate two copies of T and search for an occurence of T' in TT with Boyer-Moore, for instance.