ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

School of Computer and Communication Sciences

Exercise 4 **Graph Theory Applications**

Date: March 14, 2013 Spring 2013

Problem 1. Show that a tree without a vertex of degree 2 has more leaves than other vertices.

Problem 2. Consider a connected graph $G = (V, E)$ and its spanning tree $G'(V, E' \subset E)$. Let $d_G(u, v)$, $d_{G'}(u, v)$ denote the distance of two vertices u, v in G and in G' respectively. Show that unless G is a tree itself there is no spanning tree G' for which it holds that $d_G(u, v) = d_{G'}(u, v)$ for every $u, v \in V$.

Let r be an arbitrary vertex in G. Find a spanning tree G' for which $d_G(r, u) = d_{G'}(r, u)$ for all $u \in V$.

Problem 3. We want to create a peer-to-peer network to distribute some large files from a source node to n other peer nodes. Construct a network that has the following properties:

1. There are no parallel edges in the network.
2. The minimum possible number of edges is used.
3. There exist k edge-disjoint spanning trees in the resulting network (thus, assuming that each edge has unit capacity the source can simultaneously send k bits to all other nodes.)

What does the structure of this network look like? How many edges does it have? Can you describe a systematic way to create such a network for arbitrary number of nodes n? Assume that k is a fixed constant and $n/2$ is a much larger than k.

Problem 4. Show that a sequence $d_1, d_2, \ldots, d_n$, $n \geq 2$ of positive integers is the degree sequence of a tree with n vertices if and only if $\sum_{i=1}^{n} d_i = 2(n-1)$.

Problem 5. In a tree that has 100 vertices, how many nodes of degree 10 can be at most?

Problem 6. In class we have seen Kruskal's algorithm for finding an optimal minimum weight spanning tree of a graph. This algorithm is greedy: at any point we add an edge of minimum weight, only constrained by the fact that we are not allowed to create cycles. We will see the proof of the optimality of this algorithm in class next week.

In this exercise we will see a slightly more general perspective which explains in which situations greedy algorithms are optimal. The correct setting is the one of *matroids*. A matroid is a mathematical structure which abstracts the notion of *independence* which you are all familiar with from linear algebra.

Here is the definition. A matroid M is a tuple $M = (E, \mathcal{I})$, where E is a set of "elements" and $\mathcal{I}$) is a set of subsets of such objects.

Here is a concrete example. Think of E as the set of all binary vectors of length n and let $\mathcal{I}$ be the set of all possible subsets of *independent* vectors.

We say that M is a matroid if the following conditions are fulfilled. 1. The empty set is in $\mathcal{I}$. 2. If $A \in \mathcal{I}$ then any subset of A is also an element of $\mathcal{I}$. 3. For $A \in \mathcal{I}$, let $|A|$ denote its cardinality. If $A, B \in \mathcal{I}$ and $|A| > |B|$ then there exists an element of A, call it x, so that $B \cup \{x\} \in \mathcal{I}$.

The last property is often called the exchange property. An element of $\mathcal{I}$ of maximal cardinality is called a *bassis*. Note that due to the exchange property any basis must have the same cardinality and this cardinality is called the rank of the matroid.

1. For our example above where the elements are binary vectors, show that M is a matroid by showing that it fulfils all the above properties.

2. Let $G = (E, V)$ be a simple connected graph. Let $M = (E, \mathcal{I})$ be the matroid, where E is the set of edges, and the elements of $\mathcal{I}$ are all those subsets of edges so that these edges form a forest (i.e., there are no cycles). Show that this M indeed also forms a matroid.

There are many more examples of matroids. Why do we care? Assume now that all elements of E have non-negative weights. This is called a *weighted matroid*. Then the following simple greedy algorithm can be used to find a basis of minimum weight. Start with the empty set. At any stage add an element of minimum weight so that the resulting set is again independent by using the exchange property. It is a basic fact that this algorithm is optimal.

3. Show that this algorithm is equivalent to Kruskal's algorithm if we apply it to the minimum weight spanning tree problem.