
To pass the exercise component of the course, all but two of the assignments must be completed. This is not absolutely required but will be helpful in passing the course and can improve the grade. Up to **four** *people can work on an exercise together. But each of you should be able to explain the solutions to the TA (Bremser). Write your names* **and** *the name of your group (time, TA) on the sheets. Staple them together*

Assignment 1 Deadline: November 3, 2001 solve at least two of the following four exercises completely (or more of them partially)

Exercise 1
Describe a linear time algorithm for decomposing an undirected graph into a collection of paths and cycles. The output should be a sequence of arrays or linked lists that represent the paths and cycles. Paths should be avoided: Only odd degree nodes are allowed to be endpoints of a path. An odd degree node may be endpoint in at most one path. Each edge should appear in exactly one path or cycle. Give sufficient detail of the representation of the graph and the algorithm so that it becomes evident that your algorithm runs in time $\mathcal{O}(m+n)$.

Exercise 2

1. Give a detailed description of an algorithm for DFS in a directed graph that runs in time $\mathcal{O}(m+n)$. The algorithm should use the most compact adjacency array representation as input. Its additional space consumption should be $\mathcal{O}(n)$, i.e., no additional information "per edge" like edge markers can be stored.
2. Modify the SCC algorithm in such a way that it computes *component numbers*, i.e., if there are k SCCs it assigns numbers from $1..k$ to the nodes such that nodes in the same SCC get the same number $compNum(v)$. The component numbers should be *topologically sorted*, i.e., if there is a path from u to v then $compNum(u) \leq compNum(v)$. Prove that this property holds.

Exercise 3
Suppose you are fed a sequence of many edges of a graph over a network link. You do not have enough space to store all these edges. But you do have a few words of memory to store some information with each *node*. Outline an algorithm that computes a spanning forest of the graph. How fast can you make your algorithm. Can you achieve time $\mathcal{O}(m+n)$?

Exercise 4

1. Show that DFS of an undirected graph cannot encounter any cross edges.
2. The classification of edges into forward, backward, cross, and tree edges that we know from DFS also makes sense for BFS. Show that BFS will not produce any forward edges.
3. Suppose you attempt to translate our SCC algorithm based on DFS into an analogous algorithm based on BFS. What goes wrong. Give an example where the algorithm must produce the wrong output.