Algorithms

Judit Csima
csima@cs.bme.hu

BFS and DFS

1. Determine a BFS-tree and a DFS-tree for K_n (the undirected complete graph on n vertices).

2. Use DFS for the following directed graph G. Determine the DFS tree and the finishing times as well.
   G: **a**:b,c; **b**:-; **c**:d,e,f; **d**:f; **e**:b; **f**:-; **g**:d; **h**:d,b,g.

3. Find a topological ordering for the previous graph (given in Problem 2) using DFS.

4. A circus is designing a tower consisting of people standing atop one another's shoulders (at each layer there is exactly one acrobat). For practical and aesthetic reasons, each person must be both shorter and lighter than the person below him or her. Given the heights and weights of n acrobats, design an algorithm to determine the highest human tower the acrobats can form. (Let's suppose that any acrobat is strong enough to hold the others standing on his or her shoulders.) The height of a tower is the sum of the heights of the acrobats being in the tower. The algorithm should work in $O(n^2)$.

5. In an undirected graph G each edge is labelled by 1 or 2 or 3, the graph is given by an adjacency list. Design an algorithm to find the length of a shortest path from a given vertex v to all other vertices. The length of a path is the sum of the labels of its edges, the algo should finish in $O(n+m)$ steps.

6. An unweighted directed graph $G=(V,E)$, and two vertices $s,t \in V$ are given. Vertices of G are labeled red, green, or white, both s and t are red. Show how to find a minimum length admissible path from s to t (or determine that none exists), when a (not necessarily simple) path is admissible

   a) if it contains only red vertices. The algorithm should work in time $O(n+m)$.

   b) if it contains at most one edge from a white vertex to a green one, all the other vertices on the path are red. The algorithm should work in time $O(m \cdot (n+m))$.

   c) if it contains exactly one edge from a white vertex to a green one, all the other vertices on the path are red. The algorithm should work in time $O(n+m)$.

7. **optional HW for extra credit** We have n files, the length of the ith file is l_i (l_i is an integer for any $1 \le i \le n$), the order of the files is determined, we cannot change it. We can use two identical disks with capacity L (L is also an integer) to save the files. Our goal is to save as many files as possible, but the files must be saved in the given order, so we would like to determine the biggest k (and the corresponding saving strategy) such that the first k files can be saved. We cannot split the files, each file must be saved entirely into the first or entirely into the second disk. Design an algorithm to find the best saving strategy. The algo should work in $O(L^2)$.