École Polytechnique Fédérale de Lausanne
Prof. Eisenbrand

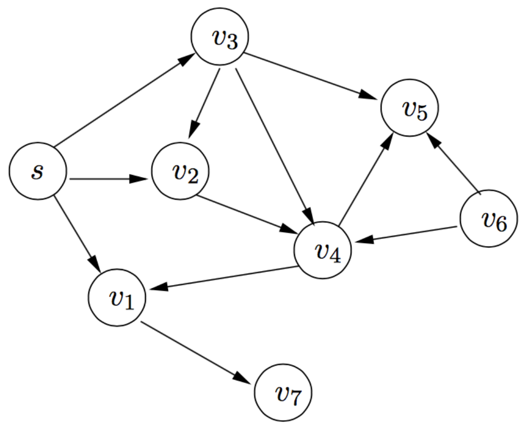
Jun 1, 2017

**Discrete Optimization** (Spring 2017)

# Assignment 13

## Problem 1

In an unweighted graph (i.e. where all edges are of the unit weight) the shortest path from $s$ to all other vertices can be computed by using the breadth-first search (BFS) algorithm. Apply it to the graph bellow.




Consider each iteration of the BFS (i.e. each time when a new vertex is taken from the head of the queue). Note which vertex has been currently processed, the distance labels and the snapshot of the queue at the end of the iteration.

## Problem 2

There are $n$ types of animals, and you want to assign them to two stables. Unfortunately, some animals would eat other animals when left unattended. Therefore you need to assign the animals carefully. There are $m$ relations of the form "$u$ eats $v$", where $u$ and $v$ are animals.
Find an $O(n + m)$ algorithm that decides whether there is an assignment of animals to the two stables such that no animal eats another one of the same stable, and outputs a feasible assignment.

*Hint:* Observe that there is a feasible assignment of the animals if and only if the underlying (undirected) graph $G = (V, E)$ is bipartite. Modify the BFS algorithm seen in class so that it can be run even if $G$ has several connected components (i.e. there is no path between each two nodes in $G$) and if $G$ is undirected. Use this modified BFS to check if $G$ is bipartite.

## Problem 3

In this problem we consider a directed graph $G = (V, A)$ with $n$ vertices and $m$ arcs.

(a) A toplogical sort of the vertices is an ordering of the vertices such that there is no edge from a vertex $u$ to $v$ if $u$ is placed after $v$ in the ordering.

Formulate an $O(m + n)$ algorithm that finds a topological sort of the vertices or decides that there is a directed cycle in $G$.

(b) Assume for this part that $G$ is acyclic, *i.e.*, there exists no directed cycle in $G$. Show how the above ordering can be used to compute single source shortest paths in a single pass using the Bellman-Ford algorithm.

(c) We will now see how the previous observation can be used to reduce the total number of iterations of the main loop in Bellman-Ford. The idea is to take an arbitrary vertex ordering $(v_1, \dots, v_n)$ and split the edge set $E$ into two sets $E_1 = \{v_i v_j \mid i < j\}$ and $E_2 = \{v_i v_j \mid i > j\}$. Both, $G_1 := (V, E_1)$ and $G_2 := (V, E_2)$, are directed acyclic graphs. How can you make use of this observation to reduce the number of iterations of the main loop in Bellman-Ford?

**Problem 4**

Given $n$ numbers $a_1, \dots, a_n$ find indices $i$ and $j$, $1 \le i \le j \le n$, such that $\sum_{k=i}^{j} a_k$ is minimized. We will develop two algorithms for this problem that run in linear time, *i.e.*, the number of operations is linear in $n$.

(a) Solve the problem using Bellman-Ford as a subroutine. In particular, construct a graph such that a shortest path in this graph yields the optimal solution to the above problem. Show that the graph can be generated in linear time and that Bellman-Ford can be implemented to run in linear time on this graph.

(b) Define $d(j) = \min_{1 \le i \le j} \sum_{k=i}^{j} a_k$. Conclude that the above problem is equivalent to computing $\min_{1 \le j \le n} d(j)$. Show how this can be done in linear time.