S. Schwoon / D. Suwimonteerabuth Winter Term 2008/09

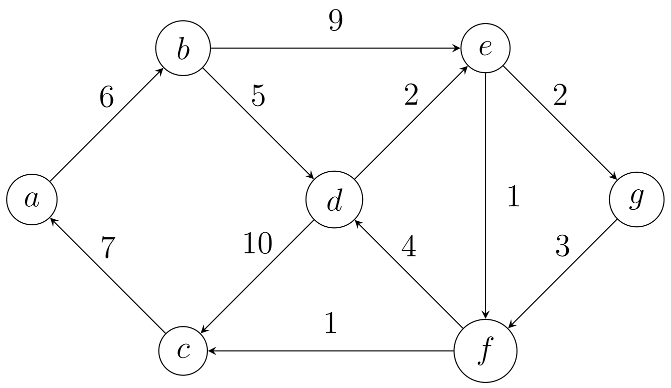
Fundamental Algorithms

Exercise Sheet 9

1. In the *knapsack* problem, we are given a knapsack, a weight W, and n types of objects, numbered 1 through n. Assume that there are enough objects of each type available. Each object type $i \in \{1, \ldots, n\}$ has a positive weight w_i and a positive value v_i. The knapsack can only be filled with objects whose total weight does not exceed W. The aim is to fill the knapsack in a way that maximizes the value of the included objects. It is not allowed to take a fraction of an object.

   Design an algorithm that solves the knapsack problem by outputting the maximum value. An easy solution is to view the problem as a search in a directed graph, where nodes contain information about objects in the knapsack. An edge in the graph corresponds to an action of putting an object into the knapsack. The search is started from the node that represents the empty knapstack, and repeats putting new objects into it. Notice that there is no need to construct the graph explicitly.

2. Consider the following weighted graph:



   (a) Find the shortest distances from a to all other nodes.

   (b) Change the weight of the edge (g, f) to -3, and run the Dijkstra's algorithm to calculate all distances from the node a. Is the output of the algorithm correct? Justify your answer.

   (c) Now modify in addition to (b) the weight of (f, d) to -4. What is the shortest path from a to d?

   (d) Give an algorithm that correctly solves the shortest-path problem when negative weights are allowed. How does your algorithm behave with the modified graph (c)? Analyze the complexity.