Informatics 2 – Introduction to Algorithms and Data Structures

Tutorial 7 - Dynamic Programming

1. Consider the weighted directed graph $G = (V, E)$ of Figure 1. Run the Bellman-Ford algorithm to compute the value of $M[i, v]$ for every node $x \in V$. Recall that $M[i, v]$ is the cost of the minimum-cost $v \sim t$ path that uses at most i edges.

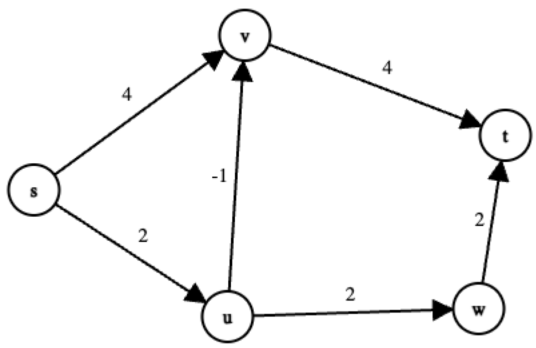

Figure 1: A directed graph with edge costs indicated. Algorithms Iluminated Example 18.2.6.

2. Assume that we wanted to use the Bellman-Ford algorithm to find the cost of the minimum-cost paths from a node s to all the nodes $x \in V$ in the graph G. Think about how to modify the algorithm to achieve this and run the modified algorithm on the graph of Figure 1 to compute the costs of all the minimum-cost paths from s to the nodes in V.

3. Consider the knapsack problem given by the following table, with capacity $W = 7$.

Item	Value	Weight
1	1	1
2	2	3
3	3	2
4	4	5
5	5	5

Use the dynamic programming algorithm presented in the lectures to compute the value of the optimal solution.

4. Recall the following simple context-free grammar for arithmetic expressions from Lecture 21. The start symbol is Exp.

$$\begin{aligned}
\mathsf{Exp} &\rightarrow \mathsf{Var} \mid \mathsf{Num} \mid (\ \mathsf{Exp}\ ) \\
\mathsf{Exp} &\rightarrow \mathsf{Exp}\ +\ \mathsf{Exp} \\
\mathsf{Exp} &\rightarrow \mathsf{Exp}\ *\ \mathsf{Exp} \\
\mathsf{Var} &\rightarrow x \mid y \mid z \\
\mathsf{Num} &\rightarrow 0 \mid \ \cdots \ \mid 9
\end{aligned}$$

   (a) How many syntax trees are there for each of the following three strings? Draw them all.

$$3 + x * y \qquad\qquad 3 + (x * y) \qquad\qquad z + 10$$

   (b) Design a new context-free grammar that generates exactly the same language as the one above, but with the property that it is *unambiguous*: every string in the language should have exactly one syntax tree. Informally, your grammar should enforce the familiar convention that * takes precedence over +. You will find it helpful to introduce some additional non-terminal symbols.

   [Hint: First try to do this for the grammar with the rule for $\mathsf{Exp} * \mathsf{Exp}$ omitted. To ensure that a string like $3 + 4 + 5$ has only one tree, you might want to draw inspiration from the grammar for comma-separated lists in Lecture 21. Then try to adapt your grammar to cater for *, building in the precedence rule.]

   (c) For the grammar you have designed in part (b), draw the *unique* syntax tree for any of the strings from part (a) that had more than one syntax tree with respect to the original grammar.