Combinatorial Optimization
Lab No. 8
Dynamic Programming

Industrial Informatics Research Center

April 11, 2017

Abstract

This lab is devoted to dynamic programming. By considering the coins change problem, we show in lab step by step, the main principles of dynamic programming.

1 Introduction

Dynamic programming is a general and powerful algorithmic paradigm in which a problem is solved by identifying a collection of subproblems (hopefully polynomially many). The subproblems are tackled one by one, smallest first, and by using the answers to small subproblems, the larger ones are solved until the whole subproblems are solved. This implies that every dynamic program has an underlying implicit *Directed Acyclic Graph* (DAG) structure: think of each node as representing a subproblem, and each edge as a precedence constraint on the order in which the subproblems can be tackled. Having node $u_1, \ldots, u_k$ point to node v means that subproblem v can only be solved once the answers to $u_1, \ldots, u_k$ are known. Hence, there is an ordering on the subproblems, and a recursive relation that shows how to solve a subproblem given the answers to smaller subproblems, that is, subproblems that appear earlier in the ordering. By solving the subproblems from the smallest to the largest, we achieve polynomial run time in the number of subproblems.

1.1 Coins exchange problem

The coins change problem is a kind of vending machine problem where there is an unlimited supply of coins of denominations $D = \{d_1, d_2, \ldots, d_n\}$, and we wish to make change for a value V; that is, we wish to find a set of coins whose total value is V. This might not be possible: for instance, if the denominations are 5 and 10 then we can make change for 35 but not for 12.

There might be other variations of the change-making problem as follows:

1. given a set of denominations $D = \{d_1, d_2, \ldots, d_n\}$, you want to make change for a value V, but you are allowed to use each denomination d_k at most once.
2. given a set of denominations $D = \{d_1, d_2, \ldots, d_n\}$. Each denominations d_k has an upper limit $l_k \geq 1$, and you want to make change for a value V, but you are allowed to use denominations d_k at most l_k times.

The optimization problem considers the minimization of the cardinality of the solution coin set.

Lab exercises:

1. Consider an unlimited supply of coins of denominations $D = \{1, 10, 25, 50, 100\}$, and $V = 380$. Show by using the dynamic programming whether there is a set of coins that sums up to V. If yes, print one of the possible sets.
2. Consider the optimization problem for the same problem instance as given by exercise 1.

1.2 Optimal coin system design

We have seen that different coin set D we can have various possibilities how to sum to V. Therefore, an interesting question is arising, which denominations D of coins you should consider as a ruler of a newly established country.

We will assume that the most commonly occurring transactions are $V \in \{1, 2, \ldots, 100\}$. We say, that the quality of coin system D of size $|D| \leq k$ is measured as

$$\text{objective}(D) = \sum_{v=1}^{100} \text{the minimum number of coins from } D \text{ that sums to value } v.$$

Therefore, it minimizes the number of coins needed to pay in the most common transactions. The question is, which D has the lowest objective with given size (number of different denominations) k:

$$\begin{aligned} &\min_{D} \ \text{objective}(D) \\ \text{subject to} \\ &|D| \leq k \end{aligned}$$

Lab exercise: What is the optimal coin system D with at most $k = 3$ denominations? What is your guess and what is the reality?