

# **Laboratório de Programação Avançada 2018/19**

## **Week 1 – Problem Solving**



UNIVERSIDADE DE COIMBRA

# Outline

1. Introduction
2. Decision vs. Optimization
3. Problem reduction

## Reading about problem reduction

- ▶ T. Cormen et al., Introduction to Algorithms, Chapter 34
- ▶ S.S. Skiena, The Algorithm Design Manual, Section 9.1

Our aim is to solve (well-defined) problems in an efficient way

- Efficiency is understood in terms of CPU-time and/or memory.
- We can solve a problem if we can recast it as another similar problem for which an algorithm is known.
- Otherwise, we use a generic problem-solving strategy, such as backtracking, dynamic programming, etc..

## What is a problem?

An abstract question that might be solved by a computer.

- There are two entities in a problem:
  - Instances
  - Solutions
- Problem instances may have no solution, one solution, or multiple solutions.

# Introduction

## Example: The Travelling Salesman Problem (TSP)

- An instance is characterized by of a set of cities and the distances between them
- A solution is a sequence of cities, describing the order in which they should be visited.



Figure credit: David Applegate, Robert Bixby, Vasek Chvatal and William Cook.

## Types of problems

- Counting problems: the output is the number of solutions.
  - Count the number of tours.
- Decision problems: the output is either yes or no.
  - Does exist a tour with a length  $\leq k$ ?
- Optimization problems: the output is the minimum/maximum.
  - Find the tour with the minimum length.

An optimization problem has a corresponding decision problem.

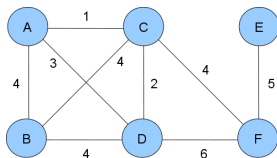
## Solution strategy

- We can solve an optimization problem by solving a sequence of related decision problems.
  - Perform binary search on  $k$ .
- We can solve a decision problem by solving the related optimization problem.
  - Check whether optimal value is  $\leq k$ .



## Decision vs. Optimization

- **Decision spanning tree:** Given a weighted graph  $G$  and  $k$ , does  $G$  have a spanning tree of total weight  $\leq k$ ?
- **Minimum spanning tree:** Given a weighted graph  $G$ , find a minimum spanning tree.



## Decision vs. Optimization

Given a knapsack of capacity  $W$  and  $n$  objects with weights  $w_1, \dots, w_n$  and values  $v_1, \dots, v_n$ .

- **Decision knapsack**: Given  $k$ , is there a subset of objects that fits in the knapsack and has total value  $\geq k$ ?
- **Optimization knapsack**: Find the subset that fits the knapsack and that maximizes the total value.

**Reduction** is a transformation of one problem into another problem (A reduces to B):

1. Transform a problem A into a problem B for which an algorithm is known.
2. Transform the solution of problem B into the solution of problem A.

We are very often interested in polynomial reductions: a reduction that takes polynomial amount of time.

### Subset sum problem

Given a set of integers  $\{s_1, \dots, s_n\}$ , is there a subset whose sum is  $B$ ?

### Partition problem

Given a set of integers  $\{p_1, \dots, p_m\}$  whose sum is  $S$ , can it be partitioned into two subsets with the same sum?

Assume that you have an algorithm for the subset sum problem.

Can you solve the partition problem?

Can you reduce the partition problem to the subset sum problem?

### Subset sum problem

Given a set of integers  $\{s_1, \dots, s_n\}$ , is there a subset whose sum is  $B$ ?

### Partition problem

Given a set of integers  $\{p_1, \dots, p_m\}$  whose sum is  $S$ , can it be partitioned into two subsets with the same sum?

Assume that you have an algorithm for the subset sum problem.

Can you solve the partition problem?

Can you reduce the partition problem to the subset sum problem?

Set  $n = m$ ,  $s_i = p_i$  and  $B = S/2$ . Solve the subset sum problem and check the output.

## Problem reduction

- In LPA, you can solve most of the problems by using **reduction techniques**. You need to recognize the underlying problem.
- Or use a **general strategy**: Break the problem down into smaller problems which you can solve, and devise how to recover the solution from the partial solutions found
- This is the main strategy of backtracking, dynamic programming, greedy algorithms and branch-&-bound.
- To know how to break the problem in the most effective manner requires a lot of training.