TIE-20106 DATA STRUCTURES AND ALGORITHMS

Bibliography

These lecture notes are based on the notes for the course OHJ-2016 Utilization of Data Structures. All editorial work is done by Terhi Kilamo and the content is based on the work of Professor Valmari and lecturer Minna Ruuska.

Most algorithms are originally from the book Introduction to Algorithms; Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

In addition the following books have been used when completing this material:

- Introduction to The Design & Analysis of Algorithms; Anany Levitin
- Olioiden ohjelmointi C++:lla; Matti Rintala, Jyke Jokinen
- Tietorakenteet ja Algoritmit; Ilkka Kokkarinen, Kirsti Ala-Mutka

1 Introduction

Let´s talk first about the motivation for studying data structures and algorithms

Algorithms in the world

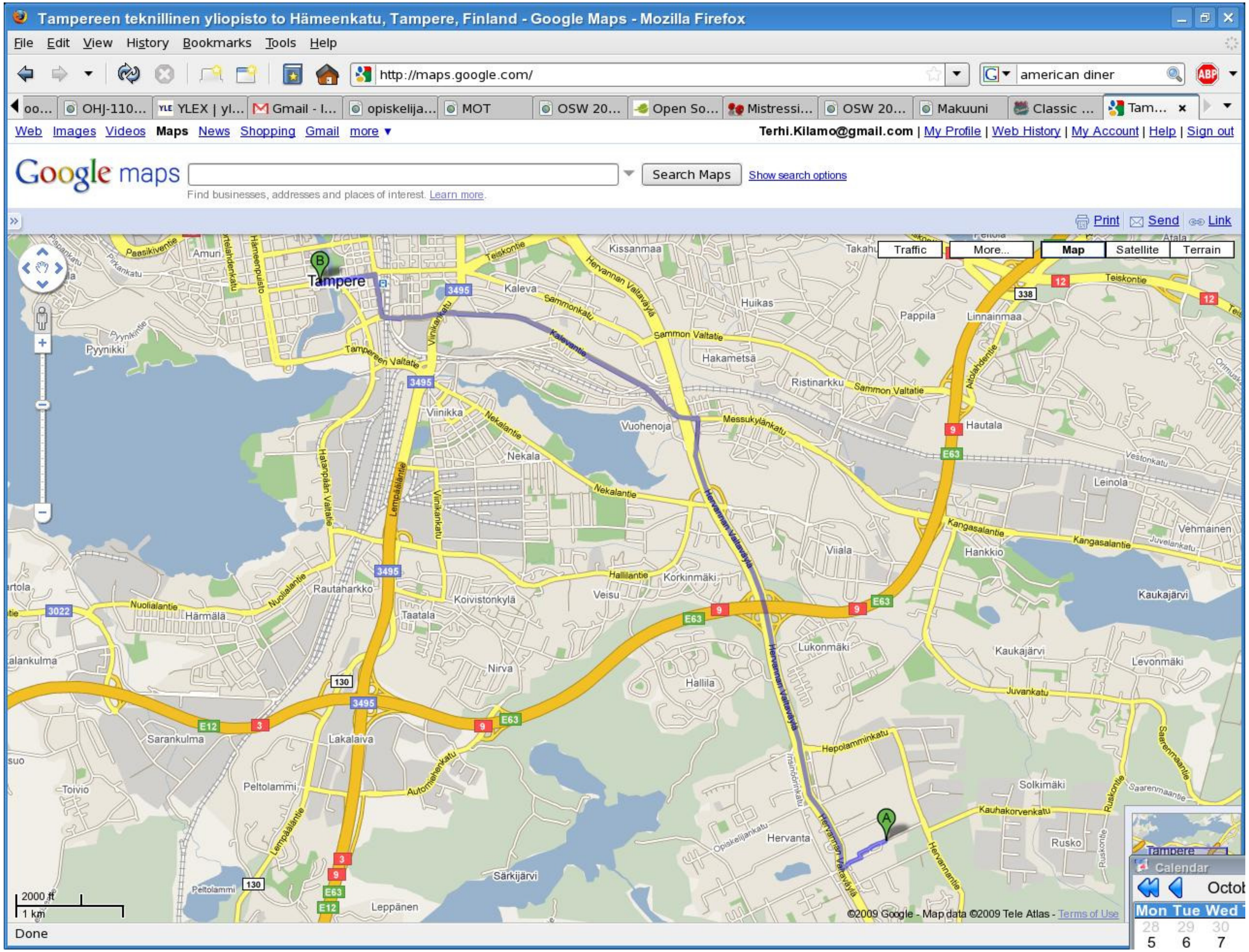
1.1 Why?

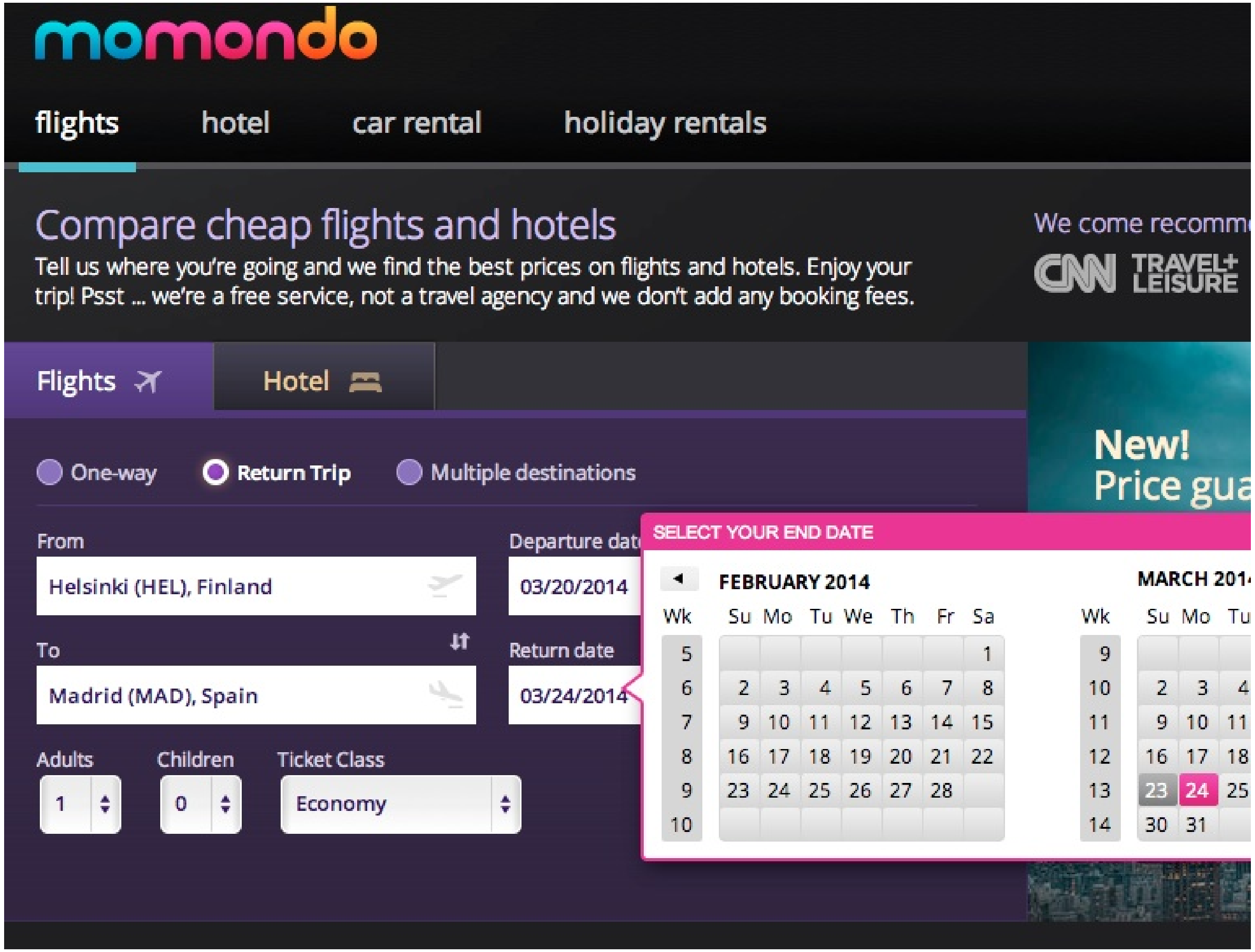
What are the three most important algorithms that affect YOUR daily life?

Picture: Chris Watt

There are no computer programs without algorithms

- algorithms make for example the following applications possible:


Tampereen teknillinen yliopisto to Hämeenkatu, Tampere, Finland - Google Maps - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://maps.google.com/
american diner
oo... OHJ-110... YLEX | yl... Gmail - I... opiskelija... MOT OSW 20... Open So... Mistressi... OSW 20... Makuuni Classic ... Tam...
Web Images Videos Maps News Shopping Gmail more ▾
Terhi.Kilamo@gmail.com | My Profile | Web History | My Account | Help | Sign out
Google maps
Search Maps
Show search options
Find businesses, addresses and places of interest. Learn more.
Print Send Link
Traffic More... Map Satellite Terrain
Tampere
Kaleva
Huikas
Pappila
Linnainmaa
Hakametsä
Ristinarkku
Pyynikki
Viinikka
Vuohenoja
Hautala
Nekala
Leinola
Vehmainen
Hankkio
Viiala
Korkinmäki
Veisu
Rautaharkko
Koivistonkylä
Taatala
Härmälä
Kaukajärvi
Lukonmäki
Nirva
Hallila
Levonmäki
Sarankulma
Lakalaiva
Peltolammi
Solkimäki
Toivio
Hervanta
Rusko
Särkijärvi
Leppänen
2000 ft
1 km
©2009 Google - Map data ©2009 Tele Atlas - Terms of Use
Calendar
Mon Tue Wed
28 29 30
5 6 7
Done



algorithms are at work whenever a computer is used

Data structures are needed to store and acces the data handled in the programs easily

- there are several different types of data structures and not all of them are suitable for all tasks
  $\Rightarrow$ it is the programmer's job to know which to choose
  $\Rightarrow$ the behaviour, strengths and weaknesses of the alternatives must be known

Modern programming languages provide easy to use library implementations for data structures (C++ standard library, JCF). Understanding the properties of these and the limitations there may be for using them requires theoretical knowledge on basic data structures.

Ever gotten frustrated on a program running slowly?

- functionality is naturally a top priority but effiency and thus the usability and user experience are not meaningless side remarks
- it is important to take memory- and time consumption into account when making decisions in program implementation
- using a library implementation seems more straightforward than is really is

This course discusses these issues

2 Terminology and conventions

This chapter covers the terminology and the syntax of the algorithms used on the course.

The differences between algorithms represented in pseudocode and the actual solution in a programming language is discussed. The sorting algorithm INSERTION-SORT is used as an example.

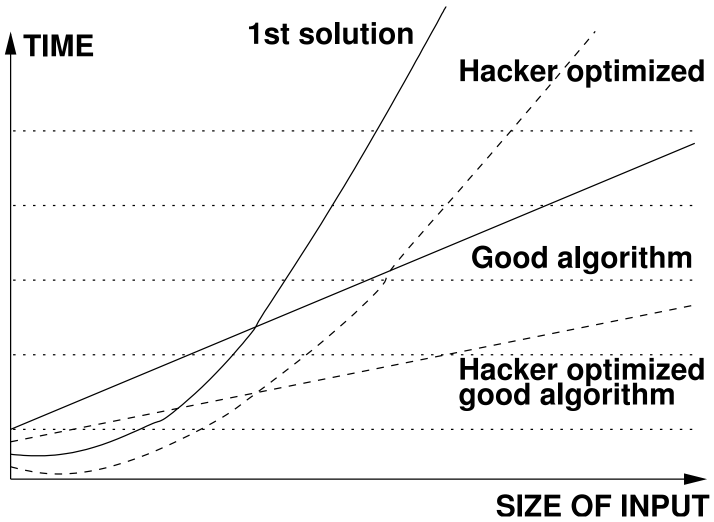
2.1 Goals of the course

As discussed earlier, the main goal of the course is to provide a sufficient knowledge on and the basic tools for choosing the most suitable solution to a given programming problem. The course also aims to give the student the ability to evaluate the decisions made during the design process on a basic level.

The data structures and algorithms commonly used in programming are covered.

- The course concentrates on choosing a suitable data structure for solving a given problem.
- In addition, common types of problems and the algorithms to solve them are covered.

- The course concentrates on the so called “good algorithms” shown in the picture on the right.
- The emphasis is on the time the algorithm uses to process the data as the size of the input gets larger. Less attention is paid to optimization details.

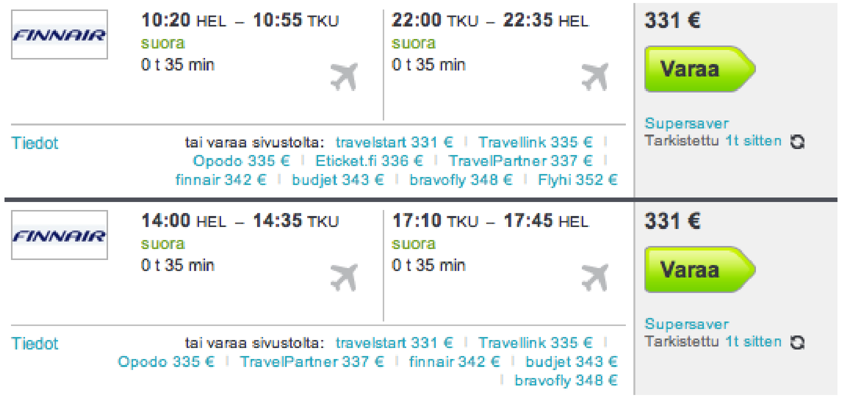


2.2 Terminology

A *data structure* is a collection of related data items stored in a segment of the computer's memory.

- data can be added and searched by using suitable algorithms.
- there can be several different levels in a data structure: a data structure can consist of other data structures.

An *algorithm* is a well defined set of instructions that takes in a set of input and produces a set of output, i.e. it gives a solution to a given problem.



- well defined =
  - each step is detailed enough for the reader (human or machine) to execute
  - each step in unambiguous
  - the same requirements apply to the execution order of the steps
  - the execution is finite, i.e. it ends after a finite amount of steps.

An algorithm solves a well defined problem.

- The relation between the results and the given input is determined by the problem
- for example:
  - sorting the contents of the array

    input: a sequence of numbers $a_1, a_2, \ldots, a_n$

    results: numbers $a_1, a_2, \ldots, a_n$ sorted into an ascending order
  - finding flight connections

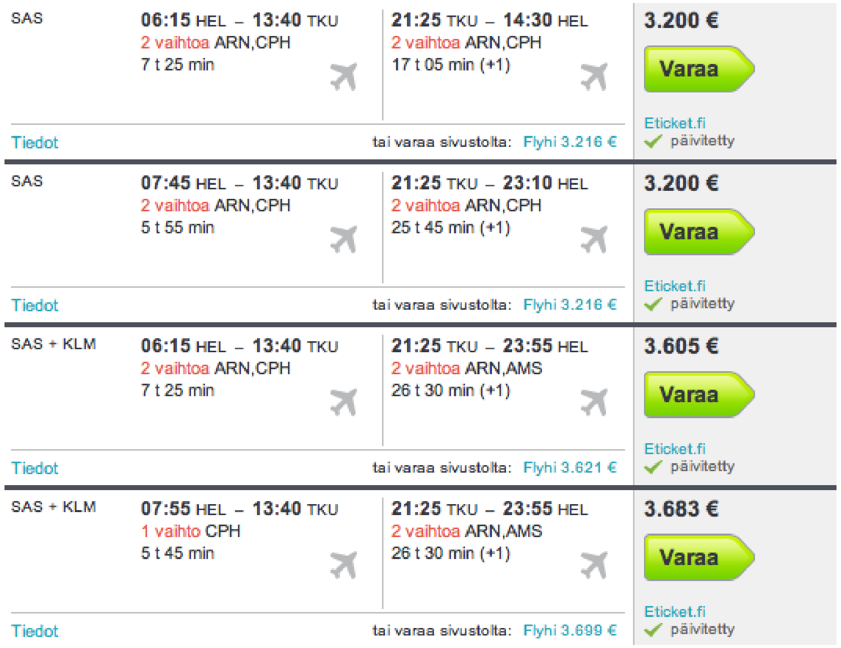
    input: a graph of flight connections, cities of departure and destination

    results: Flight numbers, connection and price information

- an instance of the problem is created by giving legal values to the elements of the problem's input
  - for example: an instance of the sorting problem: 31, 41, 59, 26, 41, 58

An algorithm is *correct*, if it halts and gives the correct output as the result each time it is given a legal input.

- A certain set of formally possible inputs can be forbidden by the definition of the algorithm or the problem



an algorithm can be incorrect in three different ways:

- it produces an incorrect result
- it crashes during execution
- it never halts, i.e. has infinite execution

an incorrect algorithm may sometimes be a very usefull one as long as a certain amount of errors is tolerated.

- for example, checking whether a number is prime

In principle any method of representing algorithms can be used as long as the result is precise and unambiguous

- usually algorithms are implemented as computer programs or in hardware
- in practise, the implementation must take several "engineering viewpoints" into accout
  - accomodation to the situation and environment
  - checking the legality of inputs
  - handling error situations
  - limitations of the programming language
  - speed limitations and practicality issues concerning the hardware and the programming language
  - maintenance issues $\Rightarrow$ modularity etc.

  $\Rightarrow$ the idea of the algorithm may get lost under the implementation details