# Advanced Data Structures

## Rationale

Efficient computing involves the use and maintenance of advanced data structures in a wide variety of algorithms used in Data Sciences. This course covers the theory and algorithms for these advanced data structures.

## Course Description

Data structures are a building block for algorithms. A data structure models some abstract structure with a specified set of operations. These include

- maintaining a set under insertions, deletions, and find operations, or
- maintaining a linear order under insertions, deletions, and comparisons, or
- maintaining a graph under insertion of edges and queries whether two points are in the same connected component, or
- answering orthogonal range queries for a point set, or
- answering substring queries for a string

In each of these situations, we have a well-specified behavior of what the structure should do, but it is not immediately clear how we should achieve that behavior. This is an algorithmic question: to design and analyze algorithms that realize the required operations, and answer questions like

How fast can we perform the operations? Can we do better if we allow amortized complexity instead of worst-case complexity? How much space does the structure need? Does the structure support changes, or only queries? How can we access past versions of the structure?

Once we know good methods to realize these structures, they are available as components to higher-level algorithms, like the heap in Dijkstra's shortest-path algorithm, or the set-union strukture in Kruskal's minimum spanning tree algorithm; whenever an algorithm performs such operations often, one should look for the most efficient realization of that operation.

In this course we will study many data structures, their analysis and implementation.

# Topic List

The topic list may include but is not limited to:

- Search and update sets of numbers, intervals, or strings by various data structures such as
  - Search trees
  - Structures for sets of intervals
  - Structures for piece-wise constant functions
  - Structures for orthogonal range searches
  - Heaps
  - Union-find structures
- Dynamization and persistence of structures
- Structures for strings
- Structures for hash tables.

# Learning Goals

Be able to determine computational complexity of and program algorithms for

- set maintenance: insertions, deletions, and find operations
- maintaining linear under under insertions, deletions and comparisons
- inserting or deleting edges in a graph
- determining whether two graph nodes are in the same connected component
- determining whether one string is a substring of another string

# Assessment

There will be four implementation homeworks of structures; test code will be provided, and homework submissions are not accepted until they pass this test. Programs are to be written in C or C++. and will be tested in a linux environment using gcc/g++.