Algorithm Engineering (2014, Q3)
Project 1 – Binary Search

In this project you should do a throughout experimentation with several binary search algorithms.

The project should be done in groups of 2-3 persons. The work should be documented in the group's final report in the course, and the evaluation of the final report will be part of the final grade.

The data structures considered should store a set S containing N integers and support the query:

$$\text{Pred}(x) = \text{return } \max \{ y \in S \mid y \leq x \} .$$

A simple solution is a sorted array and the query is solved using binary search. Another solution is a binary search tree. Further ideas on how to e.g. reduce cache misses and branch-mispredictions can be found in the papers covered in the lectures. In this project only comparison based structures should be considered.

Through experimental evaluation of the performance of the Pred operation, try to optimize your structure. Your report should document these steps of improvements, comparing the steps against each other.

During the project you should:

- Measure the running time for different input instances. Measure aspects that would possibly affect the running time, e.g. number of cache-faults on the different levels of the memory hierarchy, branch-mispredictions, TLB misses, number of instructions, number of comparisons, …
- Design experiments. What input should the data structures be tested on? For each input instance, or class of instances, what is the expected theoretical performance, e.g., what is the expected number of cache misses? For the different classes of test instances, what is the expected best possible choice of data structure?
- Make a framework to perform systematic testing (can be as simple as a few scripts). Can also include scripts for automatically generating a sequence of plots.
- Plot the measured data. How should the data be plotted to show useful insights? E.g., linear or logarithmic axis, data divided by *n*, data divided by theoretical bounds, data1 divided by data2, … How should the data be plotted to support (or disprove) your theoretical expectations?

Optional project:
Extend your work to consider dynamic sets supporting insertions and/or deletions in addition to the Pred operation. How can search trees e.g. rebalanced while keeping a good memory layout? What are good rebalancing schemes? Are there trade-offs between the performance of updates and queries?