

---

## Exercise Sheet 6

Submit until Wednesday, June 13 at 4:00pm

### Exercise 1 (12 points)

Implement the central contraction routine for Contraction Hierarchies. As usual, consider the implementation advice given in the lecture and the code design suggestions linked from the Wiki. In particular, take care that the Dijkstra searches per contraction take only very little time, as explained in the lecture.

It is key that you write a good *unit test* for this routine. Do not only test the contraction of single nodes on the original graph, but also make sure that *successive* contractions (where later contractions ignore the nodes and arcs removed in previous contractions) work as they should. For example, you could take the example graph from the lecture, and test whether contracting the nodes in that order leads to the required shortcuts. Be aware that, depending on your implementation, additional shortcuts may be added.

### Exercise 2 (6 points)

Run your contraction routine on the first 1000 nodes in a random node ordering. Consider the code design suggestions for the method *computeRandomNodeOrdering*. It suffices if you do this for one of our two datasets, preferably *BaWü*.

As usual, report your results in a row on the table linked from the Wiki. In particular, report the average time for a single contraction, and a “histogram” (see explanation on the Wiki) of (1) the number of shortcuts added per contraction, and (2) the *edge difference* per contraction. The edge difference is simply defined as the number of shortcuts added *minus* the number of arcs removed when contracting the node. Note that the edge difference can take on negative as well as positive values (as well as zero).

### Exercise 3 (2 points)

As usual, commit your code to our SVN and check that everything works on Jenkins, and also commit a text file *feedback-exercise-sheet-6.txt* where you briefly describe your experiences with this exercise sheet and the corresponding lecture.