Chair for Algorithms
and Data Structures
Prof. Dr. Hannah Bast
M. Brodesser/E. Sawin

Efficient Route Planning
SS 2012

http://ad-wiki.informatik.uni-freiburg.de/teaching

UNI FREIBURG

Exercise Sheet 3

Submit until Wednesday, May 16 at 4:00pm

Exercise 1 (6 points)

Extend your implementation of Dijkstra's algorithm so that it can also run in A* mode. Consider the implementation advice given in the lecture and the design suggestions linked from the Wiki.

Exercise 2 (6 points)

Implement A* with the landmark heuristic. Consider the implementation advice given in the lecture and the design suggestions linked from the Wiki.

Exercise 3 (6 points)

Run the A* algorithm for 100 random queries for both of our OSM graphs. Do this experiment for both of the heuristics discussed in the lecture: the straight-line heuristic (Exercise 1, no separate class needed for that) and the landmark heuristic (Exercise 2) with 42 landmarks.

As usual, report your results in a row on the table linked from the Wiki. In particular, for each of the two heuristics, report the average query time and the number of settled nodes. Also report the time to pre-compute the landmark distances. For the sake of simplicity and comparability, the query times should be measured *without* the time for computing the heuristic function.

Exercise 4 (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-3.txt* where you briefly describe your experiences with this exercise sheet and the corresponding lecture.

Exercise 5 (0 points, good exercise for exam preparation)

Verify the correctness proof of the A* algorithm. Think about how to extend the proof when the $\text{dist}(s, u) + h(u)$ are not all different.

Verify that the triangle inequality for *eucl* implies that the straight-line heuristic is admissible and monotone. Verify that for our graphs *dist* satisfies the triangle inequality.

Show that Dijkstra's algorithm, when started from a whole *set* S of nodes, as explained in the lecture, computes $\text{dist}(S, u) := \min\{\text{dist}(s, u) : s \in S\}$ for every node u in the graph.