Explanations and supplements to milestone II: Priority Queues

1. Assignment - Understanding Fibonacci-Heaps (see exercise at Dec 1st)

Define the functions of the ADT Fibonacci-Heap and explain the principle of amortized runtime analysis.

Construct on paper a Fibonacci-Heap by insertion the following priorities from left to right and explain your approach.

3, 19, 21, 4, 6, 1, 17, 33, 9, 18, 29, 24, 59, 5, 25, 36, 2

Apply the function DeleteMin to the constructed Fib-Heap and draw the result on paper.

Decrease the following priorities that are stored in the Fib-Heap. Apply the path shortening heuristic. What does it mean if a node in the heap has been marked?

1. 6 → 20
2. 21 → 28
3. 24 → 1
4. 59 → 58
5. 19 → 18

2. Make experiments with the priority queue class of Java

You do not need to implement the Fibonacci-Heaps in Java. Instead use the build-in priority queue class of Java and analyse its performance. Fore that you can use the experimental test environment of milestone I. Your results can be compared with the results of the teams that test the performance of Fibonacci-Heaps.

Proceed as follows:

1. Take n randomly generated priorities and insert them to a priority queue (Java)
2. Invoke the function DeleteMin to the priority queue and measure its runtime $T_{DeleteMin}(n)$.
3. Increase n by a suitable value e.g. by 1.000 or 10.000 and repeat steps 1 and 2 of the experiment.
4. Use the powertest to show all the results in a n - T(n)-diagram.
5. Explain the results.

3. Present your experiments at the milestone II date

As known from milestone I prepare a presentation and documentation about this assignment. Please send the documentation as a PDF to alex.maier@fh-koeln.de and heinrich.klocke@fh-koeln.de with the subject **„<teamname>_MS_<number of mildestone>“.**