- In week 47 we will start with "Graph Algorithms".
- Next week we will start with Chapter 11.

Some suggestion for the mandatory assignment 3. Please see this project as a possibility to combine your own research or study interests with Parallel Computing. The projects below are just some suggestions. Feel free to come up with your own ideas. Also use the course book to look for ideas.

- *Parallel Particle Simulation using far field forces*
  - Starting with: Ananth Grama, Vipin Kumar, Ahmed H. Sameh: Scalable Parallel Formulations of the Barnes-Hut Method for n-Body Simulations. Parallel Computing 24(5-6): 797-822 (1998)
  - Follow-up of mandatory assignment 2, possible by using a parallelization of Barnes-Hut algorithm, quad-trees should be known)
  - May not be accessible for analytical approaches
  - Maybe significant implementation requirements, 3 persons recommended (maybe 4 possible)
- *3SUM and r-SUM problem*
  - `http://en.wikipedia.org/wiki/3SUM`
  - Very easy problem. Isoefficiency analysis would be mandatory!
  - `http://www.cs.mcgill.ca/~jking/papers/3sumhard.pdf`
  - `http://cs.smith.edu/~orourke/TOPP/P11.html`
  - Isoefficiency analysis that depends on (n,p, and r)
  - max. 2 people
- *Dynamic Programming (DP)*
  - One example for parallelization of a DP problem is "Sequence Alignment" (but many other problems exist).
  - Start with reading DM813 Mandatory Assignment 1: Dynamic Programming and Sequence Alignment `http://www.imada.sdu.dk/~daniel/DM813/Assignments/mand1/`
  - Global / Local sequence alignment
  - Necessary: Reading Chapter 12 ("Dynamic Programming"), and using those methods

- *Dynamic Load Balancing / (Canonization of Graphs)*
  - Basically: Implement a simple dynamic work load balancer as described in the course book in Chapter 11 (see Fig 11.8).
  - In contrast to assume all processes work on one big instance, you can rather assume that many small tasks to be solved are given. Please note, that you should assume that the tasks have varying runtimes.
  - Please do not computing hours on Edison by running thousands of tasks that do nothing than just sleep.
  - While the implementation should work for any set of tasks (for example 1000 tasks that just sleep 0.1 to 10 seconds), it might be more fun to work on real tasks, for example Graph Canonization. For this material is provided in blackboard, the github for our own graph canon framework can be found here: `https://github.com/jakobandersen/graph_canon`.