Priority Lists, Scheduling

Read the book chapters first, then make sure you can answer the questions in the notes. Following that, work on some skills-check problems and exercises. Then take the online quizzes.

Reading 3.1, 3.2, 3.3
Skills Check 1, 2, 3, 4, 5, 7, 8, 9, 12, 13, 16
Exercises 4, 5, 6, 9, 11, 14, 17, 18, 19, 22, 23, 33, 35, 40, 42
Quiz Take the quiz[1]

3.1

- What is the context for a *machine-scheduling problem*?
- What are the main assumptions we will make in such problems?
- There are two sets of conditions on the tasks: an order-requirement digraph, and a priority list. Make sure you understand the difference between the two and why they are both important.
- What are the different goals we can aim for in such problems? What would those goals mean in some concrete example?
- In the list-processing algorithm, when do we call a task *ready*?
- What is the main step in the list-processing algorithm?
- Why do we have to keep processors idle some times?
- Show by example how the order of tasks in the priority list might affect the time to completion of the list-processing algorithm.
- Explain how the critical path relates to the determination of whether a specific schedule is optimal.
- In general, if we have tasks of total time W, and we have N processors available, and the critical path has length L, what two conditions can we impose on the time to completion of any schedule we can come up with? (page 80 covers this)
- What are all the factors that affect the list-processing algorithm?
- With a fixed priority list, we have three ways of trying to obtain an earlier completion time:
  - Reduce task times
  - Increase the number of processors
  - Loosen the constraints in the order-requirement digraph

  Study the examples in pages 81, 82 that demonstrate that in certain situations any one of these steps might in fact result in an increase in the completion time, rather than a decrease.

[1] https://moodle.hanover.edu/mod/quiz/view.php?id=4843

3.2

- The critical-path scheduling algorithm aims to produce a "good" priority list based on the requirements imposed by a critical path. Describe what the steps involved are, then carry those steps out in the examples you have seen so far.
- Does the critical-path scheduling algorithm produce a priority list that would give us the best possible scheduling?

3.3

- When do we say that we are dealing with the problem of scheduling *independent tasks*?
- Describe how the *decreasing-time-list algorithm* works.
- Does the decreasing-time-list algorithm guarantee optimal solutions?