Captain's Log: Introduction to Algorithm Analysis, Spring 2018

1: January 17 (Wednesday)

- go through the syllabus
- emphasize that the task of an algorithm is to *implement* a *specification*
- specify the *sorting* and the *selection* problems.

Cormen reading: Chapter 1.

2: January 19 (Friday)

Using the insertion sort method (**Cormen** Sections 2.1-2.2) as example, we

- illustrate how to design algorithms by the top-down approach
- show how to improve the implementation by applying the bottom-up technique
- give some simple complexity estimates.

Give a practice quiz using the *iClicker* system.

3: January 22 (Monday)

Embark on how to prove the *correctness* of algorithms:

- for iterative algorithms, cf. **Cormen** p.19 or **Howell** Sect 2.3, this requires a *loop invariant*, as illustrated by
  - the *fibonacci* example
  - a program finding the last occurrence of an element in an array
  - a program that uses binary search to *partition* a sorted array (the upcoming **Assignment 1**)

4: January 24 (Wednesday)

Continue the correctness of iterative algorithms: we

- outline how to handle *insertion sort* (**Cormen** p.19 or **Howell** Sect 2.3)
  where for the *inner* loop it is not obvious how to find an invariant that is strong enough
- construct, in the first graded quiz, an algorithm for integer division.

Assignment 1 due tomorrow night.

5: January 26 (Friday)

- Discuss how to prove (using *induction*) the correctness of recursive algorithms, cf. Section 2.2 in Howell
- Motivate and introduce the theory of asymptotic notation, cf. **Cormen** Chapter 3 and **Howell** Section 3.1.

6: January 29 (Monday)

- Introduce the "Big-O" symbol, and reason about its properties, cf. **Howell** Section 3.2
- Motivate the upcoming Assignment 2.

7: January 31 (Wednesday)
Finish asymptotic notation:

- introduce "Big-Omega" and "Big-Theta" (cf. **Howell** Section 3.3)
  while distinguishing between "best-case" and "worst-case" interpretation (cf. **Cormen** p.49)
- introduce "little-o" (and "little-omega"), cf. **Howell** 3.10 or **Cormen** p.50-51
- conclude with a quiz (with exit poll question).

Assignment 2 due tomorrow night.

8: February 2 (Friday)
Motivated by last quiz's exit poll, we

- recall the definition of little-o, and give typical examples of f,g with f in o(g)
- explain how little-o, big-O, big-Theta, big-Omega, little-omega are somewhat similar to <, <=, =, >=, >.

We next address (as in **Howell** 3.4-6) the complexity of iterative algorithms where we need to estimate sums;
the "rectangle" is (as summarized in **Howell**'s Theorem 3.28)

- trivially an upper approximation
- in our initial example, also a lower approximation, modulo a constant factor
- but in general, may not be a lower approximation (even modulo a constant factor) if the function "grows too fast"
- yet is a lower approximation (modulo a constant factor) if the function is *smooth*.

Along the way, briefly motivate the upcoming Assignment 3.

9: February 5 (Monday)
Mention various pitfalls in algorithm analysis:

- it may be that not all iterations can exhibit worst-case behavior
- numbers may become so big that they cannot realistically be handled in constant time
- what is and what is not a "barometer instruction".

Discuss various homework assignments:

- solutions to Assignment 1
- solutions to Assignment 2
- the upcoming Assignment 3.

Conclude with a quiz.
Last day for 100 % refund.

10: February 7 (Wednesday)
We address how to *solve* recurrences

- which show up when analyzing *divide and conquer* algorithms (such as *merge sort*), cf **Cormen** Section 2.3
- to *check* a proposed solution, the *substitution method* (**Cormen** Section 4.3) can be used:
  1. we illustrate by one kind of example which is rather straightforward

Further discuss Assignment 3, due tomorrow night.

11: February 9 (Friday)

Continue the substitution method:

1. recall the example from last time
2. for another kind of example, the presence of logarithms requires a careful phrasing of the induction hypothesis (two approaches are possible)
3. for a third kind of example, one needs some creativity to phrase the induction hypothesis
4. conclude with a quiz.

Finish with a brief introduction to the "Master Theorem".

12: February 12 (Monday)

Present a general recipe, the "Master Theorem" (cf, e.g, **Cormen** Sections 4.4 & 4.5) for solving recurrences:

- we illustrate by several examples
- we give a (very) informal proof
- we finish with a quiz.

Discuss the upcoming Assignment 4.

13: February 14 (Wednesday)

Elaborate on the Master Theorem:

- we show how recurrences can sometimes be twisted, by "change of variables" (cf. p.86 in Cormen), to fit the Master Theorem template
- we give an example where Howell's Theorem 3.32 (or Wikipedia) gives better precision than the Master Theorem from the slides (whereas Cormen's version gives nothing).

Assignment 4 due tomorrow night.

14: February 16 (Friday)

Cover the "Dutch National Flag" problem, cf. Section 2.4 in **Howell**:

- argue it is useful in a number of contexts
- develop a linear-time constant-space algorithm
- finish with a quiz.

15: February 19 (Monday)

Review session for the upcoming midterm exam:

- discuss solutions to Assignment 3
- discuss solutions to Assignment 4
- discuss the questions from the first midterm exam in 2017 (uploaded on Canvas) and relevant questions from the following exams

16: February 21 (Wednesday)

Exam I.

17: February 23 (Friday)

Introduce graphs, cf **Cormen** Appendix B.4, which

- are either directed or undirected

- may be acyclic and/or connected
- may be represented by an adjacency matrix or by adjacency lists, cf **Cormen** 22.1.

An important special case is a (rooted) tree, cf **Cormen** Appendix B.5.

18: February 26 (Monday)

Wrap up graphs:

- recall the notions of being acyclic and/or connected, cf. Question 1 in the upcoming Assignment 5
- compare the two possible representation choices, as illustrated by Questions 2 and 3 in the upcoming Assignment 5
- finish with a quiz.

Motivate *heaps*, by discussing how to implement a *priority queue*

19: February 28 (Wednesday)

Introduce the *heap* data structure, cf **Cormen** Chapter 6:

- define the heap property
- show how to maintain the heap property (**Cormen** 6.2)
- use heaps to implement priority queues (**Cormen** 6.5)
- show an efficient representation (**Cormen** 6.1)
- show how to convert an array into a heap (**Cormen** 6.3)
  (which can be done in linear time, as one can show by a recurrence and/or a summation)

Further discuss Assignment 5, due tomorrow night.

20: March 2 (Friday)

- Develop an efficient and in-place sorting algorithm, *heapsort* (**Cormen** 6.4).
- Explore the *Divide and Conquer* paradigm, already motivated by merge sort (**Cormen** Section 2.3) and with further examples:
  - we introduce *quicksort* (**Cormen** Section 7.1)
    where the key part to good performance (**Cormen** Section 7.2) is to properly design pivot selection
- Motivate Assignment 6.

21: March 5 (Monday)

- Discuss, and give back, graded Exam 1
- Discuss solutions to the recent Assignment 5
- Further discuss the upcoming Assignment 6

22: March 7 (Wednesday)

Elaborate on *sorting*:

- clarify some issues in the upcoming Assignment 6
- mention several reasons, listed on p.148 in **Cormen**, why one may consider sorting the *most* fundamental problem in the study of algorithms
- argue that any *comparison* sorting algorithm requires \Omega(lg(n!)) = \Omega (n lg (n)) comparisons in the worst case (**Cormen** Theorem 8.1).

Present another example of the divide-and-conquer paradigm:

- we address how to multiply n-bit integers (or polynomials of degree n, cf **Howell** Section 10.1)
- recall (much as in primary school) how two n-digit numbers can be multiplied by doing 4 multiplications of n/2-digit numbers (and then some addition)
- show that it will actually suffice with *3* multiplications (at the price of doing more additions but this doesn't affect asymptotic complexity)
- hence multiplication can be done in time O(n^b) where b is less than 2
(and may actually be arbitrarily close to 1, by generalizing the method).

Assignment 6 due tomorrow night.

23: March 9 (Friday)

we study the multiplication of n x n matrices (**Cormen** Section 4.2):

- the natural implementation of the definition takes time in O(n^3)
- a naive use of Divide & Conquer also takes time in O(n^3)
- a clever idea (Strassen) provides for a running time that is subcubic
- there is a continuing quest to further improve the running time
- finish with a quiz.

Briefly motivate the upcoming Assignment 7.

24: March 12 (Monday)

One more application of the Divide & Conquer paradigm:

- the selection problem which has a clever (deterministic) solution that always runs in linear time (**Cormen** Section 9.3).

Further discuss the upcoming Assignment 7.

25: March 14 (Wednesday)

Introduce *dynamic programming*, the topic of **Cormen** 15, which often allows an efficient implementation of exhaustive search:

- motivate by examples of how to avoid duplicated computations
- show how to apply it to the problem of giving optimal change, cf. **Howell** 12.1

26: March 16 (Friday)

Continue dynamic programming:

- apply it to the binary knapsack problem (**Howell** 12.4),
given as Exercise 16.2-2 in **Cormen** and with solution listed on page 50 here
- Motivate the upcoming Assignment 8

March 19-23

Spring break.

27: March 26 (Monday)

- Discuss in much detail the upcoming Assignment 8.
- A quiz on deterministic selection and (with an embarrassing typo) the binary knapsack problem.

28: March 28 (Wednesday)

- Present Floyd's all-pair shortest paths algorithm (**Cormen** 25.2, or **Howell** 12.3)
- Briefly discuss how to retrieve the optimal trading sequence in Assignment 8 which is due tomorrow night.

29: March 30 (Good Friday)
Wrap up dynamic programming:

- mention that a subpart of an optimal solution is not necessarily an optimal solution to the subproblem
- find an optimal solution for chained matrix multiplication (**Cormen** 15.2)
- finish with a quiz.

30: April 2 (Monday)
Review session for the upcoming midterm exam:

- look at questions from the exams in 2017 (uploaded on K-State Online)
- discuss solutions to the recent Assignments 6, 7 and 8

31: April 4 (Wednesday)
Exam II.
April 6 (Friday)
Class canceled (All-University Open House)
32: April 9 (Monday)
We embark on *greedy* algorithms, the topic of **Cormen** 16:

- motivate the concept
- show a greedy strategy for minimizing total waiting time for jobs with duration
- show a greedy strategy for the fractional knapsack problem (cf. **Cormen** Exercise 16.2-1).

Motivate the upcoming Assignment 9.
33: April 11 (Wednesday)

- Discuss, and hand back, graded Exam 2
- Further discuss Assignment 9, due tomorrow night.

34: April 13 (Friday)
Introduce depth-first traversal of graphs, directed as well as undirected (**Cormen** 22.3):

- show how to compute a spanning tree;
- show that we can view a graph as a rooted tree together with certain kinds of extra edges (back, forward, cross).

Motivate the problem of *topological sort*.
35: April 16 (Monday)
One may augment the depth-first algorithm in several ways:

- we show how to do topological sort, cf. **Cormen** 22.4

- we show how to find "articulation points", cf. **Cormen** Problem 22-2
- we briefly motivate the upcoming Assignment 10 which asks to find "bridges".

36: April 18 (Wednesday)

Embark on Network Flow, cf. **Cormen** 26.1-3:

- introduce key concepts, cf. **Cormen** 26.1
- show that a naive algorithm doesn't always generate the maximal flow
- hence motivate the Ford-Fulkerson method, cf. **Cormen** 26.2, which uses the concept of *residual networks*

Further discuss Assignment 10, due tomorrow night.

37: April 20 (Friday)

Continue Network Flow:

- recall and analyze the Ford-Fulkerson method; it
    - can often be improved by the Edmonds-Karp algorithm
    - can be used for Maximum Bipartite Matching, cf. **Cormen** 26.3
- finish with a quiz (also on depth-first search)

38: April 23 (Monday)

Finish Network Flow:

- introduce the notion of a *cut* in a flow network
- mention, and demonstrate by an example, that the maximum flow equals the minimum cut, cf. **Cormen**'s Theorem 26.6.

Embark on the *UnionFind* slides:

- mention various properties of relations: reflexive, symmetric, transitive, etc.

Discuss the upcoming Assignment 11

39: April 25 (Wednesday)

Introduce *union/find* structures (cf **Cormen** Chapter 21, Sections 1-3):

- they are useful for representing *equivalence classes*
- we can implement the operations so that we can prove they run in time O(lg(n))
- one can even implement union/find such that the "amortized" (average) cost per operation is
    - *"almost"* in O(1), or to be more specific:
    - the *inverse* of the Ackermann function introduced in Assignment 11, Q2.

Assignment 11 (last of the semester) due tomorrow night.

40: April 27 (Friday)

We address how to construct a *minimum-cost spanning tree*, cf. **Cormen** 23:

- present and illustrate Kruskal's algorithm, and analyze its complexity, cf. **Cormen** 23.2
- present and illustrate Prim's algorithm, and analyze its complexity, cf. **Cormen** 23.2
- briefly mention a general correctness proof, cf. **Cormen** Theorem 23.1.

41: April 30 (Monday)

Present Dijkstra's algorithm (cf. **Cormen** 24.3) for single-source shortest paths:

- illustrate it by the example in **Cormen** Fig 24.6
- analyze its running time
- discuss how to use it, and Floyd's algorithm, to find shortest paths for single/multiple sources and single/multiple destinations

Quiz on Kruskal's algorithm, and Union-Find structures.
Discuss Q1 from Assignment 11.

42: May 2 (Wednesday)

Wrap up Greedy Algorithms:

- schedule events with fixed start/end time, cf. **Cormen** 16.1
- introduce Huffman codes, cf. **Cormen** 16.3
- quiz on Prim's and Dijkstra's algorithms.

43: May 4 (Friday)

Prepare for final exam:

- discuss the recent Assignments 9 and 10 and 11(Q2).
- discuss the questions from the 2017 final exam

May 9 (Wednesday)

Final exam, 4:10-6:00pm

Torben Amtoft