Practice Questions and Reading Material

Note: The material from the books is provided as a supportive reference for the material covered during the lectures.

Introduction to Concepts of Algorithmic Design, Computing Running Times

Reading material: Chapter 0.1,0.2,0.3 from DPV, Chapters 1.2,3.1,3.2 from CLRS2
Practice questions:

- You may be provided an algorithm for computing the n^{th} Fibonacci number and then asked to compute its running time (think in terms of bit complexity). Alternatively, you may be asked to provide an efficient algorithm for computing Fibonacci numbers.
- Give a justification why improvements in hardware are typically not sufficient to make an algorithm with exponential running time reasonably tractable.
- We have 3 algorithms for the same problem where the first runs in exponential time, the second in logarithmic and the third in linear time as a function of the same input. Which algorithm do you prefer to use?
- You may be provided running times of different algorithms and asked to show which running time dominates asymptotically.
- Provide the definition of O-notation (or Ω, or Θ).
- You may be provided certain statements about asymptotic analysis of running times and asked to show if they are correct or not. For instance: "n^a dominates n^b if $a > b$".

Related exercises from DPV: 0.1, 0.2, 0.3
Related exercises from CLRS2: 3.1-1, 3.1-2, 3.1-3, 3.1-4, 3.2-4
Related problems from CLRS2: 1.1, 3-2, 3-3, 3-6

Number Theoretic Algorithms: Complexity of adding and multiplying 2 n-bit numbers, modulo arithmetic

Reading material: Chapters 1.1,1.2 from DPV, Chapter 31.1 from CLRS2
Practice questions:

- What is the "primality" problem and what is the "factoring" problem? Which one of the two is tractable? What are the advantages of the intractability of the other problem?
- How many digits do you need to represent a number N in base b?
- How much does the size of the representation of a number changes when we change bases?
- What is the running time of the addition operation for two n-bit numbers (think in terms of bit complexity)?
- What is the running time of the traditional multiplication operation taught in grade school for two n-bit numbers (think in terms of bit complexity)?
- Provide a recursive formula for the multiplication of two numbers.
- What is the complexity of modular addition and modular multiplication for two n-bit numbers?

Related exercises from DPV: 1.1, 1.2, 1.3, 1.4, 1.6, 1.7, 1.8, 1.9, 1.10, 1.31

RSA Cryptosystem, Fermat's Little Theorem and Modular Exponentiation

Reading material: Chapters 1.2,1.4 from DPV, Chapter 31.3, 31.6, 31.7 from CLRS2
Practice questions:

- Give a private key protocol for a cryptography application. Given an example of how it can be compromised.
- RSA is based on which basic property of modulo arithmetic?
- Describe the steps of the RSA protocol. The security of the protocol is based on which assumption?
- What are the basic operations that need to be performed according to the RSA protocol and what is their running time?
- You may be provided an example message and asked to describe the operations of the RSA protocol on it.
- What does Fermat's Little Theorem specify? Prove it.
- What is the importance of Fermat's little theorem in the RSA protocol?
- How can we efficiently perform modular exponentiation? What is the running time of the approach?

Related exercises from DPV: 1.17, 1.27, 1.28, 1.35, 1.39, 1.42, 1.43, 1.44

Greatest Common Divisor algorithms: Euclid's test, Modulo Multiplicative Inverse

Reading material: Chapters 1.2 from DPV, Chapter 31.2 from CLRS2
Practice questions:

- What does Euclid's rule specify? Prove it.
- What is Euclid's algorithm for finding the greatest common divisor? What is its running time and why?
- Show that if d divides both a and b, and $d = a \cdot x + b \cdot y$ for some integers x and y, then $d = gcd(a, b)$.
- What is the extended Euclid's algorithm for finding the greatest common divisor d of two numbers a and b, as well as numbers x and y, so that $d = a \cdot x + b \cdot y$? Prove its correctness (you will need to prove Euclid's algorithm first and then the extension).
- When does the multiplicative inverse of a number x exists modulo N and why?
- How can you compute the multiplicative inverse modulo N for two relative prime numbers? What is the running time of the corresponding algorithm?

Related exercises from DPV: 1.18, 1.19, 1.20, 1.21, 1.22, 1.23, 1.24, 1.33, 1.37

Primality Testing and Universal Hashing

Reading material: Chapters 1.3,1.5 from DPV, Chapters 31.8, 11.2, 11.3 from CLRS2
Practice questions:

- Describe a test for evaluating whether a number is prime. What is the probability of this test returning the correct answer and why? Can you increase the probability of success for this test?
- What does Lagrange's Prime Number Theorem specify and why is it helpful for primality testing?
- What properties should a good hash function provide?
- What is the property of universal hashing families of functions? Provide a universal hashing family of functions for an example problem and prove the corresponding property.

Related exercises from DPV: 1.29, 1.34, 1.35

Divide and Conquer Algorithms and Recurrence Functions, Mergesort

Reading material: Chapter 2.1-2.2-2.3 from DPV, Chapters 2.3,4.1,4.2,4.3 from CLRS2
Practice questions:

- What are the basic principles of divide-and-conquer algorithms?
- Describe an approach for multiplication of two n-bit numbers that has a better running time than $O(n^2)$. Prove its running time.
- What does the Master theorem specify? Prove it.
- You may be provided a divide-and-conquer algorithm and asked to argue about its running time by using the Master theorem.
- How does mergesort work, what is its running time and why? How does the iterative version of mergesort work?

Related exercises from DPV: 2.3, 2.4, 2.5, 2.12, 2.13, 2.14, 2.19, 2.25, 2.26, 2.31
Related exercises from CLRS2: 2.3-1, 2.3-3, 4.3-1, 4.3-2, 4.3-3, 4.3-4
Related problems from CLRS2: 4-1, 4-2, 4-3, 4-4

Quicksort, Lower bounds for comparison-based sorting

Reading material: Chapter 7.1, 7.2, 7.3, 7.4, 8.1 from CLRS2, Chapter 2.3 from DPV
Practice questions:

- What are comparison sorting algorithms? What is the lower limit for the running time of comparison sorting algorithms?
- How does quick-sort work? When does the worst-case running time arise? When does the best-case running time arise?
- Provide a rigorous proof for the worst-case performance of quick-sort.
- Provide a rigorous proof for the expected running time of quick-sort.

Related exercises from CLRS2: 7.1-4, 7.2-2, 7.2-3, 7.2-4, 7.2-5, 7.3-1, 7.3-2, 7.4-1, 7.4-2, 7.4-3, 7.4-4, 7.4-5, 8.1-1, 8.1-3
Related problems from CLRS2: 7-3, 7-4, 7-5
Related exercises from DPV: 2.17, 2.18, 2.24

Computing Medians and Other Order Statistics, Linear-time Sorting Solutions

Reading material: Chapters 2.4 from DPV, Chapters 8.2, 8.3, 8.4, 9.1, 9.2, 9.3 from CLRS2
Practice questions:

- How can we efficiently compute the median of a set of numbers? What is the running time of this solution?
- What is the main idea behind counting sort? Provide a description of the algorithm and argue about its running time.
- What is the main idea behind radix sort? Provide a description of the algorithm and argue about its running time.
- Prove the correctness of radix sort.
- What is the main idea behind bucket sort? Provide a description of the algorithm. Prove its running time.

Related exercises from DPV: 2.15, 2.16, 2.17, 2.20, 2.21, 2.22, 2.23
Related exercises from CLRS2: 8.2-2, 8.2-4, 8.3-2, 8.3-3, 8.3-4, 8.4-2, 8.4-3, 9.1-1, 9.3-5, 9-3.7, 9.3-8, 9.3-9
Related problems from CLRS2: 8-2, 8-3, 8-6, 9-1, 9-2, 9-3

Greedy Algorithms: Huffman encoding

Reading material: Chapters 5.2 from DPV, Chapter 16.2, 16.3 from CLRS2
Practice questions:

- What is the main principle behind greedy algorithms?
- Why does the greedy algorithm work for the coin changing problem given the US coin system?
- What is the idea in Huffman encoding in order to achieve data compression? What is the prefix-free property?
- Provide the Huffman encoding algorithm and argue its running time.
- Prove the greedy choice property for the Huffman encoding algorithm (first lemma).
- Prove the optimal substructure property for the Huffman encoding algorithm (second lemma).

Related exercises from DPV: 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.29, 5.30, 5.31
Related exercises from CLRS2: 16.2-4, 16.2-5, 16.2-7, 16.3-1, 16-3.2, 16.3-3, 16.3-4, 16.3-5, 16.3-6
Related problems from CLRS2: 16.1