Module 7: Discrete Mathematics: Overview of Tutorial Assignments

Note. This table includes the tutorial assignments from the book as well as some more (on the next pages). Last updated: January 31, 2016.

week	no	date & time	preparatory assignments	tutorial assignments (book)	tutorial assignments (next pages)
1	1	wed 8+9	§4.3: 4,8,14 §4.4: 1ac,2	§4.3: 10,19,27 §4.4: 10,12,17	1,2,3,4
	2	fri 6+7	§4.5: 8a,15 §11.1: 12a §11.2: 5,6,10	§4.5: 5,20,26 §11.2: 11,12 §11.3: 13,18,31,35	5
2	3	tue 6+7	§13.1: 2,3	§13.1: 1,5	6,7,8,9
	4	thu 3+4	§12.1: 7 §13.2: 1,2,6	§13.2: 3,8,9	10,11
3	5	tue 6+7	§13.3: 1,3	§13.3: 2,5	12,13
	6	fri 6+7	§11.4: 2,10,12,14	§11.4: 21,22,28,29	—
4	7	mon 6+7	§10.1: 2,3 §10.2: 1bc, 9 §10.3: 1	§10.2: 1de,10,23 §10.3: 5,6,8,12	—
	8	fri 6+7	§9.1: 5 §9.2: 1abc,2abc,5	§10.2: 33 §10.6: 5 §9.2: 1bef,6,11 §10.4: 1a	14
8	9	tue 6+7	§14.3: 31,32	§14.3:6 §16.3: 9,13	Document RSA: 1-5

Table 1: Refer to 5th ed. of Grimaldi. Read “§4.3: 4” as Chapter 4, Exercises 3: No. 4.

Tutorial Assignments & Questions

1. (Related to the proof of the lecture that $\gcd(a, b) = \min\{xa + yb > 0 \mid x, y \in \mathbb{Z}\}$.) Let $a, b \in \mathbb{Z}$ be integers, not both equal to 0. Argue formally that $\{ax + by > 0 \mid x, y \in \mathbb{Z}\ \} \neq \emptyset$.

2. (Related to the proof of the lecture that the number of iterations of the Euclidean algorithm is $O(\log b)$.) Recall the Fibonacci numbers, $F_0 = 0$, $F_1 = 1$, and for all $n \geq 2$, $F_n = F_{n-1} + F_{n-2}$. Show by mathematical induction that
$$\text{for all } n \geq 3, \quad F_n > \alpha^{n-2}\,.$$
Here, $\alpha = \big((1 + \sqrt{5})/2\big)$ is the golden ratio (hint: recall that $\alpha^2 = \alpha + 1$).

3. Prove Theorem 4.8: If a, b, c are positive integer numbers, then diophantine equation $ax + by = c$ has an integer solution (x, y) if and only if $\gcd(a, b)|c$.

4. Can you do exercise §4.3.27 (computing all divisors of a given integer n) in Python?

5. Can you do exercise §4.5.20 (computing all prime divisors of a given integer n) in Python?

6. (Correctness Fleury's algorithm.) Recall Fleury's algorithm to compute an Eulerian tour in an Eulerian graph $G = (V, E)$ from Lecture 2: Pick any starting vertex $v_0 \in V$, then iterate: leave the current vertex v on an edge e that is not a bridge (if possible), and remove e from G. Argue why this algorithm works correctly. Also give an upper bound on its computation time.

7. (Eulerization, a.k.a. Chinese Postman Problem.) Suppose we are given a (multi) graph $G = (V, E)$ that is not necessarily Eulerian. Suggest a method to make the graph Eulerian, by adding as few edges as possible, where multiple edges are allowed. (Hint: doubling all edges creates a graph that is definitely Eulerian.) Argue why your answer is correct.

8. Show by means of a simple example that Dijkstra's algorithm may fail to compute correct shortest path lengths for a digraph $G = (V, A)$ if arc lengths are allowed be negative.

9. Consider a directed graph where each edge $e \in E$ has a reliability $0 \leq r_e \leq 1$. That is, r_e is the probability that an arc will fail. Assume all r_e are independent. Suggest an algorithm to compute a most reliable (s, t)-path for given $s, t \in V$.

10. Consider an undirected graph $G = (V, E)$ with edge weights $c_e \geq 0$, $e \in E$. Prove the following claim (the "path condition"). Given an arbitrary spanning tree $T \subseteq E$ for G, denote by $P_T(v, w)$ the (unique) path from v to w in T. Then T is a minimum spanning tree if and only if $c_e \leq c_f$ for all edges $f = \{v, w\} \in E \setminus T$ and all edges $e \in P_T(v, w)$. (Hint: use the cut condition that was proved in the lecture.)

11. Let T be a tree on $n \geq 2$ nodes. Prove that the number of pendant nodes (nodes v with $d(v) = 1$) equals

$$2 + \sum_{v:d(v)\geq 3} (d(v) - 2)\,.$$

12. Consider a capacitated network, that is, a directed graph $G = (V, A)$, two designated nodes $s, t \in V$ (source and target), and integer arc capacities $u_(i, j) \geq 0$, $(i, j) \in A$. We wish to send the maximum possible flow from s to t. Now assume that each node $v \in V$ also has a maximum integer capacity $c_v \geq 0$, $v \in V$, so that the flow that can pass through node $v \in V$ is limited to at most c_v. Suggest a transformation of the network so that this problem can be solved by just one application of the Edmonds-Karp algorithm.

13. Consider the feasible flow problem as discussed in Lecture 5. That is, we are given capacitated network $G = (V, A)$ with integer arc capacities $u_{(i,j)} \geq 0$, $(i, j) \in A$, and for each node $i \in V$ we are given an integer value $b(i)$ which is the required outflow (excess) at node i. More precisely, if $x = (x_{(i,j)})_{(i,j)\in A}$ is a flow vector, then the flow out of node i minus the flow into i must be equal to $b(i)$,

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b(i)\,.$$

Here, $b(i)$ can be both positive or negative (that is, there are both producing and consuming nodes). Now let us suppose that, instead of ordinary flow capacity constraints $0 \leq x_{ij} \leq u_{ij}$, we have flow capacity constraints of the form $\ell_{ij} \leq x_{ij} \leq u_{ij}$ for some nonnegative integer lower bound ℓ_{ij} on the arc flows.

Show that this feasibility problem can be solved by application of a standard maximum (s, t)-flow algorithm in an accordingly transformed network (with capacity constraints $0 \leq x_{ij} \leq u_{ij}$).

(Hint: First get rid of the lower bounds on arc flows by modifying the balance constraints and arc capacities accordingly. Then get rid of all nodes with balance constraints $\neq 0$ by introducing (new) source and target nodes s and t. Finally, argue how one maximum flow computation in the transformed network can be used to solve the feasible flow problem.)

14. Recall algorithm `Power(a,n)` from Lecture 7, a fast implementation of exponentiation, i.e., for computing a^n. Here we show that for any n, this algorithm does the job with only $\mathrm{O}(\log n)$ multiplications.

input : a, n with $n \in \mathbb{Z}^+$
output: a^n
if $(n == 0)$ **then return** 1;
else
  if$(n$ even$)$ **return** `Power`$(a^2, n/2)$; // $a^n = (a^2)^{n/2}$
  else return $a \cdot$`Power`$(a^2, (n-1)/2)$; // $a^n = a \cdot (a^2)^{(n-1)/2}$
end

Algorithm 1: `Power(a,n)`

If we let $t(n)$ be the (worst case) number of multiplications of `Power(a,n)`, show by induction that

$$t(n) \leq 2 \log_2 n + 2 \,.$$