Solutions: Algorithms & Data Structures

see separate pdf's.

Solutions: Discrete Mathematics

4. (a) By the Bézout identity and since $\gcd(a, b) = 1$, there exist $x, y \in \mathbb{Z}$ such that $ax + by = 1$, and thus $acx + bcy = c$. By Theorem 4.3, since $a|(acx)$ and $a|bcy$, it follows that $a|c$.
Here is an alternative argument that does not refer to Theorem 4.3: Since $a|bc$, there exists $q \in \mathbb{Z}$ so that $aq = bc$, and by substituting bc in the above equation, we get $acx + aqy = c$, or equivalently $a(cx + qy) = c$. We have $(cx + qy) \in \mathbb{Z}$ and in turn, $a|c$.

   (b) The statement is correct. Assume that $g := \gcd(a, b)|c$, which means that $c = kg$ for some $k \in \mathbb{Z}$. As $g = \min\{xa+yb > 0 \mid x, y \in \mathbb{Z}\}$, we know that there exist $x, y \in \mathbb{Z}$ with $g = xa+yb$, hence $c = kg = (kx)a + (ky)b$, and $s := kx \in \mathbb{Z}$, $t := ky \in \mathbb{Z}$.

5. Let $E(s) \subseteq \delta(s)$ be the edges in $\delta(s)$ that have minimal weight (among the edges in $\delta(s)$). Since $d_e > 0$ for all $e \in E$, for any edge $e = \{s, v\} \in E(s)$, there can be no shorter (s, v)-path than $\{s, v\}$ itself. Hence $E(s) \subseteq D(s)$. We claim that $E(s) \cap T \neq \emptyset$ Assuming that $E(s) \cap T = \emptyset$, pick any $e = \{s, v\} \in E(s)$, and consider the (unique) (s, v)-path $P_T(s, v)$ in T. Of course $P_T(s, v)$ must contain some edge $f \in \delta(s)$, but $d_f > d_e$, because $E(s) \cap T = \emptyset$. This would be contradicting the path condition for minimum spanning trees, however. Therefore $T \cap E(s) \neq \emptyset$, and also $T \cap D(s) \neq \emptyset$.

6. Since the maximum flow in the network has value $\mathrm{val}(f) > k$, by the strong duality theorem it follows that the minimum (s, t)-cut (S, T) in the network has a capacity of $c(S, T) = \mathrm{val}(f) > k$. Since each edge has unit-capacity, this implies that the cut (S, T) consists of strictly more than k many (forward) edges. We note that in the network $G' = (V, E', c)$ obtained by removing an edge e from a minimum cut (S, T) (i.e., $E' = E \setminus \{e\}$ for some $e \in (S, T)$), the cut (S, T) is again a minimum cut but has its capacity decreased by exactly one unit. This follows from the fact that any (s, t)-cut in G that did not contain e retains its capacity in G', and every (s, t)-cut in G that contained e will have a capacity reduced by exactly one unit in G'. This suggests the following algorithm:

   (a) Identify a minimum cut (S, T): Compute the residual graph G_f for G with respect to f. Run the BFS algorithm from s in G_f to obtain the set S of vertices reachable from s. Set $T := V \setminus S$.

   (b) Remove k arbitrary (forward) edges from (S, T); let G^* be the obtained graph.

   We already argued that there exist (more than) k forward edges in (S, T) to be removed in the second step, and that the resulting maximum flow f^* in G^* will have value $\mathrm{val}(f^*) = \mathrm{val}(f) - k$. This is as small as possible since any (s, t)-cut has a capacity of at least $\mathrm{val}(f)$ in G and each edge removal can only decrease the capacity of any cut by at most one unit.

   Running time: Computing the residual graph G_f requires $O(m)$-time, and BFS $O(n + m)$-time. Removing the k edges requires $O(k)$ time. Since $k < m$, we have in total $O(n + m)$ time.

7. (a) The characteristic polynomial of the corresponding homogeneous recurrence relation is $x^2 - 10x + 21 = (x - 3)(x - 7)$. The roots are $x_1 = 3$ and $x_2 = 7$. Hence the general solution to the homogeneous recurrence relation is

$$a_n^{(h)} = c_1 3^n + c_2 7^n \, .$$

   We use as the particular solution to the inhomogeneous recurrence relation

$$a_n^{(p)} = An3^n \, ,$$

   (because $A3^n$ would not be linearly independent). Plugging this into the recurrence relation gives $An3^n - 10A(n-1)3^{n-1} + 21A(n-2)3^{n-2} = 60 \cdot 3^n$ for all n, so $An3^n(1 - \frac{10}{3} + \frac{7}{3}) +$

$A3^n(\frac{10}{3} - \frac{14}{3}) = 60 \cdot 3^n$, hence $A = -45$. Therefore, the general solution to the inhomogeneous recurrence relation is

$$a_n = c_1 3^n + c_2 7^n - 45n3^n \,.$$

Now we have $a_0 = 2 = c_1 + c_2$, and $a_1 = -5 = 3c_1 + 7c_2 - 135$. This yields $c_1 = -29$ and $c_2 = 31$, and the solution equals

$$a_n = -29 \cdot 3^n + 31 \cdot 7^n - 45n3^n \,.$$

(b) Let a_n^k be the number of strings (with the required properties) that end on letter k, then

$$a_n = a_n^0 + a_n^1 + a_n^2 \,.$$

Now we see that $a_n^0 = a_{n-1}$, $a_n^1 = a_{n-1}^1 + a_{n-1}^2$, and $a_n^2 = a_{n-1}^1 + a_{n-1}^2$. That yields

$$a_n = a_{n-1} + (a_{n-1} - a_{n-1}^0) + (a_{n-1} - a_{n-1}^0) = 3a_{n-1} - 2a_{n-2} \,.$$

Finally, $a_1 = 3^1 = 3$, $a_2 = 3^2 - 2(\text{for } 01 \text{ and } 02) = 7$, $a_3 = 3^3 - 3 \cdot 4(\text{for } 01X \text{ and } X01 \text{ and } 02X \text{ and } X02) = 15 (= 3 \cdot 7 - 2 \cdot 3)$.

8. $n = 91$ implies $p = 7$ and $q = 13$ (or vice-versa), and $r = (p-1)(q-1) = 6 \cdot 12 = 72$. Eve wants to compute the multiplicative inverse of e (in $\mathbb{Z}_{72}$). By using the extended Eucl. algorithm:

$$\begin{bmatrix} 72 & 1 & 0 \\ 11 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 6 & 1 & -6 \\ 11 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 6 & 1 & -6 \\ 5 & -1 & 7 \end{bmatrix} \sim \begin{bmatrix} 1 & 2 & -13 \\ 5 & -1 & 7 \end{bmatrix}$$

Therefore $1 = 2 \cdot 72 - 13 \cdot 11$, and $d = -13 \pmod{72} = 59 \pmod{72}$. Next she uses modular exponentiation to compute $M = 3^{59} \pmod{91} = 61$. In detail, $(59)_2 = 111011$ and so $3^{59} = 3^{2^5} 3^{2^4} 3^{2^3} 3^{2^1} 3^{2^0}$.

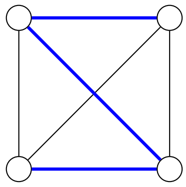
Computing $\pmod{91}$ we have

$$\begin{aligned} 3^{2^0} &= 3, \\ 3^{2^1} &= 9, \\ 3^{2^2} &= 81 = -10, \\ 3^{2^3} &= 100 = 9, \\ 3^{2^4} &= 81 = -10, \\ \text{and } 3^{2^5} &= 9. \end{aligned}$$

This gives $3^{59} \pmod{91} = 9 \cdot (-10) \cdot 9 \cdot 9 \cdot 3 \pmod{91} = 61$.

9. (Although this question does not require motivating the answers, since this is a sample test, we do include a proof for completeness.)

(a) False. Consider K_4 (that is, the complete graph on 4 vertices), with all edges with equal weights, then there are two minimum spanning trees which are the complement of each other, hence disjoint.



(b) False. Consider graph with three nodes $\{s, v, t\}$ and edges $(s, v), (v, t)$ with capacities $c(s, v) = 1$ and $c(v, t) = 2$. Then the only maximum flow falsifies the claim on edge (v, t).

(c) False. Consider graph with four nodes $\{s, u, v, t\}$ and edges $(s, u), (s, v), (u, t), (v, t)$ with weights $w(s, u) = 1$ and $w(u, t) = 6$, $w(s, v) = 3$ and $w(v, t) = 4$, then there are two shortest (s, t)-paths of length 7.

(d) True. For a proof, please refer to the tutorial sessions. Note that arguing via Kruskals's algorithm is not sufficient, even though Kruskal's algorithm computes a unique spanning tree. But potentially, there could be minimum spanning tres that can't be computed by Kruskal's algorithm... The actual proof is: Assume there exist two different MST's T_1 and T_2, then there exists at least one edge $e = \{u, v\} \in T_1 \setminus T_2$. As in T_2, u and v are also connected by a unique path $P_{T_2}(u, v)$, we know by the path condition (for T_2), that $w_f \leq w_e$ for all edges $f \in P_{T_2}(u, v)$, and since all weights are different, $w_f < w_e$ for all edges $f \in P_{T_2}(u, v)$. But now we get a contradiction to T_1 being a minimum spanning tree, because in the cut induced by $T_1 - e$, there exists at least one edge $f \in P_{T_2}(u, v)$, which is strictly cheaper than e, contradictiong the cut condition for T_1.

(e) True. G is disconnected and therefore non-Eulerian, in turn at least one edge needs to be added in order to obtain an Eulerian graph. Adding one or two (non-parallel to keep the graph simple) edges will result in at least two odd-degree vertices thus the graph cannot be Eulerian. Since at least one of the two components is not complete, there exist vertices v, u in the same component such that $(u, v) \notin E$. Let w be an arbitrary vertex in the component different to the one containing u and v. Adding the cycle consisting of edges $(u, v), (v, w)$ and (w, u) maintains the parity of all vertex-degrees, connects the two components and the resulting graph is simple.

(f) False. Matching M is not stable, because (c, B) is an unstable pair with respect to M: both c and B would prefer each-other over their partner in M.