CS 280 - Homework 5
Solutions

Problem 1. (10)

a. Evaluate $7^{1001} \bmod 11$.

$$\begin{aligned} 7^{1001} \bmod 11 &= \left(7^{10}\right)^{100} \bmod 11 \\ &= 1^{100} \bmod 11 \\ &= 1 \bmod 11 \\ &= 1 \end{aligned}$$

b. Calculate $\varphi(120)$ and $\varphi(384)$. Use the formula

$$\varphi(n) = n \cdot \prod_p \left(1 - \frac{1}{p}\right)$$

where p ranges over unique primes that divide n.

$$\begin{aligned} \varphi(120) &= \varphi(2^3 \cdot 3^1 \cdot 5^1) \\ &= \varphi(8) \cdot \varphi(3) \cdot \varphi(5) \\ &= 4 \cdot 2 \cdot 4 \\ &= 32 \end{aligned}$$

$$\begin{aligned} \varphi(384) &= \varphi(2^7 \cdot 3^1) \\ &= 384 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{3}\right) \\ &= \frac{384}{3} \\ &= 128 \end{aligned}$$

c. Show that $(n^{13} - n)$ is divisible by 2, 3, 5, 7, and 13, $\forall\ n \in \mathbb{Z}$. A lot of people went through a lot of unnecessary modular computation to find the solution to this problem. A simple application of Fermat's rule makes the problem much simpler. Fermat's little theorem (FLT) states: $a^{p-1} = 1 \pmod p$ when $a \not| p$.

We have to prove that $n^{13} - n$ is divisible by 2, 3, 5, 7, 13.

For 2: $(n^{13} - n) = n \cdot (n^{12} - 1)$ Notice that if n is not divisible by 2, then $n^{2-1} = 1 \pmod 2$ by FLT and therefore since n congruent to $1 \pmod 2$, and n is relatively prime to 2, n^{12} is congruent to 1 mod 2 and $n^{12} - 1$ is congruent to $0 \pmod 2$. Hence the term is divisible by 2.

For 3: We see that n^2 is congruent to $1 \pmod 3$ by FLT. Therefore $\left(n^2\right)^6$ is also congruent to $1 \pmod 3$, and therefore $n^{12} - 1$ is a multiple of 3.

For 5, 7 and 13, notice that n^4, n^6 and n^{12} are all congruent to 1 modulo 5, 7 and 13 respectively. Therefore $\left(n^4\right)^3$, $\left(n^6\right)^2$ and $\left(n^{12}\right)^1$ (which are all equal to n^{12}) are congruent to 1 mod 5, 7 and 13 respectively. Therefore $n^{12} - 1$ is divisible by all of 5, 7, and 13. Hence proved.

d. Show that if $n \geq 2$ and p is a prime s.t. $p \mid n$ but $p^2 \nmid n$, then $p^{\varphi(n)+1} \equiv p \bmod n$. Let $k = \frac{n}{p}$ for the purposes of this problem. Notice that k is necessarily coprime with respect to p, because if k was a multiple of p, then this contradicts the claim that p^2 does not divide n (since this implies that another multiple of p is built into k). Therefore if we assume truth of the statement, then:

$$\begin{aligned} p^{\varphi(n)+1} &\equiv p \bmod n \\ p^{\varphi(n)} \cdot p &\equiv p \bmod n \\ p^{\varphi(n)} &\equiv 1 \bmod k \qquad \text{[we divided both sides by p]} \end{aligned}$$

Notice that $n = k \cdot p$, so $p^{\varphi(n)} = p^{\varphi(k \cdot p)}$. Therefore $p^{\varphi(k*p)} \equiv 1 \bmod k$. Thus $\left(p^{\varphi(k)}\right)^p \equiv 1 \bmod k$, or $\left(p^{k-1}\right)^p \equiv 1 \bmod k$. Since $p^{k-1} \equiv 1 \bmod k$, it stands to reason that $\left(p^{k-1}\right)^p \equiv 1 \bmod k$. Hence our initial assumption that the terms were equal holds.

e. Explain carefully what happens if in the RSA cryptosystem we were to allow larger block sizes (relative to the two primes used) so that some of the blocks could fail to be relatively prime to $n = pq$. If the block size is big enough to encode n, then encoding n and 0 result in the same number, namely 0. There are many different examples that would meet this criterion.

Problem 2. (6)

a. We are given $p = 17$, $q = 43$, $e = 29$, $n = pq = 731$, and $E =$ `290 369 203 405 033 511 584 612 213`.

$$\varphi(n) = \varphi(731) = \varphi(17) \cdot \varphi(43) = 16 \cdot 42 = 672$$

We know that $29 \cdot d = 1 \bmod 672$, so using the fact that $[29]^{-1} = -139$, we find $d = -139 \bmod 672 = 533$. We now find the first M, using the formula $M = E^d \pmod n$:

$$\begin{aligned} M_1 &= 290^{533} \pmod{731} \\ &= 290^{13 \cdot 41} \pmod{731} \\ &= 29^{13 \cdot 41} 10^{13 \cdot 41} \pmod{731} \\ &= 405^{41} 640^{41} \pmod{731} \\ &= 700 \cdot 533 \pmod{731} \\ &= 290 \pmod{731} \end{aligned}$$

Since $290 \bmod 26 = 4 \Rightarrow$ `E` and $(290 - 4)/26 = 11 \Rightarrow$ `L`, we see that the first two letters of the plaintext are `LE`.

The rest of the blocks are decrypted in the same manner, yielding the plaintext `LE TT HE BU YE RB EW AR EX`.

b. We are given $p = 113$, $q = 157$, $e = 12707$, and $n = 113 \cdot 157 = 17741$. `THISISMAD` is encoded by breaking it into 3-letter chunks:

$$\begin{aligned} \texttt{THI} &\to 19 \cdot 26^2 + 7 \cdot 26^1 + 8 \cdot 26^0 = 13034 \\ \texttt{SIS} &\to 18 \cdot 26^2 + 8 \cdot 26^1 + 18 \cdot 26^0 = 12394 \\ \texttt{MAD} &\to 12 \cdot 26^2 + 0 \cdot 26^1 + 3 \cdot 26^0 = 8115 \end{aligned}$$

So the ciphertext is:

$$\begin{aligned} 13034^{12707} \pmod{17741} &= \left(2^{97}\right)^{131} \cdot \left(7^{97}\right)^{131} \cdot \left(7^{97}\right)^{131} \cdot \left(7^{97}\right)^{131} \cdot \left(19^{97}\right)^{131} \pmod{17741} \\ &= 1635 \cdot 155453 \cdot 10078 \pmod{17741} \\ &= 4453 \end{aligned}$$

The rest of the blocks are encrypted in the same manner, yielding the ciphertext `04453 14649 09416`.

c. This problem is solved in the same way as (a), yielding `DON TWO RRY BEH APP YXX`.

Problem 3. (6)

a. Since $m = pq$ and $n = (p-1)(q-1)$:

$$\begin{aligned} m - n + 1 &= pq - (p-1)(q-1) + 1 \\ &= pq - (pq - p - q + 1) + 1 \\ &= p + q \end{aligned}$$

b. Since $q = \frac{m}{p}$ and $p + q = m - n + 1$, we can substitute $\frac{m}{p}$ for q, giving:

$$\begin{aligned} p + \frac{m}{p} &= m - n + 1 \\ p^2 + m &= mp - np + p \\ p^2 + m &= (m - n + 1)p \\ p^2 + (n - m - 1)p + m &= 0 \end{aligned}$$

c. By applying the quadratic formula to the equation from (b), we get

$$p = \frac{(m - n + 1) \pm \sqrt{(m - n + 1)^2 - 4m}}{2}$$

We choose the plus sign to get the desired equation.

$$\begin{aligned} q &= (m - n + 1) - p \\ &= \frac{2(m - n + 1)}{2} - \frac{(m - n + 1) + \sqrt{(m - n + 1)^2 - 4m}}{2} \\ &= \frac{(m - n + 1) - \sqrt{(m - n + 1)^2 - 4m}}{2} \end{aligned}$$

d. We are given that $m = pq = 5336063$ and $n = (p-1)(q-1) = 5331408$; we use the values in the formulas from (c) to obtain $p = 2617$ and $q = 2039$.

Problem 4. (4)

a. Evelyn's goal is to recover the plaintext of the message, which is x. She knows both e_a and e_z and that they are coprime, so $\gcd(e_a, e_z) = 1$. Thus there exist s, t with $se_a + te_z = 1$, and they can be easily calculated with the extended Euclidean algorithm. Evelyn knows the two ciphertexts, x^{e_a} and x^{e_z}, and thus can easily recover the plaintext:

$$\left(x^{e_a}\right)^s \left(x^{e_z}\right)^t = x^{se_a + te_z} \bmod m = x^1 = x$$

Since this is the plaintext, we are done.

b. For $m = 4171$, $e_a = 47$, and $e_z = 101$, find the numerical plaintext from which the ciphertexts `2467` and `2664` were computed.

The extended Euclidean algorithm yields the final matrix row $\begin{bmatrix}43 & -20 & 1\end{bmatrix}$; from this we define $s = 43$, $t = -20$ such that $43 \cdot 47 + (-20) \cdot 101 = 1$. Hence:

$$\begin{aligned} x &= 2467^s \cdot 2664^t \pmod{4171} \\ &= 2467^{43} \cdot 2664^{-20} \pmod{4171} \\ &= 2467^{43} \cdot \left(2664^{-1}\right)^{20} \pmod{4171} \end{aligned}$$

With the Euclidean algorithm, we find $2664^{-1} \bmod 4171 = 2300$.

$$\begin{aligned} 2467^{43} &= 1306 \pmod{4171} \\ 2300^{20} &= 3137 \pmod{4171} \end{aligned}$$

Hence $x = 1306 \cdot 3137 \bmod 4171 = 1000$.

Problem 5. (8) Let G be a group under multiplication and define an action of G (as a group) on G (as a set) by $g.a = gag^{-1}$.

a. To show that $\left(a \sim b \iff \exists\, g \in G \text{ s.t. } g.a = b\right)$ is an equivalence relation on G, we must show that it is reflexive, symmetric, and transitive.
Is $a \sim a$? $1a1^{-1} = 1a1 = a$, so $a \sim a$.
Does $a \sim b \implies b \sim a$? If $a \sim b$, then there exists g s.t. $gag^{-1} = b$. Take $h = g^{-1}$. Then $h.b = h(gag^{-1})h^{-1} = g^{-1}gag^{-1}g = 1a1 = a$, and $b \sim a$.
Does $a \sim b \wedge b \sim c \implies a \sim c$? If $a \sim b$ and $b \sim a$, then there exist g and h s.t. $gag^{-1} = b$ and $hbh^{-1} = c$. Take $i = hg$. Then $i.a = (hg)a(hg)^{-1} = hgag^{-1}h^{-1} = h(g.a)h^{-1} = hbh^{-1} = h.b = c$, and $a \sim c$.
Thus $\sim$ is an equivalence relation on G.

b. We have $N(x) = \left\{g \in G \,\middle|\, gxg^1 = x\right\}$; to show that $N(x)$ is a subgroup of G, we must show that it contains the identity element, is closed under multiplication, and is closed under inversion.
Is $1 \in N(x)$? $1x1^{-1} = 1x1 = x$, so $1 \in N(x)$.
Does $g, h \in N(x) \implies gh \in N(x)$? $(gh)x(gh)^{-1} = ghxh^{-1}g^{-1} = gxg^{-1} = x$, so $gh \in N(x)$.
Does $g \in N(x) \implies g^{-1} \in N(x)$?

$$\begin{aligned} gxg^{-1} &= x \\ g^{-1}gxg^{-1} &= g^{-1}x \\ xg^{-1} &= g^{-1}x \\ xg^{-1}g &= g^{-1}xg \\ x &= g^{-1}xg \end{aligned}$$

Thus $g^{-1} \in N(x)$, and $N(x)$ is a subgroup of G.

c. Defining $Z(G) = \{g \in G \,|\, gyg^{-1} = y \forall\ y \in G\}$, show that $Z(G)$ is a subgroup of G. We mirror the proof of (b).
Is $1 \in Z(G)$? $\forall\ y \in G, 1y1^{-1} = 1y1 = y$, so $1 \in Z(G)$.
Does $g, h \in Z(G) \implies gh \in Z(G)$? $\forall\ y \in G (gh)y(gh)^{-1} = ghyh^{-1}g^{-1} = gyg^{-1} = y$, so $gh \in Z(G)$.
Does $g \in Z(G) \implies g^{-1} \in Z(G)$?

$$\begin{aligned} \forall\ y \in G, gyg^{-1} &= y \\ \forall\ y \in G, g^{-1}gyg^{-1} &= g^{-1}y \\ \forall\ y \in G, yg^{-1} &= g^{-1}y \\ \forall\ y \in G, yg^{-1}g &= g^{-1}yg \\ \forall\ y \in G, y &= g^{-1}yg \end{aligned}$$

Thus $g^{-1} \in Z(G)$, and $Z(G)$ is a subgroup of G.
It was possible to show that

$$Z(G) = \bigcap_{x \in G} N(x)$$

and thus that if $N(X)$ is a subgroup, then the intersection of all $N(x)$ must be a subgroup. You needed to prove that the subgroup relation is closed under intersection, though, which many people neglected to do.

Problem 6. (4) To produce a maximal spanning tree, we simply modify an existing Minimum Spanning Tree algorithm (Prim's, Kruskal's, Boruvka's, etc.) and change the edge sorting and/or selection criteria from lightest to heaviest. Proving correctness will mirror the correctness proofs for the MST algorithm with the same modifications.

Alternatively, you could simply write a wrapper for an existing Minimum Spanning Tree algorithm that simply negates the weights of all the edges and then calls a known Minimum Spanning Tree algorithm. The proof of correctness would simply rely on the fact that Minimum Spanning Trees make no restriction on negative weight edges, and that the Minimum Spanning Tree of the negated graph is indeed a Maximal Spanning Tree of the original graph.

Problem 7. (4) Prove that a simple graph without loops has at least 2 vertices of the same degree.

□ The maximal degree on a graph with n nodes is $n-1$, and the minimal degree is 0. Consider two cases. In the first case, the maximal degree is attained, and so there exists some node which is directly connected to all others. Thus all nodes must have degree at least 1, and the degrees of the n nodes range from 1 to $n-1$. In the second case, there is no node with degree $n-1$. It is thus possible for there to be isolated nodes, but since the highest possible degree is now $n-2$, the degrees of the n nodes range in this case from 0 to $n-2$.

Note that in either case, there are only $n-1$ possible degrees, and n nodes. Thus, by the Pigeonhole principle, there must exist some degree k such that at least 2 nodes have degree k. ■

Problem 8. (8)

a. The two maximal weakly connected components are $\{1, 2, 3\}$ and $\{1, 2, 4\}$. If there were a way to partition the graph into maximal, weakly connected subgraphs, then node 4 must be contained in a maximal WCS. We know that this component is $\{1, 2, 4\}$. This implies that 3 is itself a maximally connected subgraph, but this contradicts the fact that $\{1, 2, 3\}$ is the maximal WCS containing 3. Thus we cannot achieve the desired partition.

b)Maximal SCC's are: $\{3, 6, 5\}, \{1, 7, 2\}, \{4\}$

c)Take node $n \in V(J)$ and $n \in V(K)$. Then for any node $j \in V(J)$, $\exists$ a path from j to n and from n to j, since J is a SCC. Also, for any node $k \in V(K)$, $\exists$ a path from n to k and from k to n. Thus we can construct a path from j to k and from k to j through n by concatenating the path from j to n with the one from n to k, and the path from k to n with that from n to j. Therefore $J \cup K$ is a strongly connected component.

d)We want to show that every vertex is in exactly one maximal strongly connected component. We know from (c) that if a vertex is present in more than 1 SCC, then those two SCC's can be combined to form another (larger) SCC. Thus if a vertex V is in a maximal SCC, then it cannot be a member of any other maximal SCC — else those two SCC's could be merged into a larger SCC, contradicting the initial maximality. Finally, all vertices will be a part of at least one SCC - namely, the SCC containing just that one vertex.

Problem 10. (4) Solve for x: $3x \equiv 1 \pmod 5$ with $2x \equiv 6 \pmod 8$. The inverse of 3 (mod 5) is 2. Multiplying the first equation by 2 yields $6x \equiv 2 \pmod 5$, therefore $x \equiv 2 \pmod 5$. So $x = 5s + 2$ for some integer s. (*)

Substituting in the second equation, we get:

$$\begin{aligned} 10s + 4 &\equiv 6 \pmod 8 \\ 10s &\equiv 2 \pmod 8 \\ 2s &\equiv 2 \pmod 8 \\ s &\equiv 1 \pmod 4 \end{aligned}$$

So $s = 4t + 1$; substituting that in for s in equation (*) above yields $x = 20t + 5 + 2 = 20t + 7$ for some integer t. Therefore $x \equiv 7 \pmod{20}$ is the solution, with $x = 7$ the simplest such solution.

Problem 11. (4) We must solve the set of simultaneous equations:

$$\begin{aligned} &(1) && x \equiv 3 \pmod{15} \\ &(2) && x \equiv 2 \pmod 7 \\ &(3) && x \equiv 0 \pmod 4 \end{aligned}$$

Let us first solve the latter pair for x:

From (3), we have $x = 4 \cdot s$ for some integer s. Substituting into (2) gives us $4 \cdot s \equiv 2 \pmod 7$ By inspection, 2 is the inverse of 4 (mod 7), so we multiply both sides to yield $2 \cdot 4 \cdot s \equiv 2 \cdot 2 \pmod 7$ and thus $s \equiv 4 \pmod 7$. Thus $s = 7 \cdot t + 4$ for some integer t. Substituting back into $x = 4 \cdot s$, we have $x = 4(7 \cdot t + 4) = 28 * t + 16$. Substituting this into (1) yields $28 \cdot t + 16 \equiv 3 \pmod{15}$. Subtracting 16 from both sides and reducing 28 to 13 mod 15 gives $13 \cdot t \equiv -13 \equiv 2 \pmod{15}$. By inspection, 7 is the inverse of 13 (mod 15), thus $7 \cdot 13 \cdot t \equiv 7 \cdot 2 \pmod{15}$ and thus $t \equiv 14 \pmod{15}$. Thus $t = 15 \cdot u + 14$ for some integer u. Substituting back into $x = 28 \cdot t + 16$, we have $x = 28(15 \cdot u + 14) + 16$. We are interested in the smallest positive such solution x, which arises when $u = 0$: $x = 28(14) + 16 = 408$ pieces of gold.

Problem 9. (8)

a. Here's one possible path:

$$(vs)(st)(ty)(yx)(xw)(wu)(uv)(vu)(uw)(ws)(sw)(wt)(tw)(wx)(xy)(yt)(ts)(sv)$$

b. Here's the algorithm in Java-style pseudocode (the book's algorithmic language is hard to understand):
Say you have an object `Chamber`, with a method `getPaths()`, that gives you a list of paths. Each path object has a method, `marked()`, that tells you if it has been traversed, and a method `mark()` that marks it. Lastly, path has another method, `getOtherEnd(Chamber c)`, that gives you the chamber that this path connects `c` to. So here's the algorithm:

```
public void TarrySearch(Chamber c) {
 List l = c.getPaths();
 \\ go through all the paths connected to chamber c
 for (Iterator iter = l.iterator(); iter.hasNext();) {
  Path p = (Path)iter.next();
  \\ if the path has not already been traversed
  if (!p.marked()) {
   p.mark();
   Chamber next = p.getOtherEnd(c);
   List l2 = next.getPaths();
   boolean marked = false;
   \\ see if the other chamber has been visited
   for (Iterator iter2 = l2.iterator(); iter2.hasNext();) {
    Path p2 = (Path)iter2.next();
    if (p2.marked())
      marked = true;
   }
   \\ if it hasn't, go to it
   if (!marked) TarrySearch(next);
  }
 }
}
```

Note that there is no need to handle the marking with E's, as that is handled by the recursive function returning.

c. We were quite lenient in grading this part. Anyone who provided a convincing argument that the algorithm would traverse every edge in both directions received full credit.

Basically, the argument should have been as follows. There are two different cases when traversing an edge. Either you find that the chamber at the other end has already been visited, or you find that it has not. In the first case, you will immediately travel back along the same edge, exploring it twice. In the second case, you will continue on you merry way. Note that in this case, the edge will be marked with an E. Yet once you reach the point where there are no longer any unvisited edges, you will begin traversing back, following the edges with E's marked on them. It is at this point that all those edges will be traversed for the second time. One can easily see that only one such path of E's can exist, and so you will end up back at the start with everything visited twice.

d. The Tarry search is quite similar to a DFS, but it is NOT exactly a DFS. It does work in the same depth first manner, except it works off of edges rather than nodes. A DFS keeps track of what nodes have been seen, and so will do a bit less work than the Tarry algorithm (none of that traversing-and-then-going-back stuff).