Exercise 1.6

Given the shift cipher S over $\mathbb{Z}_{26}$, a key is an involutory key if the encryption and decryption function are the same, so if and only if $x = d_k(e_k(x)) = e_k(e_k(x)) = x + 2k \pmod{26}\ \forall x$ which yields the condition $x = x + 2k \pmod{26}\ \forall x$. We can find exactly two keys k that solve this equation: $k = 0$ and $k = 13$. Thus, these are the involutory keys of S.

Exercise 1.7

Given an affine cipher over $\mathbb{Z}_m$ for $m \in \{30, 100, 1225\}$ we must only choose keys (a, b) such that $\gcd(a, 26) = 1$. We can determine the size of the set $A = \{a \mid \gcd(a, m) = 1\}$ using Euler's totient function as follows

$$|A| = \phi(m) = m \prod_{p|m} \left(1 - \frac{1}{p}\right)$$

where $\prod_{p|m}$ is the product over the distinct primes in the unique prime factorization of m. We can then determine the size of the key space K_m of the affine cipher for m as $|K_m| = m \times \phi(m)$. For $m \in \{30, 100, 1225\}$ we obtain

$$\begin{aligned}
\phi(30) &= 30 \times \left(1 - \frac{1}{2}\right) \times \left(1 - \frac{1}{3}\right) \times \left(1 - \frac{1}{5}\right) = 8 \text{ with prime factorization } 30 = 2 \times 3 \times 5,\\
\phi(100) &= 100 \times \left(1 - \frac{1}{2}\right) \times \left(1 - \frac{1}{5}\right) = 40 \text{ with prime factorization } 100 = 2 \times 5 \times 5,\\
\phi(1225) &= 30 \times \left(1 - \frac{1}{5}\right) \times \left(1 - \frac{1}{7}\right) = 840 \text{ with prime factorization } 1225 = 5 \times 5 \times 7 \times 7
\end{aligned}$$

and thus the key space sizes are

$$\begin{aligned}
K_{30} &= 30 \times \phi(30) = 30 \times 8 = 240,\\
K_{100} &= 100 \times \phi(100) = 100 \times 40 = 4000,\\
K_{1225} &= 1225 \times \phi(1225) = 1225 \times 840 = 1029000.
\end{aligned}$$

Exercise 1.17

(a) A key k in the permutation cipher is given by a permutation $k = \pi$. The encryption and decryption functions can then be written as $e_k(x) = \pi(x)$ and $d_k(y) = \pi^{-1}(y)$. A key is called involutory exactly if $e_k(x) = d_k(x)\ \forall x$ so if $e_k(e_k(x)) = \pi(\pi(x)) = x$. We can see that $\pi(\pi(x)) = x$ is true exactly when the permutation is its own inverse $\pi = \pi^{-1}$. Thus, $\pi(i) = j$ implies $\pi(j) = i$ for all $i, j \in \{1, \ldots, m\}$.

(b) For $m = 2$ there are 2 involutory keys (identity and swapping). As we have seen in (a) each involutory key in the permutation cipher is a permutation π that satisfies $\pi^2 = \text{id}$ where id is the identity function. We call these permutations of order 2. Thus, the number of involutory keys for m is the number of permutations π of order 2 over $\{1, \ldots, m\}$. We know that for any such permutation $\pi(i) = j$ implies $\pi(j) = i$ for all $i, j \in \{1, \ldots, m\}$, so these permutations can be written as a product of disjointed 2-cycles.

Now suppose π is a product of k disjoint 2-cycles. It permutes $2k$ plaintext elements (as each cycle contains two different elements). There are $\binom{m}{2k}$ ways to choose exactly $2k$ elements from m. Forming 2-cycles over these $2k$ elements is equivalent to forming pairs of numbers and there are exactly $\frac{(2k)!}{k!2^k}$ ways to form k pairs with $2k$ elements. Plugging this together, we obtain

$$\binom{m}{2k}\frac{(2k)!}{k!2^k}$$

for the number of permutations with k disjoint 2-cycles. We can now sum over the number of possible 2-cycles in a permutation over m elements: there are at most $\lfloor \frac{m}{2} \rfloor$ 2-cycles over m. Thus, the total number of permutations of order 2 is given by

$$\sum_{k=0}^{\lfloor \frac{m}{2} \rfloor} \binom{m}{2k}\frac{(2k)!}{k!2^k}$$

which is also the number of involutory keys of the permutation cipher for m. We can then determine the number of involutory keys for any given m:

For $m = 2$:	2 involutory keys
For $m = 3$:	4 involutory keys
For $m = 4$:	10 involutory keys
For $m = 5$:	26 involutory keys
For $m = 6$:	76 involutory keys

Exercise 1.30

We can use exhaustive key search for the described stream cipher to obtain the correct key. Using a short Python script (see figure 1) we obtain all possible pairs of keys and plaintexts (see figure 2). Given this list we can easily identify $K = 11$ as the correct key as this is the only key giving a decoding that is an English language sentence. The plaintext for the given ciphertext is thus `THEFIRSTDEPOSITCONSISTEDOFONETHOUSANDANDFOURTEENPOUNDSOFGOLD` which (with some added spaces) gives us the plaintext sentence "The first deposit consisted of one thousand and fourteen pounds of gold".

Exercise 2.9

We perform Huffman's algorithm and write the result as a tree where each vertex v is the sum of probabilities in the subtree with root v. Each leaf node l represents a plaintext $p \in X$. Each edge denotes the encoding for the given subtree so that an encoding of a leaf node can be read as the labels along the path starting at the root node. The algorithm yields the tree seen in figure 3. From the tree we obtain the following Huffman Encoding: a = 00, b = 10, c = 11, d = 011, e = 010. We can compute $H(X)$ by

$$\begin{aligned} H(X) = &-(0.32 \times \log_2(0.32) + 0.23 \times \log_2(0.23) + 0.2 \times \log_2(0.2) \\ &+ 0.15 \times \log_2(0.15) + 0.1 \times \log_2(0.1)) \\ \approx\ & 2.22082 \end{aligned}$$

Our encoding takes 2.4 bits on average, which is slightly higher, but very close to $H(X)$.

```python
# Generate dictionary for number (0-25) to alphabet
alph = {chr(i+97):i for i in range(0,26)}
invalph = {v: k for k, v in alph.items()}

# Permutation dictionary and its inverse
pi = {0: 23, 1: 13, 2: 24, 3: 0, 4: 7, 5: 15, 6: 14, 7: 6, 8: 25, 9: 16, 10:
     22, 11: 1, 12: 19, 13: 18, 14: 5, 15: 11, 16: 17, 17: 2, 18: 21, 19: 12,
     20: 20, 21: 4, 22: 10, 23: 9, 24: 3, 25: 8}
invpi = {v: k for k, v in pi.items()}

# Decryption function
def dec(list, key):
    x = []
    for i, y in enumerate(list):
        z = (key + i - 1) % 26
        x.append(invpi[(y - z) % 26])
    return x

# Convert string to its numeric representation
def string2num(c):
    return [alph[x] for x in c.lower()]

# Convert numeric representation back to string
def num2string(y):
    return ''.join([invalph[i] for i in y]).upper()

# Obtain numeric representation of ciphertext
c = "WRTCNRLDSAFARWKXFTXCZRNHNYPDTZUUKMPLUSOXNEUDOKLXRMCBKGRCCURR"
num = string2num(c)

# Exhaustive key search over key space
for i in range(0,26):
    y = dec(num,i)
    p = num2string(y)
    print("K = " + str(i) + ": " + p)
```

Figure 1: Python script for exhaustive key search on the given stream cipher for the given ciphertext. The script outputs all possible keys and their respective plaintexts. Due to the small number of keys it is easy to then identify the correct key-plaintext pair.

```
K = 0: AQNDWBHAPNKJHWAXJYHWHANPJDJYNAQJOHZYPZYPDJOBANNYKJOYPHJDIJTP
K = 1: KJQIXTOKWQSFOXKZFROXOKQWFIFRQKJFVOERWERWIFVTKQQRSFVRWOFICFPW
K = 2: SFJCZPVSXJUGVZSEGLVZVSJXGCGLJSFGYVHLXHLXCGYPSJJLUGYLXVGCAGWX
K = 3: UGFAEWYUZFMBYEUHBDYEYUFZBABDFUGBRYODZODZABRWUFFDMBRDZYBAKBXZ
K = 4: MBGKHXRMEGNTRHMOTIRHRMGETKTIGMBTLRVIEVIEKTLXMGGINTLIERTKSTZE
K = 5: NTBSOZLNHBQPLONVPCLOLNBHPSPCBNTPDLYCHYCHSPDZNBBCQPDCHLPSUPEH
K = 6: QPTUVEDQOTJWDVQYWADVDQTOWUWATQPWIDRAORAOUWIEQTTAJWIAODWUMWHO
K = 7: JWPMYHIJVPFXIYJRXKIYIJPVXMXKPJWXCILKVLKVMXCHJPPKFXCKVIXMNXOV
K = 8: FXWNROCFYWGZCRFLZSCRCFWYZNZSWFXZACDSYDSYNZAOFWWSGZASYCZNQZVY
K = 9: GZXQLVAGRXBEALGDEUALAGXREQEUXGZEKAIURIURQEKVGXXUBEKURAEQJEYR
K = 10: BEZJDYKBLZTHKDBIHMKDKBZLHJHMZBEHSKCMLCMLJHSYBZZMTHSMLKHJFHRL
K = 11: THEFIRSTDEPOSITCONSISTEDOFONETHOUSANDANDFOURTEENPOUNDSOFGOLD
K = 12: POHGCLUPIHWVUCPAVQUCUPHIVGVQHPOVMUKQIKQIGVMLPHHQWVMQIUVGBVDI
K = 13: WVOBADMWCOXYMAWKYJMAMWOCYBYJOWVYNMSJCSJCBYNDWOOJXYNJCMYBTYIC
K = 14: XYVTKINXAVZRNKXSRFNKNXVARTRFVXYRQNUFAUFATRQIXVVFZRQFANRTPRCA
K = 15: ZRYPSCQZKYELQSZULGQSQZYKLPLGYZRLJQMGKMGKPLJCZYYGELJGKQLPWLAK
K = 16: ELRWUAJESRHDJUEMDBJUJERSDWDBRELDFJNBSNBSWDFAERRBHDFBSJDWXDKS
K = 17: HDLXMKFHULOIFMHNITFMFHLUIXITLHDIGFQTUQTUXIGKHLLTOIGTUFIXZISU
K = 18: OIDZNSGOMDVCGNOQCPGNGODMCZCPDOICBGJPMJPMZCBSODDPVCBPMGCZECUM
K = 19: VCIEQUBVNIYABQVJAWBQBVINAEAWIVCATBFWNFWNEATUVIIWYATWNBAEHAMN
K = 20: YACHJMTYQCRKTJYFKXTJTYCQKHKXCYAKPTGXQGXQHKPMYCCXRKPXQTKHOKNQ
K = 21: RKAOFNPRJALSPFRGSZPFPRAJSOSZARKSWPBZJBZJOSWNRAAZLSWZJPSOVSQJ
K = 22: LSKVGQWLFKDUWGLBUEWGWLKFUVUEKLSUXWTEFTEFVUXQLKKEDUXEFWUVYUJF
K = 23: DUSYBJXDGSIMXBDTMHXBXDSGMYMHSDUMZXPHGPHGYMZJDSSHIMZHGXMYRMFG
K = 24: IMURTFZIBUCNZTIPNOZTZIUBNRNOUIMNEZWOBWOBRNEFIUUOCNEOBZNRLNGB
K = 25: CNMLPGECTMAQEPCWQVEPECMTQLQVMCNQHEXVTXVTLQHGCMMVAQHVTEQLDQBT
```

Figure 2: A list of all possible keys and their respective plaintexts. Output of the Python script from figure 1.

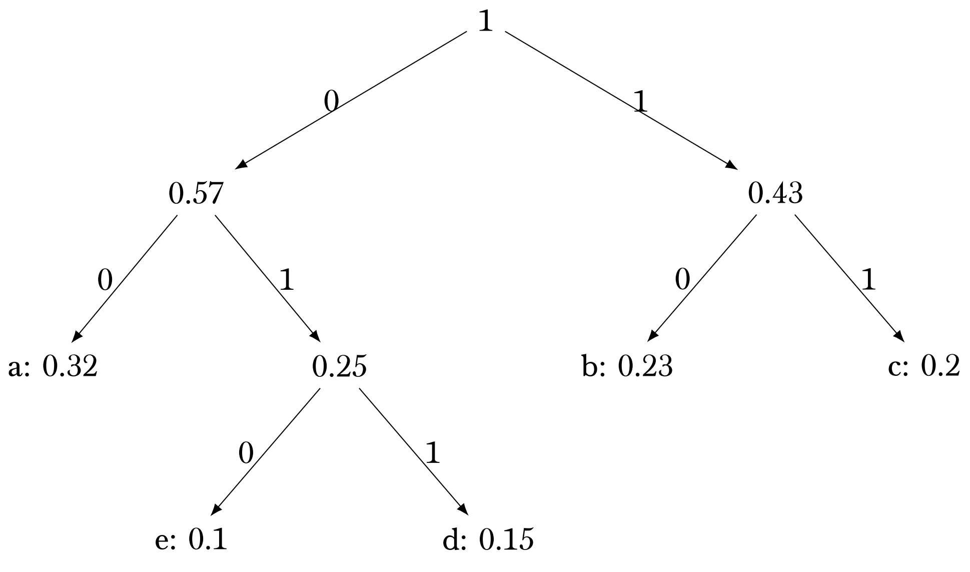

Figure 3: Tree resulting from the Huffman algorithm

Exercise 2.13

Entropy H of a random variable X is defined as

$$H(X) = -\sum_{v \in V} \Pr(X = v) \log_2 \Pr(X = v).$$

We can then compute the entropy for the plaintext $H(P)$, for the ciphertext $H(C)$, for the key space $H(K)$, the key equivocation $H(K|C)$ and $H(P|C)$ like:

$$\begin{aligned} H(P) &= -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{6}\log_2\left(\frac{1}{6}\right)\right) \\ &= \left(\frac{1}{2}\log_2(2) + \frac{1}{3}\log_2(3) + \frac{1}{6}\log_2(6)\right) \\ &\approx 1.45915 \end{aligned}$$

Compute ciphertext probabilities by weighted sum of number of occurences:

$$\begin{aligned} \Pr(1) &= \Pr(a) \times \frac{1}{3} + \Pr(b) \times \frac{0}{3} + \Pr(c) \times \frac{1}{3} = \frac{2}{9} \\ \Pr(2) &= \Pr(a) \times \frac{1}{3} + \Pr(b) \times \frac{1}{3} + \Pr(c) \times \frac{0}{3} = \frac{5}{18} \\ \Pr(3) &= \Pr(a) \times \frac{1}{3} + \Pr(b) \times \frac{1}{3} + \Pr(c) \times \frac{1}{3} = \frac{1}{3} \\ \Pr(4) &= \Pr(a) \times \frac{1}{3} + \Pr(b) \times \frac{1}{3} + \Pr(c) \times \frac{0}{3} = \frac{1}{6} \end{aligned}$$

$$H(C) = -\left(\frac{2}{9}\log_2\left(\frac{2}{9}\right) + \frac{5}{18}\log_2\left(\frac{5}{18}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{6}\log_2\left(\frac{1}{6}\right)\right) \approx 1.95469$$

As the three keys are equiprobable we get

$$H(K) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) = \log_2 3 \approx 1.58496$$

and given the theorem we can compute $H(K|C)$ as follows:

$$H(K|C) = H(K) + H(P) - H(C) \approx 1.58496 + 1.45915 - 1.95469 = 1.08942$$

Finally, as each plaintext is uniquely defined by its key given a ciphertext:

$$H(P|C) = H(K|C) \approx 1.08942$$

Exercise 2.19

The cryptosystem given by $S_1 \times S_2$ has an encryption function $e_k(x) = x + (s_1 + s_2) \pmod{26}$ and decryption function $d_k(x) = x - (s_1 + s_2) \pmod{26}$ for a key $k = (s_1, s_2)$. We can now define $s = s_1 + s_2$ and write the encryption function as $e_k(x) = x + s \pmod{26}$ and conversely the decryption function as $d_k(x) = x - s \pmod{26}$. It is easy to see that this is the definition of the shift cipher. Let Pr_1, Pr_2 be the probability distributions over the keys of S_1, S_2. We have $\text{Pr}(s_1, s_2) = \text{Pr}_1(s_1)\text{Pr}_2(s_2)$, so the keys are chosen independently and for $s = s_1 + s_2$ we see that $\text{Pr}(s)$ is equiprobable for all s because Pr_1 is equiprobable. Then $S_1 \times S_2$ is exactly the shift cipher and it follows that $S_1 \times S_2 = S_1$.