1. (a) 由大数定律

$$\frac{1}{n}\log q(X_1,\ldots,X_n)=\frac{1}{n}\sum_{i=1}^{n}\log q(X_i)\xrightarrow{P}\sum_{j=1}^{m}p(x_j)\log q(x_j)$$

(b)

$$\frac{1}{n}\log\frac{q(X_1,\ldots,X_n)}{p(X_1,\ldots,X_n)}\xrightarrow{P}-\sum_{j=1}^{m}p(x_j)\log\frac{p(x_j)}{q(x_j)}=-D(p||q)$$

2. (a) A^n 是典型集，由典型集的性质 $|A^n|\le 2^{n(H+\epsilon)}$ 及 $|A^n\cap B^n|\le 2^{n(H+\epsilon)}$ 可得。

(b) 由大数定律, 当 n 充分大时有

$$\begin{aligned}\Pr(A^n)&=\Pr\{|-\frac{1}{n}\log p(X_1,\ldots,X_n)-H(X)|<\epsilon\} &\ge\frac{3}{4}\\ \Pr(B^n)&=\Pr\{|-\frac{1}{n}\sum_{i=1}^{n}x_i-\mu|\le\epsilon\} &\ge\frac{3}{4}\end{aligned}$$

$$\Rightarrow\Pr(A^n\cap B^n)=1-\Pr(\bar{A}^n)-\Pr(\bar{B}^n)\ge\tfrac{1}{2}$$

$$\begin{aligned}|A^n\cap B^n|&=\sum_{x\in A^n\cap B^n}1\\ &\ge\sum_{x\in A^n\cap B^n}p(x^n)2^{n(H-\epsilon)}\\ &=\Pr(A^n\cap B^n)2^{n(H-\epsilon)}\\ &\ge(1/2)2^{n(H-\epsilon)}\end{aligned}$$

3. (a) $\log(1+\binom{100}{1}+\binom{100}{2}+\binom{100}{3})\approx 17.3$，所以至少需要 18 位长的码字。

(b) $p=1-\sum_{i=1}^{3}\binom{100}{i}p(0)^{101-i}p(1)^{i-1}=1.6‰$

(c) 设 $Y=\sum_{i=1}^{100}X_i\Rightarrow\mathbb{E}[Y]=0.5,\mathrm{Var}[Y]=0.4975$ 由 Chebyshev 不等式

$$\Pr[|Y-\mathbb{E}[Y]|\ge a]\le\frac{1}{a^2}\mathrm{Var}[Y]$$

取 $a=3.5$ and consider also that Y take discrete values $1,2,3,4,\ldots$. Therefore

$$\Pr[Y\ge 4]\le 4\%$$

which is much larger than the accurate result in (b).

4. We expand $p = 0.p_1p_2\ldots$ as a binary number. Let $U = 0.Z_1Z_2\ldots$, the sequence Z treated as a binary number. It is well known that U is uniformly distributed on $[0, 1)$. Thus, we generate $X = 1$ if $U < p$ and 0 otherwise.

   The procedure for generated X would therefore examine $Z_1, Z_2\ldots$ and compare with $p_1, p_2\ldots$, and generate a_1 at the first time one of the Z_i's is less than the corresponding p_i(which means $Z_i = 0, p_i = 1$, and its probability is $\frac{1}{4}$) and generate a 0 the first time one of the Z_i's is greater than the corresponding p_i's($\Pr(Z_i = 1, p_i = 0) = \frac{1}{4}$). Thus the probability that X is generated after seeing the first bit of Z is the probability that $Z_1 \neq p1$, i.e., with probability $\frac{1}{2}$. Similarly, X is generated after 2 bits of Z if $Z_1 = p_1$ and $Z_2 \neq p_2$, which occurs with probability $\frac{1}{4}$. Thus

$$\begin{aligned}\mathbb{E}[N] &= 1 \cdot \frac{1}{2} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{8} + \ldots \\ &= 2\end{aligned}$$

5. (a) $C1$ is nonsingular, since 00 can be decoded as 0 and 0 or 00;

   $C2$ is nonsingular, since 010 can be decoded as 0 and 10 or 010;

   $C3$ is instantaneous since no code is the suffix of the other codes.

   $C4$ is nonsingular, since 1010 can be decoded as $10, 10$ or $101, 0$.

   (b) Since code C is nonsingular, $C(x_i) \neq C(x_j) \iff x_i \neq x_j \Rightarrow$ H(C(X))=H(X).

   Since code C is nonuniquely decodable code, the state of $C(X^n)$ is less than X^n, and by the simple inequality

   $-(p_1 + p_2)\log(p_1 + p_2) < -p_1 \log p_1 - p_2 \log p_2 \Rightarrow H(C(X^n)) < H(X^n)$

6. (a) Since $-\log q(x) \leq l(x) < -\log q(x) + 1 \Rightarrow$

$$-\sum_{x\in\mathcal{X}} p(x)\log q(x) \leq l(x) < -\sum_{x\in\mathcal{X}} p(x)\log q(x) + 1 \Rightarrow$$

   $H(p) + D(p||q) \leq \mathbb{E}_p[l(X)] < H(p) + D(p||q) + 1$

7. (a) consider $\hat{W}_i$, which can be regarded as probability distribution for m random variables. Then the merge process can be modeled as building Huffman tree. If we merge W_i and W_j, it means that the code sequence length for x_i and x_j added by one. Therefore $\frac{V}{W}$ is the average code length of Huffman encoding. For any other encoding by using binary tree, it is also a scheme of instantaneous codes and by the property of Huffman encoding, we have $\frac{V}{W} \leq \frac{V'}{W}$, where V' is achieved by arbitrary sequences of pairwise merges. And we can get that V is the minimum.

   (b) By the property of Huffman encoding, $H(\hat{W}) \leq \frac{V}{W} \leq H(\hat{W})+1 \Rightarrow WH(\hat{W}) \leq V \leq WH(\hat{W}) + W$

8. (a) The first thing to recognize in this problem is that the player cannot cover more than 63 $(1 + 2 + 4 + 8 + 16 + 32)$ values of X with 6 questions. This can be easily seen by induction. With one question, there is only one value of X that can be covered. With two questions, there is one value of X that can be covered with the first question, and depending on the answer to the first question, there are two possible values of X that can be asked in the next question. By extending this argument, we see that we can ask at more 63 different questions of the form "Is $X = i$?" with 6 questions. (The fact that we have narrowed the range at the end is irrelevant, if we have not isolated the value of X.)

   This observation is important since it is not comman sense. Consider the special case when $p_i = \frac{1}{100}$. People can really think half-divide is the only optimal solution, however, for any 63 numbers out of $\{1, \ldots, 100\}$, we can use half-divide (from median) and the probability of winning all equals 63%. Therefore, there are at least $\binom{100}{63}$ optimal procedures. For general case, we can pick out the most favorable 63 outcomes, that is, the largest 63 numbers of $p(i)$. Then proceed as usual to get the maximal expected winnings as $\sum_{k=1}^{63} p(j_k)v(j_k)$, where $p(j_1)v(j_1) \geq \cdots \geq p(j_{100})v(j_{100})$ and $j_1, \ldots, j_{100}$ is a permutation of $1, \ldots, 100$.

   (b) In this case, we assume we can ask question like "Is $X = 2.5$

?", that is,decimal number is allowed. Since each number must be guessed before terminiting. The "Yes" or "No"(high or low) sequence produced by the given strategy can be treated as binary encoding of the number from 1 to 100. Then the number of questions is equivalent to the length of code. To maximum $\sum_{x=1}^{100} p(x)(v(x) - l(x))$ for given $p(x), v(x)$, we should minimize $\sum_{x=1}^{100} p(x)l(x)$. And Huffman encoding can be used to accomplish this.

Our strategy follows the Huffman tree. We label each node in the tree with a number. Each leaf node of Huffman tree corresponds to a number from 1 to 100 and non-leaf node are stuffed by decimal numbers (not integer) such that it is larger than the largest in the left sub-tree and smaller than the smallest in the right-tree. Then we start from the root node (with value a e.g.), we ask "Is $X = a$?". Is a is not integer (leaf node), then the answer can only be high (follow right sub-tree) or low (follow left sub-tree). Our construction gurantees that when we come to leaf node, we get the right answer.

The upper bound of the expected return is $\sum_{x=1}^{100} p(x)v(x) - H(X)$

(c) We solve a problem of optimization.

$$\min \sum_{x=1}^{100} p(x)v(x) - H(X) \tag{1}$$

$$s.t. \sum_{x=1}^{100} p(x) = 1 \tag{2}$$

$\Rightarrow p(x) = \frac{\exp(-v(x))}{\sum_{x=1}^{100} \exp(-v(x))}$ and the expected return of the user is $-\log(\sum_{x=1}^{100} \exp(-v(x)))$

9. (a) By the principle of Rearrangement Inequality, taste the bottle of milk with larger probability of souring first. Then we can get the minimum expected number of tasting as :$[p_1, p_2, \ldots, p_6] \cdot [1, 2, 3, 4, 5, 5] = 2.89$

   (b) See (a) for detail.

(c) The number of tasting is equivalent to code length of encoding $X \sim (p_1, p_2, \ldots, p_6)$. We should use Huffman encoding to achieve the minimum. The code lengths are $[2, 2, 3, 3, 3, 3]$ respectively. Therefore, the minimum is $[p_1, p_2, \ldots, p_6] \cdot [1, 2, 3, 4, 5, 5] = 2.54$ if mixing is allowed.

10. (a) If l_i is unconstrained, then to minimize C the constraint inequality can be achieved by equality $\sum 2^{-l_i} = 1$. We use Lagrange multiplier to calculate the optimal l_i for such a problem by differentiate on $\sum_{i=1}^{m}(p_i c_i l_i - \lambda 2^{-l_i}) - 1 \Rightarrow 2^{-l_i} = \frac{p_i c_i}{\lambda \ln 2} \Rightarrow$

$$l_i^* = -\log \frac{p_i c_i}{\sum_{i=1}^{m} p_i c_i} \Rightarrow C^* = u \log u - \sum_{i=1}^{m} p_i c_i \log(p_i c_i), \text{where } u = \sum_{i=1}^{m} p_i c_i$$

(b) Combine the smallest two $p_i c_i$ and by the same proof technique we can get the smallest cost code scheme.

(c) Since C_{Huffman} is obtained by further imposing integer constrains on l_i, we have $C^* \leq C_{\text{Huffman}}$. Next, since l_i^* obtained in (a) satisfies $\sum 2^{-l_i^*} = 1$, let $\tilde{l}_i = \lceil l_i^* \rceil$. Then $\tilde{l}_i \geq l_i^*$, therefore $\sum 2^{-\tilde{l}_i} \leq 1$. By Kraft's inequality, there exists a code scheme C' such that its code length is the sequnce $\{\tilde{l}_1, \ldots, \tilde{l}_m\}$. On the other hand, we have $\tilde{l}_i < l_i^* + 1$. Multiply this inequality with $p_i c_i$ and sum over i, we can get $C' < C^* + \sum_{i=1}^{m} p_i c_i$. By the optimal property of Huffman encoding, $C_{\text{Huffman}} \leq C'$. In conclusion:

$$C^* \leq C_{\text{Huffman}} \leq C' < C^* + \sum_{i=1}^{m} p_i c_i$$