Solutions to Homework 3

1. Problem 4.11 on pg. 93 of the text.

Solution:

Stationary processes

(a) By stationarity and the chain rule for entropy, we have

\[ H(X_0) + H(X_n|X_0) = H(X_0, X_n) \]
\[ = H(X_{-n}, X_0) \]
\[ = H(X_0) + H(X_{-n}|X_0) \]

so that \( H(X_n|X_0) = H(X_{-n}|X_0) \) is true for any stationary stochastic process.

(b) This is false in general. Consider, for instance, the \{0,1\}-valued stationary stochastic process defined by

\[ P((X_0, X_1, X_2) = (0,0,1)) = P((X_0, X_1, X_2) = (0,1,0)) = P((X_0, X_1, X_2) = (1,0,0)) = 1/3 \]

with

\[ X_n = X_{n+3k} \text{ for all } n \in \mathbb{Z} \text{ and all } k \in \mathbb{Z} \]

\((X_n)\) can be thought of as the stationary discrete time renewal process corresponding to a renewal time of 3.) Then \( H(X_3|X_0) = 0 \), while

\[ H(X_2|X_0) = \frac{1}{3} H(X_2|X_0 = 1) + \frac{2}{3} H(X_2|X_0 = 0) \]
\[ = \frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 1 \]
\[ = \frac{2}{3} \]
\[ > 0 \cdot \]

Here we have observed that conditional on \( X_0 = 1 \), we have \( X_2 = 0 \), while conditional on \( X_0 = 0 \), \( X_2 \) is either 0 or 1, equiprobably.

Note however that if \((X_n)\) is a Markov process, the claim is true, because

\[ H(X_n|X_0) \geq H(X_n|X_0, X_1) \]
\[ = H(X_n|X_1) \]
\[ = H(X_{n-1}|X_0) \]

where the Markov property was used in the second step.
(c) This statement is true for any stationary stochastic process. We have
\[ H(X_{n+1}|X_1^n, X_{n+2}) \leq H(X_{n+1}|X_2^n, X_{n+2}) = H(X_n|X_1^{n-1}, X_{n+1}) \]
where the first step is because conditioning does not increase entropy, and the second step uses the stationarity of the stochastic process.

(d) This statement is true for any stationary stochastic process, because
\[ H(X_{n+1} | X_0^n, X_{n+2}^{2(n+1)}) \overset{(a)}{=} H(X_0 | X_{-(n+1)}^{-1}, X_1^{n+1}) \]
\[ \overset{(b)}{\leq} H(X_0 | X_{-n}^{-1}, X_1^n) \]
\[ \overset{(c)}{=} H(X_n | X_0^{-1}, X_{n+1}^{2n}) \]
where in steps (a) and (c) we have used the stationarity of the process and in step (b) we observe that conditioning reduces entropy.

2. Problem 5.27 on pp. 149 -150 of the text.

Solution:
Sardinas-Patterson test for unique decodability

(a) For a pair of nonempty subsets of strings over the alphabet, \( A \) and \( B \), let
\[ \mathcal{D}_{AB} \overset{\Delta}{=} \{ d \neq \emptyset : ad = b \text{ for some } a \in A \text{ and some } b \in B \} . \]
Here \( ad \) denotes the concatenation of \( a \) and \( d \). Thus \( \mathcal{D}_{AB} \) is the set of dangling suffixes generated from strings in \( B \) by erasing prefixes in \( A \).

The set \( S \) can be generated from the code \( C \) by the following algorithm. To start with, let
\[ S_1 \overset{\Delta}{=} \mathcal{D}_{CC} . \]
\( S_1 = \emptyset \) iff \( C \) is a prefix code, in which case, set \( S = \emptyset \). If \( S_1 \neq \emptyset \), define sequentially, starting with \( k = 1 \) :
\[ S_{k+1} = S_k \cup \mathcal{D}_{S_kC} \cup \mathcal{D}_{CS_k} , \]
stopping at the first \( k \) for which \( S_{k+1} = S_k \). This algorithm must terminate in finitely many steps. Set \( S = S_k \) for the \( k \) at which the algorithm terminates.

(b) Let \( C \) be comprised of \( M \) codewords of lengths \( l_1, \ldots, l_M \), as indicated. A simple bound on the size of \( S \) may be arrived at by reasoning as follows : each string in \( S \) is a suffix of one of the codewords. The total number of distinct suffixes of a string of length \( l \) is \( l \). Hence
\[ |S| \leq \sum_{m=1}^{M} l_m . \]
Better bounds are also welcome.
(c) 
  i. This is a prefix code, hence $\mathcal{S} = \emptyset$ and the code is uniquely decodable.
  ii. Here $\mathcal{S} = \{1\}$, which is disjoint from $\mathcal{C}$, so this code is uniquely decodable.
  iii. Here $\mathcal{S} = \{0, 1\}$ and $0 \in \mathcal{C}$ so this code is not uniquely decodable. An example of a string that is not uniquely decodable is 010 which can be parsed either as 0, 10 or as 01, 0.
  iv. Here $\mathcal{S} = \{1\}$ which is disjoint from $\mathcal{C}$, so the code is uniquely decodable.
  v. This is a prefix code, hence $\mathcal{S} = \emptyset$ and the code is uniquely decodable.
  vi. Here $\mathcal{S} = \{0\}$, which is disjoint from $\mathcal{C}$, so the code is uniquely decodable.
  vii. Here $\mathcal{S} = \{0\}$, which is disjoint from $\mathcal{C}$, so the code is uniquely decodable.
In each of the uniquely decodable cases above, it is instructive to think through how one would uniquely parse a given finite string of bits that admits a parsing in terms of the codewords of the code.

(d) Any infinite string that has a parsing into the codewords of a prefix code has only one such parsing. If it had two such parsings, there would be a phrase in one of the parsings which has a phrase in the other parsing as a prefix, contradicting the prefix nature of the code.

Thus, to solve this problem we need only consider cases (ii), (iv), (vi), and (vii) from the cases above.

In case (ii) consider the string

$0111111\ldots$

i.e., 0 followed by an infinite string of 1’s. This has two distinct parsings: 0, 11, 11, 11, \ldots or 01, 11, 11, \ldots.

In case (iv) any infinite string that admits a parsing into phrases from the code must start with 0. If it is an infinite string of 0’s it has a unique parsing. If not, there is a first place where it is 1, and this is followed by a 0 and the portion of the string up to and including the first place where it is 1 has a unique parsing, and so one is left with parsing the rest of the string which starts with the 0 following the first 1. From this reasoning we conclude that if an infinite string has a parsing in terms of the codewords of this code, this parsing is unique.

In case (vi) any infinite string that admits a parsing into phrases from the code must start with 1. If it is an infinite string of 1’s it has a unique parsing. If not, there is a first place where it is 0, and this is followed by 1, and the portion of the string up to and including the first 0 has a unique parsing and one is then left with the problem of parsing the rest of the string which starts with the 1 following the first 0. From this reasoning we conclude that if an infinite string has a parsing in terms of the codewords of this code, this parsing is unique.

In case (vii) consider the infinite string

$100000000000\ldots$

i.e. 1 followed by an infinite sequence of 0’s. This has two distinct parsings: 10, 00, 00 \ldots and 100, 00, 00 \ldots.

Solution:

Counting

Let $\mathcal{X} = \{1, 2, \ldots, m\}$ as in the text. If $\alpha > \max_{x \in \mathcal{X}} g(x)$ there are no sequences $x^n \in \mathcal{X}^n$ satisfying $\frac{1}{n} \sum_{i=1}^{n} g(x_i) \geq \alpha$. Also $H^* = -\infty$, since as defined in the text

$$
H^* = \sup_{P : \sum_{x \in \mathcal{X}} P(x)g(x) \geq \alpha} H(P),
$$

and the supremum over an empty set is $-\infty$. If $\alpha = \max_{x \in \mathcal{X}} g(x)$, let $d = |\{x \in \mathcal{X} : g(x) = \alpha\}|$. The number of sequences $x^n \in \mathcal{X}^n$ satisfying $\frac{1}{n} \sum_{i=1}^{n} g(x_i) \geq \alpha$ is then $d^n$. Also, a probability distribution $P$ on $\mathcal{X}$ satisfies $\sum_{x \in \mathcal{X}} P(x)g(x) \geq \alpha$ iff $P(x) = 0$ for all $x$ such that $g(x) < \alpha$. Thus $H^* = \log d$. In both these cases, the claim is verified, so we may now assume that $\alpha < \max_{x \in \mathcal{X}} g(x)$.

$H(P)$ is a continuous strictly concave function of $P$ ranging over probability distributions on $\mathcal{X}$. Thus there is a unique probability distribution $P^*$ on $\mathcal{X}$ such that $H^* = H(P^*)$. By the continuity of $H(P)$ and the fact that the set $\{P : \sum_{x \in \mathcal{X}} P(x)g(x) \geq \alpha\}$ has nonempty interior, for any $\epsilon > 0$ for all sufficiently large $n$ there is some $P_n \in \mathcal{P}_n$ with $\sum_{x \in \mathcal{X}} P_n(x)g(x) \geq \alpha$ and $H(P_n) \geq H^* - \epsilon$. Thus, for all $n$ sufficiently large the total number of sequences $x^n \in \mathcal{X}^n$ satisfying $\frac{1}{n} \sum_{i=1}^{n} g(x_i) \geq \alpha$ is at least as large as the cardinality of the type class of $P_n$, i.e. at least as large as $\frac{1}{(n+1)^n} 2^{nH^*}$. Since the sequences $x^n \in \mathcal{X}^n$ satisfying $\frac{1}{n} \sum_{i=1}^{n} g(x_i) \geq \alpha$ are precisely those whose type lies in the set $\{P : \sum_{x \in \mathcal{X}} P(x)g(x) \geq \alpha\}$, the total number of such sequences is at most $(n + 1)^n 2^{nH^*}$. The claim is now proved, by first taking the limit as $n \to \infty$ and then taking the limit as $\epsilon \to 0$.


Solution:

Sanov’s theorem:

There are four bullets in the suggested scheme of proof. The problem is not strictly correct as stated. The summation on the right hand side of equation (11.326) should start at $\lfloor np \rfloor$ for the problem statement to be correct.

For the first bullet, observe that $(x_1, \ldots, x_n)$ satisfies $\bar{x}_n \geq p$ (where $\bar{x}_n$ is defined as $\frac{1}{n} \sum_{i=1}^{n} x_i$) iff the total number of 1’s in $(x_1, \ldots, x_n)$ is $\lfloor np \rfloor$ or bigger. This gives (the corrected version of) equation (11.326).

For the second bullet, consider the ratio of the $i + 1$-st to the $i$-th term in the summation in equation (11.326). It is $\frac{n-i+1}{n+i} \frac{1}{1-q}$. Since $\frac{n-i+1}{n+i} \leq \frac{1-p}{p}$ when $i \geq \lfloor np \rfloor$, this ratio is at most 1, and so the largest of the terms in this summation is the one corresponding to $i = \lfloor np \rfloor$.

For the third bullet, the proof comes as a special case of the general estimate in equation (11.54) on page 354 of the text. Take $Q$ to be the Binomial $(q)$ distribution and $P$ to be the Binomial $(\lfloor np \rfloor)$ distribution and apply this estimate. We get, for all $n$,

$$
\frac{1}{(n+1)^2} 2^{-nD} \leq \binom{n}{\lfloor np \rfloor} q^{\lfloor np \rfloor} (1-q)^{n-\lfloor np \rfloor} \leq 2^{-nD}.
$$
You can get a sharper estimate, if you like, by instead applying equation (11.40) on pg. 353 of the text.

For the last bullet, we get the upper bound by noting that the total number of terms in the sum in equation (11.326) is no more than \( n \). We get the lower bound by noting that the left hand side of equation (11.326) can be lower bounded by term \( \left\lfloor \frac{n}{n-p} \right\rfloor q^i (1-q)^{n-i} \) and using the same kind of estimate as in the third bullet above.

5. Problem 13.4 on pg. 458 of the text.

**Solution:**

**Arithmetic Coding**

(a) Under the stationary distribution the states 0 and 1 are equiprobable.

We can think of 0 as owning the interval \([0, \frac{1}{2})\) and 1 as owning the interval \([\frac{1}{2}, 1)\). Then 00 owns the interval \([0, \frac{1}{4})\), 01 owns the interval \([\frac{1}{4}, \frac{1}{2})\), 10 owns the interval \([\frac{1}{2}, \frac{3}{4})\), and 11 owns the interval \([\frac{3}{4}, 1)\). The general pattern is that any interval owned by a string whose last bit is 0 is split into two intervals, the left one being \(\frac{1}{3}\)-rd of its total length and the right one being \(\frac{2}{3}\)-rd of its total length, and these now belong to the extensions of the parent string by 0 and 1 respectively. Likewise, any interval owned by a string whose last bit is 1 is split into two intervals, the left one being \(\frac{2}{3}\)-rd of its total length and the right one being \(\frac{1}{3}\)-rd of its total length, and these now belong to the extensions of the parent string by 0 and 1 respectively.

To determine the interval owned by \((01110)\) we may proceed step by step through the intervals \([0, \frac{1}{2})\), \([\frac{1}{4}, \frac{1}{3})\), \([\frac{1}{8}, \frac{1}{4})\), \([\frac{5}{16}, \frac{7}{24})\), to \([\frac{75}{162}, \frac{79}{162})\), where we have renormalized the denominators on the fractions to make the steps transparent. In the notation of the textbook, we get

\[
F(.01110) = \frac{79}{162}.
\]

(b) The question should be interpreted from the point of view of the receiver. In arithmetic coding, the transmitter will transmit the bit string corresponding to smallest basic dyadic interval that contains the interval owned by the target string (recall that a basic dyadic interval is any interval of the form \([\frac{k}{2^m}, \frac{k+1}{2^m})\), where \(0 \leq k \leq 2^{-m} - 1\) and \(m \geq 1\); it can also be thought of as any interval owned by a binary string under the model that the individual bits are i.i.d. and equiprobable). Some calculation shows that the smallest basic dyadic interval that contains the interval \([\frac{75}{162}, \frac{79}{162})\) is \([\frac{7}{16}, \frac{9}{16})\). Hence, having seen \((01110)\) the transmitter would have output \((0111)\).

The receiver, on receiving a binary string, thinks of it as defining a basic dyadic interval and then looks for the smallest interval owned by one of the strings of the original source that contains this dyadic interval. The receiver can be sure that the string that owns this (containing) interval in the original source is a prefix of the original source sequence. In this example, that interval is seen to be \([\frac{7}{16}, \frac{9}{16})\), which is owned by the string \((011)\) in the original source. Thus the receiver can be sure of the first three bits of the binary fractional representation of the overall source string \((01110X_6X_7\ldots)\).
6. *Run length coding versus Huffman coding*

This is problem 3.17 of the book of Gallager.

A source produces a sequence of independent bits with \( p(0) = 0.9 \) and \( p(1) = 0.1 \). We will encode the sequence in two stages, first counting the number of zeros between successive ones in the source output and then encoding their run lengths into binary code words. The first stage of encoding maps source sequences into intermediate digits by the following rule:

<table>
<thead>
<tr>
<th>Source sequence</th>
<th>Intermediate digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
</tr>
<tr>
<td>0001</td>
<td>3</td>
</tr>
<tr>
<td>00001</td>
<td>4</td>
</tr>
<tr>
<td>000001</td>
<td>5</td>
</tr>
<tr>
<td>0000001</td>
<td>6</td>
</tr>
<tr>
<td>00000001</td>
<td>7</td>
</tr>
<tr>
<td>00000000</td>
<td>8</td>
</tr>
</tbody>
</table>

As an example to clarify the description consider the following sequence and make sure you understand how the intermediate encoding is created:

\[
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1
\]

(Imagine that every finite source sequence is terminated with a 1 before the first stage so that there is no problem at the end of the sequence; after decoding this 1 can be discarded.)

The final stage of the encoding assigns the bit string 1 to the intermediate digit 8 and the eight bit strings of length 4 that start with the bit 0 in some one to one way to the other intermediate digits 0, \ldots, 7.

(a) Argue that the overall code is uniquely decodable.

(b) Find the average number of source digits per intermediate digit.

(c) Find the average number of encoded bits per intermediate digit.

(d) Based on the preceding two parts and an appeal to the law of large numbers, determine the average number of bits per source symbol used by the overall code.

(e) Find a Huffman code for the source based on encoding four source digits at a time and find the average number of bits per symbol used by this Huffman code.

(f) Compare your answers in the preceding two parts. Did you find this surprising?

**Solution:**

(a) If the first received bit is 1, we know that the first intermediate digit is 8 and so the source sequence starts with a string of 8 zeros.
If the first received bit is 0, we look at the first four received bits and thereby
determine the first intermediate digit, which must be one of the digits from 0 through
7. From this we can determine the prefix of the source sequence.

This decoding procedure can now be repeated to eventually reconstruct the entire
source sequence.

(b) For each intermediate digit, the length and the probability of the source string that
it codes for are listed below:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.09</td>
<td>0.081</td>
<td>0.0729</td>
<td>0.06561</td>
<td>0.05905</td>
<td>0.05314</td>
<td>0.04783</td>
<td>0.43047</td>
</tr>
</tbody>
</table>

From this, the mean length of the source string coded for by an intermediate digit
is seen to be $10(1 - (0.9)^8) = 5.6953$.

(c) For each intermediate digit, its probability and the number of bits needed to encode
it are listed below:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.09</td>
<td>0.081</td>
<td>0.0729</td>
<td>0.06561</td>
<td>0.05905</td>
<td>0.05314</td>
<td>0.04783</td>
<td>0.43047</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

From this, the mean number of bits needed to encode an intermediate digit is seen
to be $4 - 3(0.9)^8 = 2.7086$.

(d) Consider an infinitely long sequence from the source. The sequence of intermediate
digits that results from the first stage of coding is an i.i.d. sequence from \(\{0, 1, \ldots, 8\}\)
with probability distribution given as in either of the tables above. Let \(L_1, L_2, \ldots\) be
the lengths of the source strings coded for by these intermediate digits and \(B_1, B_2, \ldots\)
be the lengths of the bit strings used to encode these intermediate digits. Each of
these respectively is an i.i.d. sequence of random variables, with respective means
5.6953 and 2.7086. The number of bits per source bit need to encoded the portion
of the source sequence that requires \(k\) intermediate digits to code is

$$
\frac{B_1 + \ldots + B_k}{L_1 + \ldots + L_k} = \frac{B_1 + \ldots + B_k}{L_1 + \ldots + L_k}.
$$

Letting \(k \to \infty\) and appealing to the law of large numbers, this converges to
\(\frac{2.7086}{5.6953} = 0.4756\) code bits per source bit.

(e) There are 16 source strings of length 4, one with probability 0.6561, four with
probability 0.0729, six with probability 0.081, four with probability 0.009, and one with
probability 0.0001. Listing these in lexicographic order and constructing a Huffman
code using the algorithm as described in class, with the lower branch of each
binary fork getting bit 0 and the upper branch getting bit 1 results in the code:
The mean length of this Huffman code is 1.9702 bits, i.e. 0.4926 code bits per source bit. 

(f) The Huffman code of the preceding part is not as good as the run length code described earlier. Whether this is surprising or not depends on you. The entropy of the source is 0.4690. Since the source is i.i.d., and the Huffman code is based on taking blocks of four source symbols, it is guaranteed to give a compression rate of between 0.4690 and 0.4690 + 0.25 = 0.7190 code bits per source bit, so it is actually doing pretty well relative to these bounds.


Solution:

Tunstall coding

The original source has alphabet $X = \{0, 1, \ldots, m - 1\}$. Since the strings defining any variable to fixed length code must close out all ways of escaping from the root in the $m$-ary tree, the size $D$ of the alphabet of the code must satisfy $D = 1 + k(m - 1)$ for some $k \geq 1$.

(a) Given a variable to fixed length code, let $(Y_1, Y_2, \ldots)$ denote the sequence of symbols from the $D$-ary alphabet that results from the source sequence $(X_1, X_2, \ldots)$. Note that $Y_1, Y_2, \ldots$ are independent and identically distributed. (To verify this formally actually requires some care. You need to argue that if $(Z_1, Z_2, \ldots)$ denote the successive source symbols after the first phrase in the source, then $(Z_1, Z_2, \ldots)$ has the same distribution as $(X_1, X_2, \ldots)$ and is independent of the first phrase of the source. You can do this by conditioning on all possible realizations of the first phrase and noticing that the conditioning does not matter.)

Since $Y_1$ takes values in an alphabet of size $D$, we have $H(Y_1) \leq \log D$. We will argue that every variable to fixed length code has the property $H(Y_1) = H(X)E[L(A_D)]$, completing the proof. We argue iteratively, following the construction of any variable
to fixed length code. The basic variable to fixed length code has \( D = m \), mean length 1 and \( H(Y_1) \), being the same as the entropy of the distribution on the phrases, equals \( H(X) \), because each phrase in the basic variable to fixed length code is just a symbol from the source. Suppose now that we have shown the claim for some variable to fixed length code and we grow one of its leaves, which has probability say \( p \). The mean length of the code then increases by \( p \). Also, the entropy of the phrase distribution increases by \( pH(X) \). Thus the property holds for the grown variable to fixed length code. This verifies that the property holds for all variable to fixed length codes, as desired.

(b) Let \( D = 1 + k(m - 1) \) be fixed. Consider an optimal variable to fixed length code for a source with alphabet of size \( m \), having \( D \) leaves. It suffices to show that in the \( m \)-ary tree, at every depth \( l \geq 1 \), any node that is split has higher probability than any node at that depth that is not split. Suppose this is not the case. Then there is a node at depth \( l \) with probability \( p \) that is not split and another node at depth \( l \) with probability \( q < p \) that is split. We may move the entire subtree that has the latter node as its root to root it at the former node, getting another variable to fixed length code that has strictly larger mean length than the one we started with (in the new code the former node will be split and the latter node will not be split). This contradicts the optimality of the given code, completing the proof.

(c) We can in fact show that for every \( \epsilon > 0 \) there exists \( D \) with \( R(A^*_D) < H(X) + 2\epsilon \).

Given \( \delta > 0 \), let \( n \) be large enough that the set of \( \epsilon \)-weakly typical strings of length \( n \) from the source has probability at least \( 1 - \delta \). In the \( m \)-ary tree, start out by marking with the color red each node at depth \( n \) that corresponds to an \( \epsilon \)-weakly typical sequence. There is a total of at most \( 2^n(H(X) + \epsilon) \) such nodes. Next, for each path from the root in the \( m \)-ary tree to any node at depth \( n \) that is not already marked red, mark as blue the node that is the first node encountered on path that does not lie on a path to a node marked red. The blue and the red nodes together determine the phrases of a variable to fixed length code, because they define a prefix free set of strings that closes out all paths from the root. Further, the total number of blue nodes is at most \((m - 1)n + 1\) times the total number of red nodes; this can be seen by noting that for every blue node there is some red node such that on the path from the root to that red node there is a branch of a single step into that blue node. This gives the upper bound

\[
D \leq ((m - 1)n + 1)2^n(H(X) + \epsilon) \leq mn2^n(H(X) + \epsilon) ,
\]

so we have \( \log D \leq \log m + \log n + n(H(X) + \epsilon) \). The mean length of this variable to fixed length code is at least \( n(1 - \delta) \). For this \( D \) (whatever it might be), we therefore get

\[
R(A^*_D) \leq \frac{\log m + \log n + n(H(X) + \epsilon)}{n(1 - \delta)} .
\]

If \( \delta > 0 \) is chosen small enough and \( n \) is sufficiently large, the right hand side is bounded above by \( H(X) + 2\epsilon \), completing the proof.