CS 70 Discrete Mathematics and Probability Theory
Spring 2019 Course Notes

HW 6

1 Error-Detecting Codes

Suppose Alice wants to transmit a message of n symbols, so that Bob is able to *detect* rather than *correct* any errors that have occured on the way. That is, Alice wants to find an encoding so that Bob, upon receiving the code, is able to either

(I) tell that there are no errors and decode the message, or

(II) realize that the transmitted code contains at least one error, and throw away the message.

Assuming that we are guaranteed a maximum of k errors, how should Alice extend her message (i.e. by how many symbols should she extend the message, and how should she choose these symbols)? You may assume that we work in GF(p) for very large prime p. Show that your scheme works, and that adding any lesser number of symbols is not good enough.

Solution:

Since k bits can break, it seems reasonable to extend our message by k symbols for a total of $n+k$. And indeed, we show that this works: Let Alice generate her message $y_0, \dots y_{n-1}$ of length n by constructing the unique polynomial f of degree $\leq n-1$ that passes through (i, y_i) for $i \in \{0, \dots, n-1\}$, and add the k extra symbols $y_j = f(j)$, where $j \in \{n, \dots, n+k-1\}$. Now Bob receives the message $r_i, i \in \{0, \dots, n+k-1\}$, upon which he interpolates the unique degree $\leq n-1$ polynomial g that passes through the points $(0, r_0), \dots, (n-1, r_{n-1})$. We claim that the message is corrupted if and only if $g(i) \neq r_i$ for some $i \in \{n, \dots, n+k-1\}$.

The backward direction becomes clear when stated as its contrapositive: If the message contains no error, then $g(i)$ and $f(i)$ coincide on all of n points $\{0, \dots, n-1\}$. Since they are both of degree $n-1$, they must be the same polynomial and hence $g(i) = f(i) = r_i$ for all i.

Let us prove the forward direction: Since we know that at most k errors occured, there must exist a subset $A \subset \{0, \dots, n+k-1\}$ of size n on which $r_i = y_i$. Now either

1. $A = \{0, \dots, n-1\}$, in which case $g = f$ and at least one error must have occured for some $j_0 \in \{n, \dots, n+k-1\}$. But then $r_{j_0} \neq y_{j_0} = f(j_0) = g(j_0)$, which is what we wanted to show.
2. Or at least one error occured for an index $i \in \{0, \dots, n-1\}$ in which case $g \neq f$. But since g and f are of degree $n-1$ and $| A | = n$, f and g cannot take the same values on A, so there must be some element $j_0 \in A, j_0 \in \{n, \dots, n+k-1\}$ for which $g(j_0) \neq f(j_0) = y_{j_0} = r_{j_0}$.

Lastly, we need to show that our algorithm doesn't work if Alice extends her message by less than k symbols, which we can do by crafting a counterexample: Assume Alice sends $m < n+k-1$

symbols in the same fashion as above, then we may corrupt $y_{n-1}, \ldots y_{m-1}$ by setting $r_{n-1} \neq y_{n-1}$ and $r_j = h(j)$ for $j \in \{n, \ldots, m-1\}$, where h is the unique polynomial of degree $\leq n-1$ passing through $(0, y_0), \ldots (n-2, y_{n-2}), (n-1, r_{n-1})$. Since Bob is going to reconstruct $g = h$, $g(j) = r_j$ for all $j \in \{n, \ldots, m-1\}$ and he will not notice the corruption.

2 Berlekamp-Welch Algorithm with Fewer Errors

In class we derived how the Berlekamp-Welch algorithm can be used to correct k general errors, given $n+2k$ points transmitted. In real life, it is usually difficult to determine the number of errors that will occur. What if we have less than k errors? This is a follow up to the exercise posed in the notes.

Suppose Alice wants to send 1 message to Bob and wants to guard against 1 general error. She decides to encode the message with $P(x) = 4$ (on GF(7)) such that $P(0) = 4$ is the message she want to send. She then sends $P(0), P(1), P(2) = (4, 4, 4)$ to Bob.

(a) Suppose Bob receives the message $(4, 5, 4)$. Without performing Gaussian elimination explicitly, find $E(x)$ and $Q(x)$.

(b) Now, suppose there were no general errors and Bob receives the original message $(4, 4, 4)$. Show that the $Q(x), E(x)$ that you found in part (a) still satisfies $Q(i) = r_i E(i)$ for all $i = 0, 1, 2$.

(c) Verify that $E(x) = x$, $Q(x) = 4x$ is another possible set of polynomials that satisfies $Q(i) = r_i E(i)$ for all $i = 0, 1, 2$.

(d) Suppose you're actually trying to decode the received message $(4, 4, 4)$. Based on what you showed in the previous two parts, what will happen during row reduction when you try to solve for the unknowns?

(e) Prove that no matter what the solution of $Q(x)$ and $E(x)$ are though, the recovered $P(x)$ will always be the same.

Solution:

(a) $E(x) = x - 1$ and $Q(x) = P(x)E(x) = 4x - 4$.

(b) This is true because there were no errors, so $P(i) = r_i$ for $i = 0, 1, 2$.

(c) Since $Q(x) = P(x)E(x)$ and $P(i) = r_i$ for $i = 0, 1, 2$, we must have $Q(i) = r_i E(i)$ for all $i = 0, 1, 2$.

(d) There are multiple solutions to the system of equations.

(e) Suppose we got two solutions $Q'(x), E'(x)$ and $Q(x), E(x)$. Since they are both solutions, by definition, we have $Q'(i) = r_i E'(i)$ and $Q(i) = r_i E(i)$ for $1 \leq i \leq n+2k$. Therefore, $Q'(i)E(i) =$

$Q(i)E'(i) = r_i E(i)E'(i)$. However, $Q'(x)E(x) - Q(x)E'(x)$ is a degree $n+2k-1$ polynomial, which is 0 at $n+2k$ points. Thus, $Q'(x)E(x) = Q(x)E'(x)$ for all x, so we arrive at

$$\frac{Q'(x)}{E'(x)} = \frac{Q(x)}{E(x)}.$$

This proves that the final solution for $P(x)$ is the same.

3 Counting Cartesian Products

For two sets A and B, define the cartesian product as $A \times B = \{(a,b) : a \in A, b \in B\}$.

(a) Given two countable sets A and B, prove that $A \times B$ is countable.

(b) Given a finite number of countable sets $A_1, A_2, \ldots, A_n$, prove that

$$A_1 \times A_2 \times \cdots \times A_n$$

is countable.

(c) Consider an infinite number of countable sets: $B_1, B_2, \ldots$. Under what condition(s) is $B_1 \times B_2 \times \cdots$ countable? Prove that if this condition is violated, $B_1 \times B_2 \times \cdots$ is uncountable.

Solution:

(a) As shown in lecture, $\mathbb{N} \times \mathbb{N}$ is countable by creating a zigzag map that enumerates through the pairs: $(0,0), (1,0), (0,1), (2,0), (1,1), \ldots$. Since A and B are both countable, there exists a bijection between each set and a subset of $\mathbb{N}$. Thus we know that $A \times B$ is countable because there is a bijection between a subset of $\mathbb{N} \times \mathbb{N}$ and $A \times B : f(i,j) = (A_i, B_j)$. We can enumerate the pairs (a,b) similarly.

(b) Proceed by induction.
Base Case: $n = 2$. We showed in part (a) that $A_1 \times A_2$ is countable since both A_1 and A_2 are countable.
Induction Hypothesis: Assume that for some $n \in \mathbb{N}$, $A_1 \times A_2 \times \cdots \times A_n$ is countable.
Induction Step: Consider $A_1 \times \cdots \times A_n \times A_{n+1}$. We know from our hypothesis that $A_1 \times \cdots \times A_n$ is countable, call it $C = A_1 \times \cdots \times A_n$. We proved in part (a) that since C is countable and A_{n+1} are countable, $C \times A_{n+1}$ is countable, which proves our claim.

(c) If any of the B_i are empty, then the infinite Cartesian Product is also empty. Hence, we assume that none of the B_i are empty.

A necessary and sufficient condition for the infinite Cartesian Product to be countable is for all but finitely many of the B_i to have exactly one element.

Necessary: Suppose not; then, infinitely many B_i have at least two elements. Notice that for any B_i with only one element, every element of the infinite Cartesian Product must necessarily

use the single element of B_i. Therefore, we can ignore each B_i with only one element and assume that every B_i has at least two elements. Proceed with a diagonalization argument by assuming for the sake of contradiction that $B_1 \times B_2 \times \cdots$ is countable and its elements can be enumerated in a list:

$$\begin{aligned}
&(b_{1,1}, b_{2,1}, b_{3,1}, b_{4,1}, \ldots)\\
&(b_{1,2}, b_{2,2}, b_{3,2}, b_{4,2}, \ldots)\\
&(b_{1,3}, b_{2,3}, b_{3,3}, b_{4,3}, \ldots)\\
&(b_{1,4}, b_{2,4}, b_{3,4}, b_{4,4}, \ldots)\\
&\vdots
\end{aligned}$$

where $b_{i,j}$ represents the item from set B_i that is included in the jth element of the Cartesian Product. Now consider the element $(\overline{b_{1,1}}, \overline{b_{2,2}}, \overline{b_{3,3}}, \overline{b_{4,4}}, \ldots)$, where $\overline{b_{i,j}}$ represents any item from set B_i that differs from $b_{i,j}$ (i.e. any other element in the set). This is a valid element that should exist in the Cartesian Product $B_1, B_2, \ldots$, yet it is not in the enumerated list. This is a contradiction, so $B_1, B_2, \ldots$ must be uncountable.

Notice that this relies on the fact that each set B_i has some other item we can choose when constructing our diagonal element.

Sufficient: If all but finitely many of the B_i have one element, then there are only a finite number of B_i which have more than one element. Again, we can ignore the B_i with only one element. By the previous part, we showed that the Cartesian Product of a finite number of countable sets is countable, so the infinite Cartesian Product in this case is countable.

4 Counting Tools

Are the following sets countable or uncountable? Please prove your claims.

(a) $\bigcup_{i \in A} B_i$, where A, B_i are all countable.

(b) The set of all functions f from $\mathbb{N}$ to $\mathbb{N}$ such that f is non-decreasing. That is, $f(x) \leq f(y)$ whenever $x \leq y$.

(c) The set of all functions f from $\mathbb{N}$ to $\mathbb{N}$ such that f is non-increasing. That is, $f(x) \geq f(y)$ whenever $x \leq y$.

(d) The set of all bijective functions from $\mathbb{N}$ to $\mathbb{N}$.

Solution:

(a) Countable: Since A and the B_i are countable, we can enumerate the elements of each set. Let us call $b_{i,j}$ the j^{th} element in B_i, then $U = \bigcup_{i \in A} B_i = \bigcup_{i,j \in \mathbb{N}} b_{i,j}$ and so U has at most as many elements as $\bigcup_{i,j \in \mathbb{N}} (i, j) = \mathbb{N} \times \mathbb{N}$.

(b) Uncountable: Let us assume the contrary and proceed with a diagonal argument. If there are countably many such function we can enumerate them as

	0	1	2	3	…
f_0	$f_0(0)$	$f_0(1)$	$f_0(2)$	$f_0(3)$	…
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$f_1(3)$	…
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)$	$f_2(3)$	…
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	…
⋮	⋮	⋮	⋮	⋮	⋱

Now go along the diagonal and define f such that $f(x) > f_x(x)$ and $f(y) > f(x)$ if $y > x$, which is possible because at step k we only need to find a number $\in \mathbb{N}$ greater than all the $f_j(j)$ for $j \in \{0,\dots,k\}$. This function differs from each f_i and therefore cannot be on the list, hence the list does not exhaust all non-decreasing functions. As a result, there must be uncountable many such functions.

Alternative Solution: Look at the subset $\mathscr{S}$ of strictly increasing functions. Any such f is uniquely identified by its image which is an infinite subset of $\mathbb{N}$. But the set of infinite subsets of $\mathbb{N}$ is uncountable, so $\mathscr{S}$ is uncountable and hence the set of all non-decreasing functions must be too.

Alternative Solution 2: We can inject the set of infinitely long binary strings into the set of non-decreasing functions as follows. Let $b(i)$ be the ith digit of the string b. Now, map $b(i)$ to the subset of non-decreasing functions with the following property: for all $i \in \mathbb{N}$, $b(i) = 1 \implies f(i) > f(i-1)$, and $b(i) = 0 \implies f(i) = f(i-1)$. In this way, every infinite binary string gets paired with some set of functions. For example, the string $11111\dots$ is the set of strictly increasing functions. Because we showed the set of infinite binary strings is uncountable, and we produced an injection from that set to the set of non-decreasing functions, that set must be uncountable as well.

(c) Countable: Let D_n be the subset of non-increasing functions for which $f(0) = n$. Any such function must stop decreasing at some point (because $\mathbb{N}$ has a smallest number), so there can only be finitely many (at most n) points $X_f = \{x_1,\dots,x_k\}$ at which f decreases. Let y_i be the amount by which f decreases at x_i, then f is fully described by $\{(x_1,y_1),\dots,(x_k,y_k),$ $(-1,0),\dots,(-1,0)\} \in \mathbb{N}^n = \mathbb{N}\times\mathbb{N}\times\cdots\times\mathbb{N}$ (n times), where we padded the k values associated with f with $n-k$ $(-1,0)$s. But $\mathbb{N}^n$ is countable by repeated application of part (a), and hence D_n is countable. Now the set of all non-increasing functions is $\bigcup_{i\in\mathbb{N}} D_n$, which is the countable union of countable sets, and thus countable by part (a).

(d) Uncountable: We will show that the set of bijections of $\mathbb{N}$ is at least as large as the powerset $\mathscr{P}(\mathbb{N})$ of $\mathbb{N}$, which we know to be uncountable. To do so, we need a little lemma:

Lemma (Shufflability of subsets): If A is a subset of $\mathbb{N}$ and $|A| > 1$, then we can find a bijection $h : A \to A$, so that for all $x \in A$, $h(x) \neq x$. That is, h maps every element to an element *other than itself*.

Proof: If $|A| = 1 < n < \infty$, then we can write $A = \{a_1,\dots,a_n\}$ and define $h(a_i) = a_{i+1 \bmod n}$ and are done. If $|A| = \infty$, then we write $A = \{a_1,a_2,\dots\}$ and define h to swap any two consecutive

elements, i.e. $h(a_1) = a_2, h(a_2) = a_1, h(a_3) = a_4, h(a_4) = a_3$, etc.

Now we are in shape to associate with each subset S of $\mathbb{N}$ (ignoring subsets that are of the form $\mathbb{N} \setminus \{x\}$, which we will take care of later), a bijection g_S of $\mathbb{N}$: Namely, let us define g_S so that for all $x \in S$, $g_S(x) = x$, and on $\mathbb{N} \setminus S$, we let g_S be any function h_S from the lemma above. All we need to prove is that g_S and $g_{S'}$ are distinct for distinct S and S'. But if $S \neq S'$, then without loss of generality there exists some $s \in S \setminus S'$. For this s, we have $g_S(s) = s \neq g_{S'}(s)$ and so g_S and $g_{S'}$ must be different. Now, we have constructed an injection that maps the power set $\mathscr{P}(\mathbb{N})$ to a subset of bijective functions on $\mathbb{N}$, except for the special subsets of the form $\mathbb{N} \setminus \{x\}$ for some number x. The reason we excluded these sets is because then we would have to apply the shuffability lemma to the singleton $\{x\}$, which is not possible. Does this break our proof? No! The number of sets that we have ignored is countable, so the *remaining* subset of the power set that we have mapped into bijective functions *is still uncountable*, and thus the set of bijective functions from $\mathbb{N} \to \mathbb{N}$ is uncountable.

5 Fixed Points

Consider the problem of determining if a function F has any fixed points; that is, we want to know if there is any input x such that $F(x)$ outputs x. Prove that this problem is undecidable.

Solution:

We can prove this by reducing from the Halting Problem. Suppose we had some function `FixedPoint`(F) that solved the fixed-point problem. We can define `TestHalt`(F,x) as follows:

```
def TestHalt(F, x):
  def F_prime(y):
    F(x)
    return y
  return FixedPoint(F_prime)
```

If $F(x)$ halts, we have that $F'(y)$ will always just return y, so every input is a fixed point. On the other hand, if $F(x)$ does not halt, F' won't return anything for any input y, so there can't be any fixed points. Thus, our definition of `TestHalt` must always work, which is a contradiction; this tells us that `FixedPoint` cannot exist.

6 Kolmogorov Complexity

Compression of a bit string x of length n involves creating a program shorter than n bits that returns x. The Kolmogorov complexity of a string $K(x)$ is the length of shortest program that returns x (i.e. the length of a maximally compressed version of x).

(a) Explain why "the smallest positive integer not definable in under 100 characters" is paradoxical.

(b) Prove that for any length n, there must be at least one bit string that cannot be compressed.

(c) Imagine you had the program K, which outputs the Kolmogorov complexity of string. Design a program P that when given integer n outputs the bit string of length n with the highest Kolmogorov complexity. If there are multiple strings with the highest complexity, output the lexicographically first (i.e. the one that would come first in a dictionary).

(d) Suppose the program P you just wrote can be written in m bits. Show that P and by extension, K, cannot exist, for a sufficiently large input n.

Solution:

(a) Since there are only a finite number of characters then there are only a finite number of positive integers that can be defined in under 100 characters. Therefore there must be positive integers that are not definable in 100 characters and by the well-ordering principle there is a smallest member of that set. However the statement "the smallest positive integer not definable in under 100 characters" defines the smallest such an integer using only 67 characters (including spaces). Hence, we have a paradox (called the Berry Paradox).

(b) The number of strings of length n is 2^n. The number of strings shorter than length n is $\sum_{i=0}^{n-1} 2^i$. We know that sum is equal to $2^n - 1$ (remember how binary works). Therefore the cardinality of the set of strings shorter than n is smaller than the cardinality of strings of length n. Therefore there must be strings of length n that cannot be compressed to shorter strings.

(c) We write such a program as follows:

```
def P(n):
  complex_string = "0" * n
  for j in range(1, 2 ** n):
    # some fancy Python to convert j into binary
    bit_string = "0:b".format(j)
    # length should now be n characters
    bit_string = (n - len(bit_string)) * "0" + bit_string
    if K(bit_string) > K(complex_string):
      complex_string = bit_string
   return complex_string
```

(d) We know that for every value of n there must be an incompressible string. Such an incompressible string would have a Kolmogorov complexity greater than or equal to its actual length. Therefore our program P must return an incompressible string. However, suppose we choose size n_k such that $n_k \gg m$. Our program $P(n_k)$ will output a string x of length n_k that is not compressible meaning $K(x) \geq n_k$. However we have designed a program that outputs x using fewer bits than n_k. This is a contradiction. Therefore K cannot exist.