CS 70 Discrete Mathematics and Probability Theory
Fall 2012 Vazirani Section 7

1 Counting practice

You can leave your answers as (tidy) expressions involving factorials, binomial coefficients, etc., rather than evaluating them as decimal numbers.

1. How many different 13-card bridge hands contain exactly 4 spades? [Recall that there are 13 spades in a standard 52-card deck. A bridge hand is obtained by selecting 13 cards from a standard 52-card deck. The order of the cards within a bridge hand is considered irrelevant.]

   [*Solution*] We need our hand to contain 4 out of the 13 spade cards, and 9 out of the 39 non-spade cards, and these choices can be made separately. Hence, there are $\binom{13}{4}\binom{39}{9}$ ways to make up the hand.

2. How many ways are there to order a standard 52-card deck?

   [*Solution*] The first position of the ordering has 52 choices for the card, the second position has 51 choices, and so on, until the last position where there is only one choice. Thus, there are 52! ways to order the deck.

3. How many different anagrams of "ALABAMA" are there? (An anagram of "ALABAMA" is any re-ordering of the letters of "ALABAMA"; i.e., any string made up of the eight letters A, L, A, B, A, M, and A, in any order. The anagram does not have to be an English word.)

   [*Solution*] In this 7 letter word, the letter A is repeated 4 times while the other letters appear once. Hence, the number 7! overcounts the number of different anagrams by a factor of 4! (which is the number of ways of permuting the 4 A's among themselves). Hence, there are $\frac{7!}{4!}$ anagrams.

4. Suppose you are given 8 indistinguishable balls and 5 bins. How many distinguishable ways are there to distribute these 8 balls among these 5 bins such that no bin is empty? Assume the bins are distinguishable (e.g., numbered 1 through 5).

   [*Solution*] Since each bin is required to be non-empty, let's throw one ball into each bin at the outset. Now we have 3 identical balls left which we want to throw into 8 distinguishable bins. The number of ways to put k identical balls into n distinguishable bins is $\binom{n+k-1}{k}$. Taking $k = 3$ and $n = 8$, we get $\binom{10}{3}$ ways to do this.

5. How many different ways are there to throw these 8 identical balls into 6 bins?

   [*Solution*] Since there is no restriction on how many balls a bin needs to have, this is just the problem of throwing k identical balls into n distinguishable bins, which can be done in $\binom{n+k-1}{k}$ ways. Here $k = 8$ and $n = 6$, so there are $\binom{13}{8}$ ways.

2 Hypercube routing

Recall that an n-dimensional hypercube contains 2^n vertices, each labeled with a distinct n bit string, and two vertices are adjacent iff their bit strings differ in exactly one position.

1. The hypercube is a popular architecture for parallel computation.Let each vertex of the hypercube represent a processor and each edge represent a communication link. Suppose we want to send a packet for vertex x to vertex y. Consider the following "bit-fixing" algorithm:

   > In each step, the current processor compares its address to the destination address of the packet. Let's say that the two addresses match up to the first k positions. The processor then forwards the packet and the destination address on to its neighboring processor whose address matches the destination address in at least the first $k+1$ positions. This process continues until the packet arrives at its destination.

   Consider the following example where $n = 4$: Suppose that the source vertex is (1001) and the destination vertex is (0100). Give the sequence of processors that the packet is forwarded to using the bit-fixing algorithm.

   [*Solution*] $1001 \to 0001 \to 0101 \to 0100$

2. In general, for an arbitrary source vertex and arbitrary destination vertex, how many edges must the packet traverse under this algorithm? Give an exact answer in terms of the n-bit strings labeling source and destination vertices. You *may* want to use the definition from **??**.

   [*Solution*] Each edge traversal corresponds to a separate bit position in which the two n-bit strings differ. Therefore, the number of edge traversals will be exactly the number of position at which the two strings differ.

   You can express this by looking at the bit-wise exclusive-or of the two strings: for 2 n-bit strings a and b, $c = a \oplus b$ is another n-bit string with $c_i = a_i \oplus b_i$, which is 1 when $a_i \neq b_i$ and 0 when $a_i = b_i$. The *number* of positions at which a and b differ is just the number of ones in $a \oplus b$, that is, $|a \oplus b|_1$.

3. Consider any two vertices x and y in the hypercube. Consider the graph $G = (V_{x,y}, E_{x,y})$ where $E_{x,y}$ is the set of all edges on shortest paths between x and y and $V_{x,y}$ is the set of all vertices on shortest paths between x and y.

   Consider the following example where $n = 3$: Suppose that x is (010) and y is (100). What is the length of the shortest path between x and y? Explicitly show the sets $V_{x,y}$ and $E_{x,y}$.

   [*Solution*] Since each edge changes just 1 bit, you need at least as many edges to get from x to y as there are differing bit positions between x and y. Since x and y differ in 2 positions, the length of the shortest path is at least two. Part (b) guarantees that paths of length 2 exist. One possible path is $010 \to 110 \to 100$, which is indeed of length 2, and thus is a shortest path.

   The only other way to get from x to y in 2 steps is $010 \to 000 \to 100$, so $V_{x,y} = \{010, 000, 110, 100\}$ and $E_{x,y} = \{\{010, 000\}, \{010, 110\}, \{000, 100\}, \{110, 100\}\}$ (as shown in Figure 1.

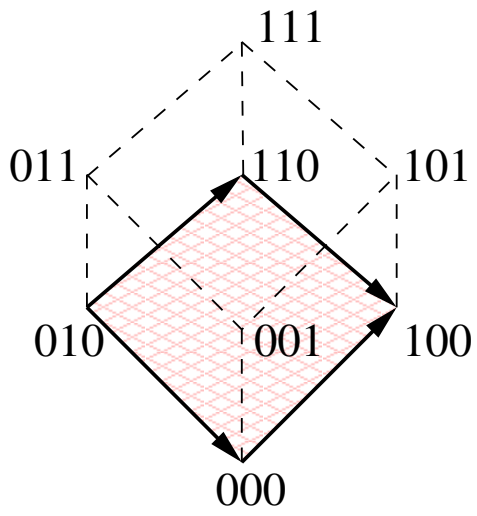
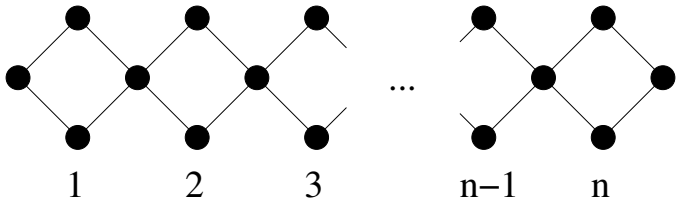


Figure 1: The possible shortest paths from 010 to 100. The subhypercube defined by all (both) such shortest paths is shaded, and $E_{x,y}$ is shown with solid arrows.

3 Chains

Consider "chain" graphs, shaped like this (with n "links"):



How many different Eulerian tours, starting and ending at the leftmost vertex, does a chain graph with n links have? (Hint: Argue that there must be the first vertex after which an Eulerian tour takes an edge 'in the left direction'. There are 4 types of vertices in the graph, and only one them can be the vertex.)

[*Solution*] Consider an Eulerian tour starting with the leftmost vertex. Intuitively, any such tour will have to start by moving rightward (taking, for every "chain link", either the top two edges or the bottom two edges arbitrarily), and will have to keep going until it gets to the rightmost node, at which point it will turn around and go back using whichever half of each link it didn't use before.

To prove this formally, let v be the first vertex after which an Eulerian tour takes an edge "in the left direction". This has to happen at some point, since we must leave the leftmost vertex, so at some point we'll have to move left again to get back to it. Now, v must be the rightmost vertex, because any of the other 3 cases yields a contradiction:

1. We must've arrived to v from the left; thus, v cannot be in the top or bottom rows of vertices, since there's coming in from the left and leaving to the left would require using the same edge (Eulerian tours can't do this).

2. If v is one of the degree-4 vertices, there's another problem — since we've been moving rightward up until v, we couldn't have visited any of the edges to the right of v thus far. And, since we must enter v on one of the edges to its left, and leave along the other one, we'll end up the left of v after that, with no unused edges left that would let us go back and hit the edges to the right of v.

3. Of course v can't be the leftmost vertex, since there's noway to go to the left.

Thus, the tour *must* start by first taking some sequence of edges exclusively towards the right and ending up at the rightmost vertex. From the structure of the graph, this allows 2 choices at each of the n non-rightmost vertices in the middle row, and 1 choice at each top or bottom vertex. Then, once you've gotten to the rightmost vertex, there'll be exactly one way to finish the tour, by taking all the unused edges in order on the way back.

Therefore, there are a total of 2^n Eulerian tours originating at the leftmost vertex.

4 Error-correcting codes

In this question we will go through an example of error-correcting codes with general errors. We will send a message (m_0, m_1, m_2) of length $n = 3$. We will use an error-correcting code for $k = 1$ general error, doing arithmetic modulo 5.

(a) Suppose $(m_0, m_1, m_2) = (4, 3, 2)$. Use Lagrange interpolation to construct a polynomial $P(x)$ of degree 2 (remember all arithmetic is mod5) so that $(P(0), P(1), P(2)) = (m_0, m_1, m_2)$. Then extend the message to lengeth $n+2k$ by appending $P(3), P(4)$. What is the polynomial $P(x)$ and what is the message $(c_0, c_1, c_2, c_3, c_4) = (P(0), P(1), P(2), P(3), P(4))$ that is sent?

(b) Suppose the message is corrupted by changing c_0 to 0. We will locate the error using the Berlekamp–Welsh method. Let $E(x) = x + b_0$ be the error-locator polynomial, and $Q(x) = P(x)E(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ be a polynomial with unknown coefficients. Write down the system of linear equations (involving unknowns a_0, a_1, a_2, a_3, b_0) in the Berlekamp–Welsh method. You need not solve the equations.

(c) The solution to the equations in part (b) is $b_0 = 0, a_0 = 0, a_1 = 4, a_2 = 4, a_3 = 0$. Show how the recipient can recover the original message (m_0, m_1, m_2).