We will consider a communication problem and provide both randomized and deterministic protocols to solve it. The problem is the following. Two parties, one on the moon and the other on earth, have each a string of n bits, say x and y respectively, and they want to determine whether $x = y$ or not. They would also like to minize the communication cost, which is defined as the total number of bits communicated back and forth. We will give a simple randomized protocol achieving $O(\log n)$ cost, and we will show that any deterministic algorithm requires $\Omega(n)$ bits to be transmitted.

Randomized Protocol

We first recall a fact from number theory that we will make use of. Let $\pi(m)$ be the number of primes less than or equal to m, then there exist constants $a < 1$ and $b > 1$ such that $a\frac{m}{\log m} \leq \pi(m) \leq b\frac{m}{\log m}$.

The protocol is rather simple. First, the party on earth picks a prime number p at random between 2 and $cn \log n$ for some suitable fixed constant c. Then, it transmits the pair $(p, y \bmod p)$ to the moon. Once the party on the moon receives the message, it checks to see whether $x = y \bmod p$. If they are not equal he knows that $x \neq y$. If $x = y \bmod p$ he may conclude that it is likely that $x = y$.

We want to show that if $x \neq y$ then it is very likely that $y \neq x \bmod p$. In other words we would like to prove that if $x \neq y$, then $y \neq x \bmod p$ with positive probability, say $\frac{1}{2}$.

Proof: Suppose $x \neq y$. Then $|x - y| \leq 2^n$. Therefore at most n primes divide $|x - y|$. But we have selected p uniformly among at least $a\frac{cn \log n}{\log(cn \log n)} \geq \frac{9}{10}acn \geq 2n$ primes. Therefore the fraction of primes on which we fail to detect that $x \neq y$ is less than or equal to $\frac{1}{2}$.

The total number of bits required in this exchange is $O(\log n)$. Suppose the parties repeat this process t times, and they declare x and y to be equal if none of the trials proved them to be different. Then, this will be a one-sided error randomized protocol for which the probability of error is 0 if $x = y$, and 2^{-t} if $x \neq y$. The protocol has a cost of $O(t \log n)$.

Deterministic Protocols

In this section we show that every deterministic protocol for the problem above must require $\Omega(n)$ transmissions.

Proof: The protocols are designed for a fixed value of n. For a given protocol Π consider a table T whose rows are indexed by the possible values of x and the columns by the possible values of y. Since both x and y are in $\{0,1\}^n$ there are 2^n rows and columns in T. For every $1 \leq i, j \leq 2^n$, let $T_{i,j} = 1$ if $i = j$ and 0 otherwise.

As both parties carry out protocol Π, at any moment the entries in T that represent the possible values of x and y given the exchange of information carried so far can be partitioned into combinatorial rectangles, that is the entries need not be adjacent (figure 1). The number of rectangles increases at most by a factor of two after each transmission.

Let's consider the entries in the diagonal of T, the 1 entries, at the end of the protocol, when at least one of the parties knows whether the two strings are the same or different. Each one

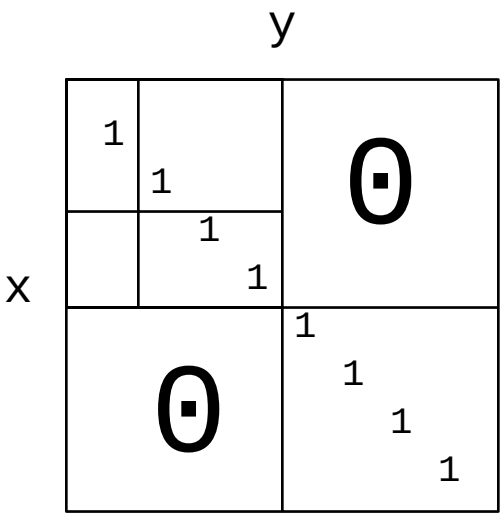

Figure 1:

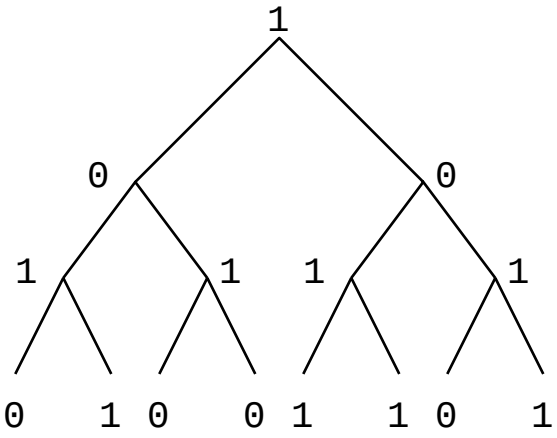

Figure 2:

of these entries is in some combinatorial rectangle. If two of them are in the same rectangle R, then there must also be 0 entries in the opposite corners, but this would mean that neither player was sure of the answer and so the protocol is incorrect. Therefore there are at least 2^n rectangles (one associated with each of the diagonal elements), and in the worst case the protocol will take n transmissions.

Game Tree Evaluation

By a game tree here we mean a strict binary tree where all nodes are labeled with either a 1 or a 0 (figure 2).

A given game tree is meant to represent a game between two players, the 0-player and the 1-player. The game is played as follows. The player that corresponds to the label at the root starts the game by moving down through the left or right child of the root. The player that corresponds to the label of the node just reached moves next by again picking a child of the current node. This process is repeated until a leaf is reached. The goal of each player is to reach a leaf that is labeled with its number. That is a 0 for the 0-player and a 1 for the 1-player.

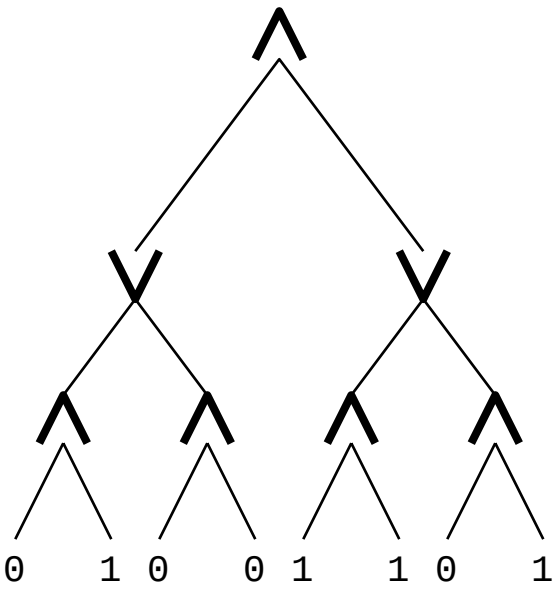

Figure 3:

Let C_T be the boolean circuit that results from relabeling the internal nodes of T as follows (figure 3). If v is an internal node of T labeled with 1 then, v is labeled with a $\vee$ in C_T. If v on the other hand is an internal node of T labeled with a 0 then v is labeled with an $\wedge$ in C_T.

We claim that given a game tree T, determining which player wins is equivalent to evaluating the circuit C_T. By this we mean that player 1 wins the game iff C_T evaluates to 1.

Proof: The proof is by induction in the height of the game tree. For the base case we consider game trees of height one. Let T be such a tree. If the label at the root is 0 then the 1-player will win iff both leaves are labeled 1. Now, the root of the circuit C_T is labeled with $\wedge$ and therefore it will evaluate to 1 iff both leaves are labeled 1. The case where the root of T is labeled 1-player is similar.

For the inductive step, assume that given a game tree of height at most n, the 1-player wins iff the corresponding circuit evaluates to 1. Now, let T be a game tree of height $n+1$, and let T_r and T_l be the subtrees rooted at the right and left child of T respectively.

If the root of T is labeled with a 0, then the 1-player wins the game iff it wins the games represented by T_r and T_l. Since the root of C_T is labeled with $\wedge$, this means that the 1-player wins iff C_T evaluates to 1. If the root of T is labeled with a 1 then the 1-player wins the game iff it wins either one of the games represented by T_r or T_l. Now, since C_T in this case is labeled with a $\vee$, this is equivalent to saying that the 1-player wins iff C_T evaluates to 1.