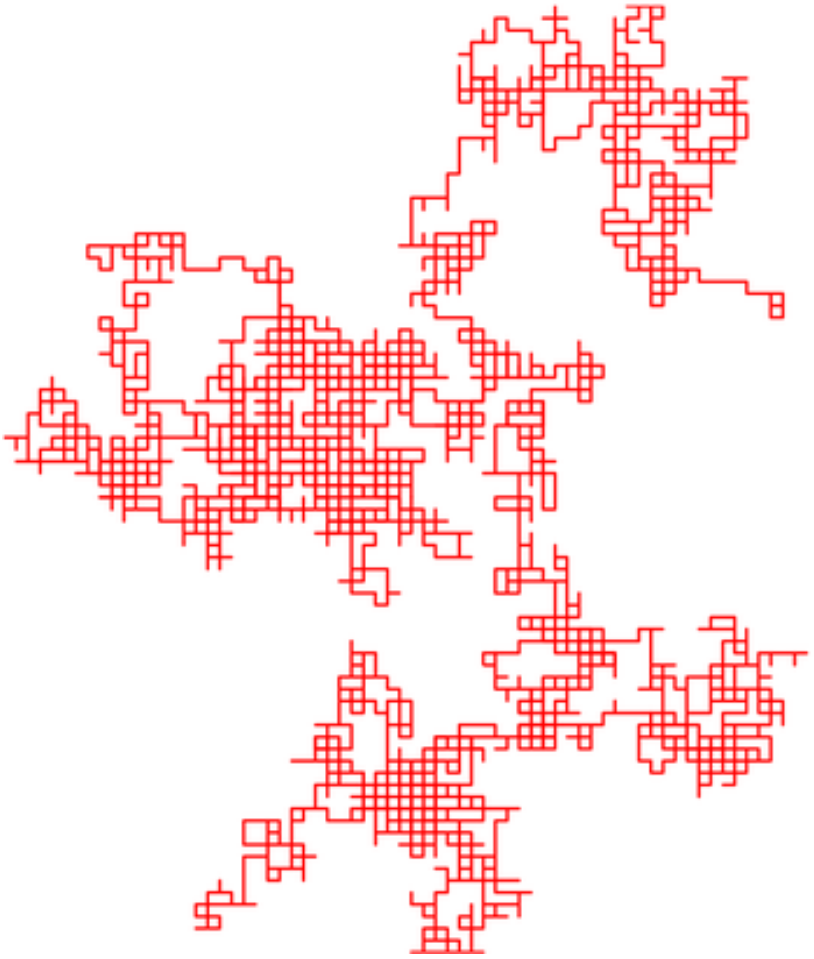
# Randomness and Computation

NJ Governor's School

# Attribution

- These slides were prepared for the New Jersey Governor's School course "The Math Behind the Machine" taught in the summer of 2011 by Grant Schoenebeck
- Large parts of these slides were copied or modified from the previous years' courses given by Troy Lee in 2010 and Ryan and Virginia Williams in 2009.

# Random Walk



# Randomness in Algorithms

- Suppose we:
  - Allow our ideal computer access to truly random bits
  - Are satisfied with the correct answer 99.9999999% of the time.
- Do we get any additional computational power?
- Such computers can do things that a deterministic
- computer cannot.
  - For example, output a random string.
    - Given n, a randomized algorithm can output a n-bit string that is random with high probability.
    - If there was a deterministic program which could do this, it would (for large enough n) contradict the fact that the string is incompressible!

# A guessing game...

- I am thinking of 10 numbers from 1 to 20...
- Can you guess one of them?
- No matter what my choice, the probability a random algorithm would not succeed in 10 guesses is $2^{-10}$

# Primality Testing

- Given integer n, is n prime?
- Sieve of Eratosthenes takes time $\sqrt{n}$
- Rabin (1980) gave a randomized algorithm taking time $\log(n)^3$
  - (called Miller-Rabin test)
  - Repeating the test 50 times, the probability a composite is declared prime is at most $2^{-100}$
- In 2002, Agrawal, Kayal, and Saxena gave an efficient deterministic algorithm for primality testing.
  - Kayal and Saxena were still undergraduates at the time.
  - The running time of this algorithm is about $\log(n)^6$

# Can Randomness Help?

- Seems inconceivable that access to a random string helps you compute.
- Reasonable complexity assumptions imply it ain’t so.
- Some practical algorithms can only be done with randomness or are faster with randomness.
- Provably help in other contexts!

# Polynomial Identity Testing

- Polynomial Identity Testing: Given a polynomial is it identically 0?

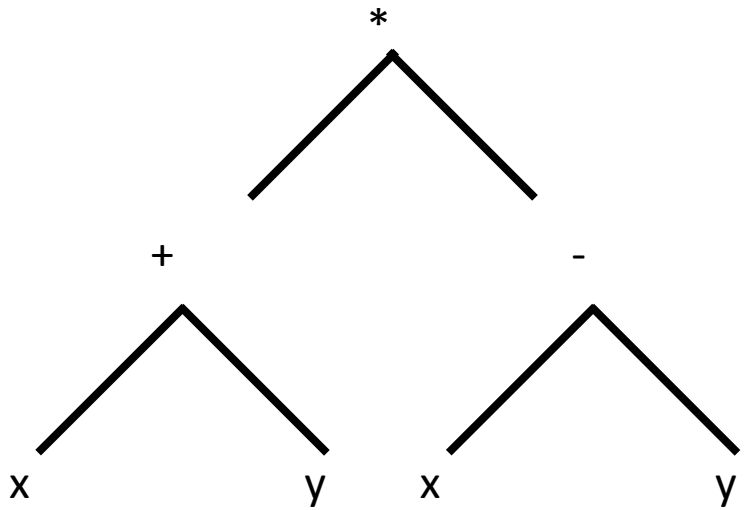
$$p(x,y) = x^2 - y^2 - (x+y)(x-y)$$

$$\begin{aligned} p(x_1,\ldots,x_4,y_1,\ldots,y_4) &= (x_1^2+x_2^2+x_3^2+x_4^2)+(y_1^2+y_2^2+y_3^2+y_4^2) \\ &-(x_1y_1-x_2y_2-x_3y_3-x_4y_4)^2-(x_1y_2+x_2y_1+x_3y_4-y_4x_3)^2 \\ &-(x_1y_3-x_2y_4+x_3y_1+x_4y_2)^2-(x_1y_4+x_2y_3-x_3y_2+x_4y_1)^2 \end{aligned}$$

- What the problem? We just expand the polynomial and see if everything cancels?
- A polynomial with n variables and degree d can have $\binom{n+d}{d}$ terms. Ideally, running time poly in n and d.

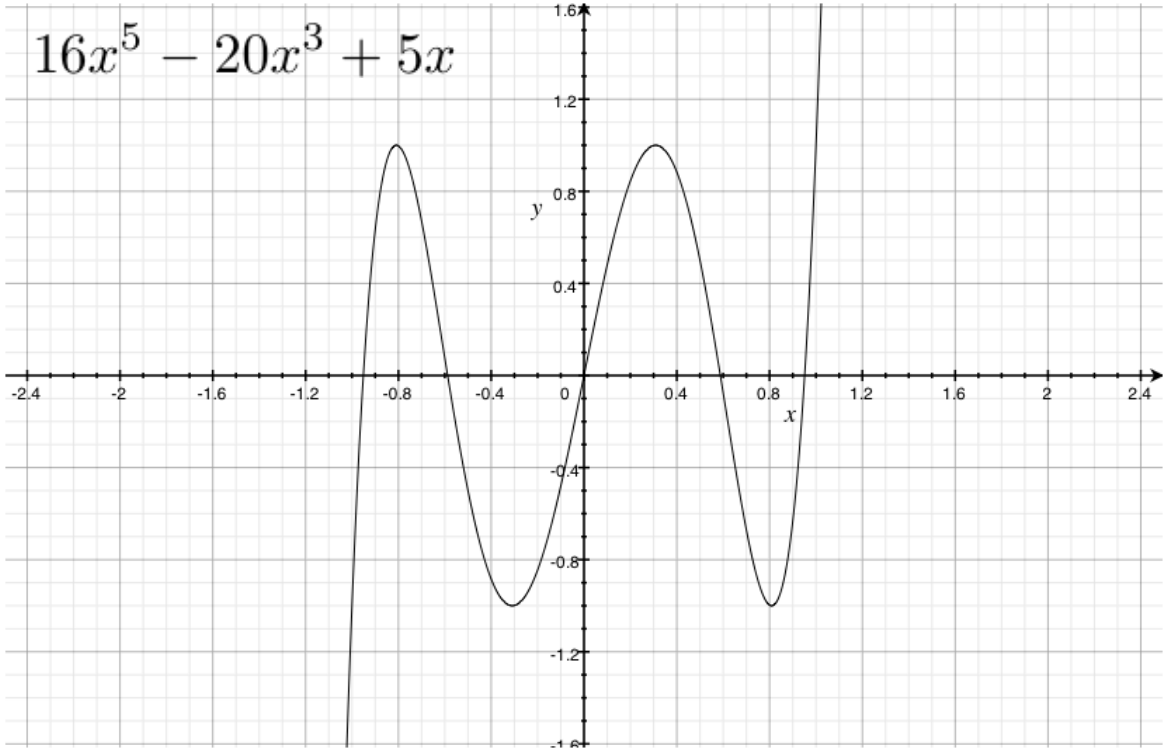
# Representation

- Polynomial given to us as a circuit.
- Degree of a polynomial is maximum sum of exponents in a monomial.
  - E.g degree$(x^2y^3z + xy^2z)$= ?



# Univariate Case

Say we have a univariate degree d polynomial.

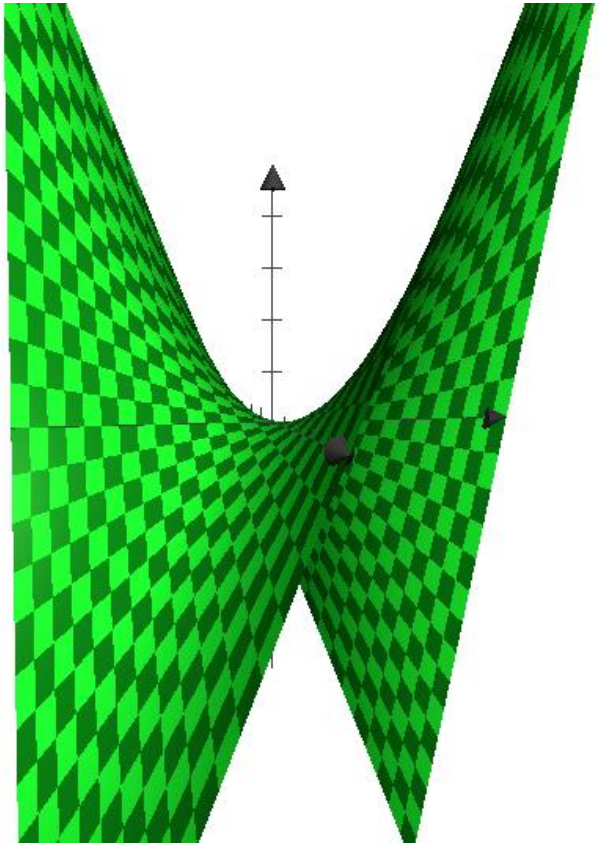



This will have
at most d roots!

Evaluate the polynomial at $1, 2, 3, \ldots, d+1.$

# With more variables...

- f(x, y) = xy
- Now there are an infinite number of zeros!



# Return of the guessing game

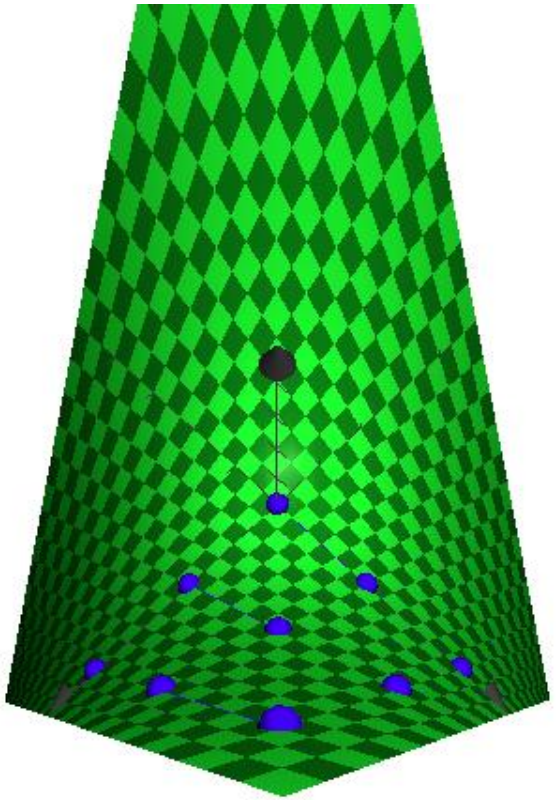
- It still looks like we would have to pretty unlucky for all the points we choose to be roots of the polynomial.
- Try evaluating the polynomial at random points!

# Classic Example of a Randomized Algorithm

- Let $p(x_1, x_2, \ldots, x_n)$be an n-variate polynomial of degree d.
- Let S = {1, 2,…, 2d} Choose $a_1$, $a_2$, …, $a_n$ uniformly from S.
- Evaluate $p(a_1, a_2, \ldots, a_n)$
  - If zero say it's the zero polynomial.
  - o. w. we know its not the zero poly.

# Example

- f(x, y) = xy
  evaluated at points
  {0, 1, 2} X {0, 1, 2}



# Schwartz-Zippel Lemma

- Lemma:
  $p(x_1, \ldots, x_n)$ st degree(p) = d and not $\equiv 0$.
  For any set S, if $a_1, a_2, \ldots, a_n$ are each chosen at random from S then $Pr[p(a_1, \ldots, a_n) = 0] < {}^{d}/_{|S|} = {}^{1}/_{2}$

# Perspectives

- The deterministic primality testing algorithm first formulated the problem as checking a polynomial identity.
- Giving an efficient deterministic algorithm for polynomial identity testing is one of the most important open problems in TCS.

# Communication Complexity
# Another place were Randomness Helps!

- The dialogues of Alice and Bob...

# Alice and Bob make a date

Are you free on Friday?

No, have to work at the Krusty Krab.

How about Saturday?

No, have to work at the Krusty Krab.

⋮

How many emails can it take to set a date?

# Measures of Communication

- We want to quantify the amount of communication sent back and forth.
- Several ways to do this: number of emails, total number of characters, volume of breath...
- Being computer scientists, we will use bits.

# Alice's schedule

Alice's schedule for the week:

| Mon | | | | Tue | | | | ... |
|---|---|---|---|---|---|---|---|---|
| | Morn | Aftn | Eve | | Morn | Aftn | Eve | ... |
| | Free | Busy | Busy | | Busy | Free | Free | ... |

We can convert this into a string of bits where

0=Busy, 1=Free

Any 21 bit string is a valid input.

# Set Intersection

Alice's schedule for the week:

| Mon | | | | Tue | | | | ... |
|---|---|---|---|---|---|---|---|---|
| | Morn | Aftn | Eve | | Morn | Aftn | Eve | ... |
| | 1 | 0 | 0 | | 0 | 1 | 1 | ... |

Bob's schedule for the week:

| Mon | | | | Tue | | | | ... |
|---|---|---|---|---|---|---|---|---|
| | Morn | Aftn | Eve | | Morn | Aftn | Eve | ... |
| | 0 | 0 | 1 | | 0 | 0 | 1 | ... |

# Set Intersection

Alice's schedule for the week:

100 011 100 011 101 010 000

Bob's schedule for the week:

001 001 001 001 001 110 110

Q: Is there a position where both strings have a '1'?

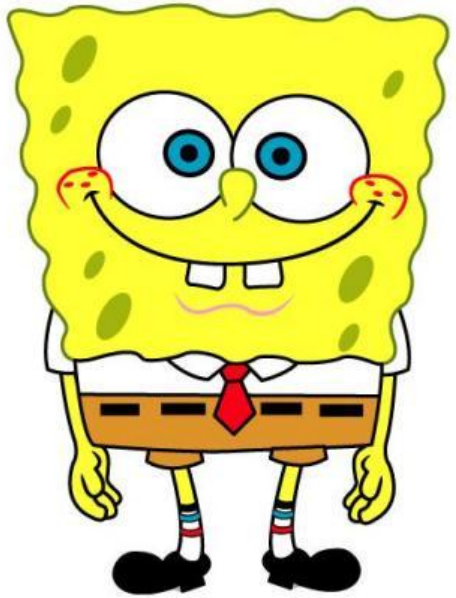
Known as the set intersection problem.

# Set Intersection

- In general, Alice and Bob will each hold a n bit binary string.
- Again, the task is to decide if these strings have a common position with a ‘1’ or not.
- Notice the problem can always be solved with n+1 bits of communication
  - Alice can send her entire input to Bob, he can produce the correct output.
  - This is known as the trivial protocol.

# Try It!

?



# Communication Matrix

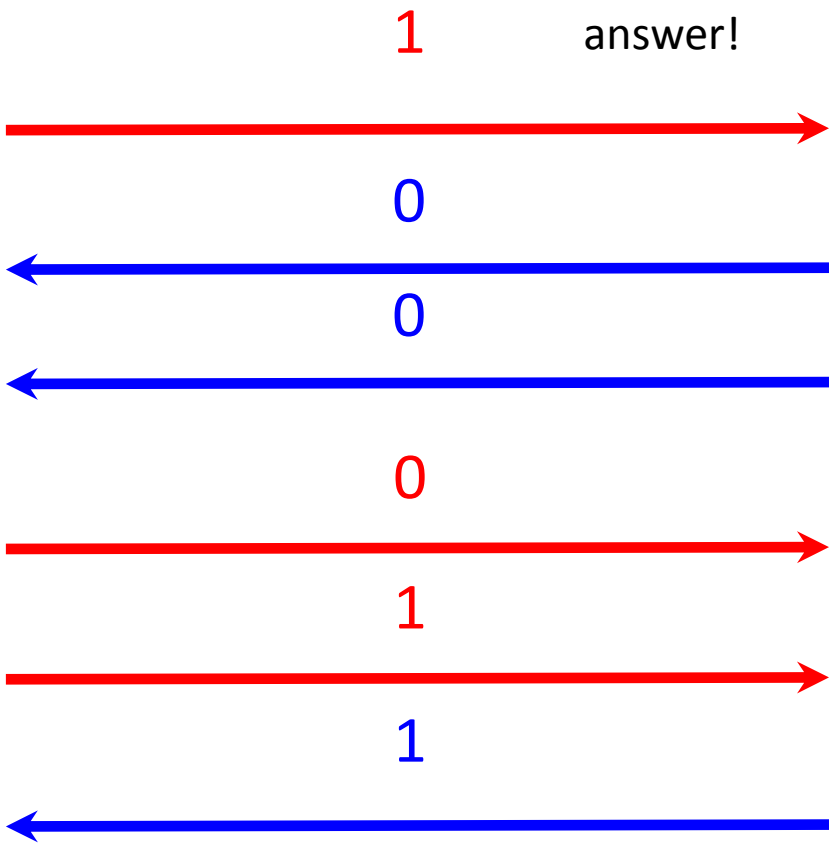
# Communication Matrix

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 010 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 011 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 100 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 101 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 110 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 111 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Communication Protocol

We also assume the communication is in bits.

The last bit of the protocol is the answer!




100 011...

001 001...

# Communication Protocol

The first bit Alice sends depends only on her input. She knows nothing about Bob's schedule...

1

100 011...

001 001...

# Communication Protocol

The first bit Alice sends depends only on her input. She knows nothing about Bob's schedule...

100 011...

1

001 001...

100 111...

0

001 001...

# Communication Protocol

The first bit Alice sends depends only on her input. She knows nothing about Bob's schedule...

100 011...

1

001 001...

100 111...

0

001 001...

111 111...

1

001 001...

# Communication Protocol

The first bit Alice sends depends only on her input. She knows nothing about Bob's schedule...

1

111 111...

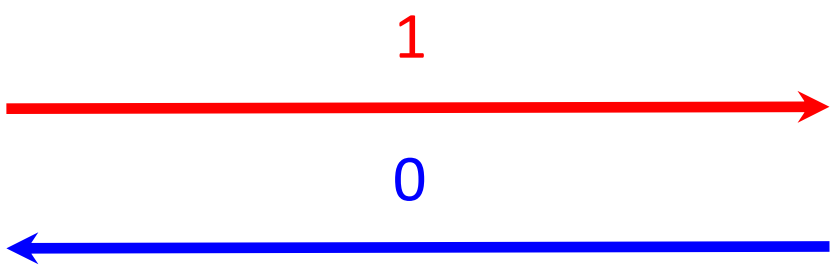
001 001...

We can divide Alice's inputs into two groups:

-those where she first says 1
-those where she first says 0

# Communication Protocol

The first bit Bob sends depends only on his input and the bit sent by Alice.




100 011...

001 001...

# Communication Protocol

The first bit Bob sends depends only on his input and the bit sent by Alice.

100 011...

1

0

001 001...

100 011...

1

1

011 001...

# Communication Protocol

The first bit Bob sends depends only on his input and the bit sent by Alice.

100 011...

1

1

011 001...

Conclusion: Bob groups his inputs into two sets, conditioned on Alice's message.

# Communication Matrix

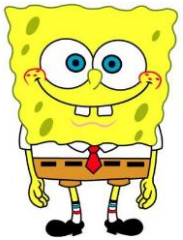
# Communication Matrix

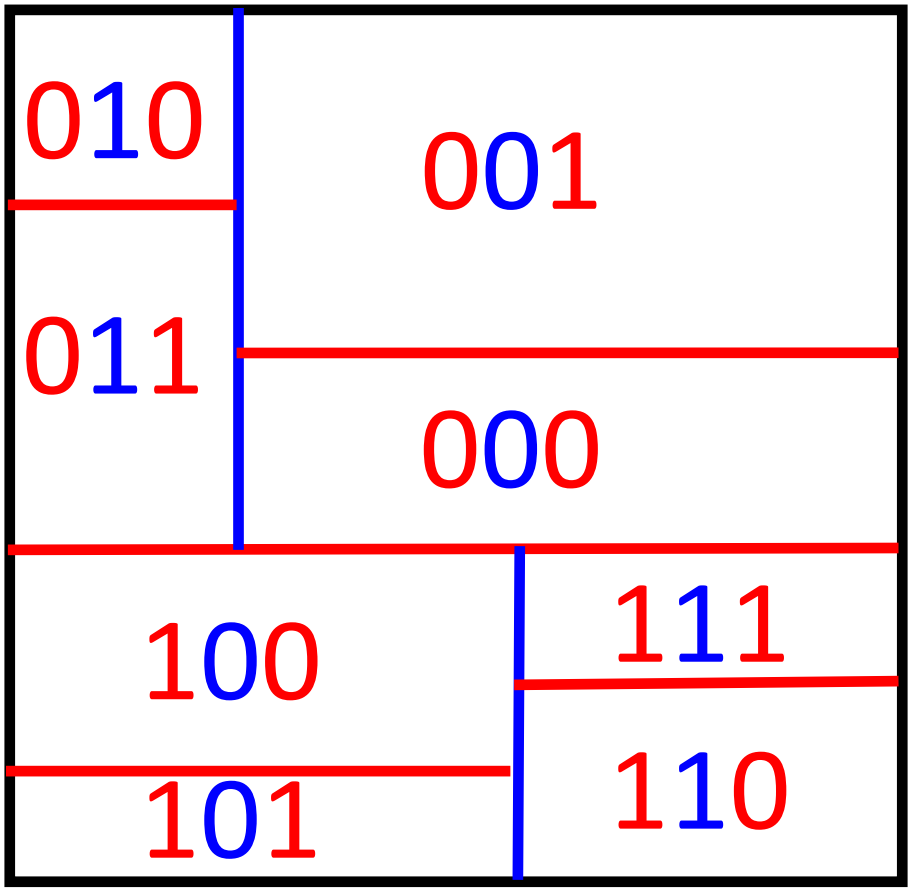
| | |
|---|---|
| 0 | |
| 1 | |

# Communication Matrix

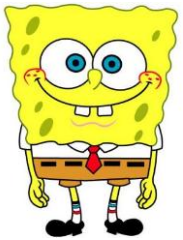
| | |
|---|---|
| 01 | 00 |
| 10 | 11 |

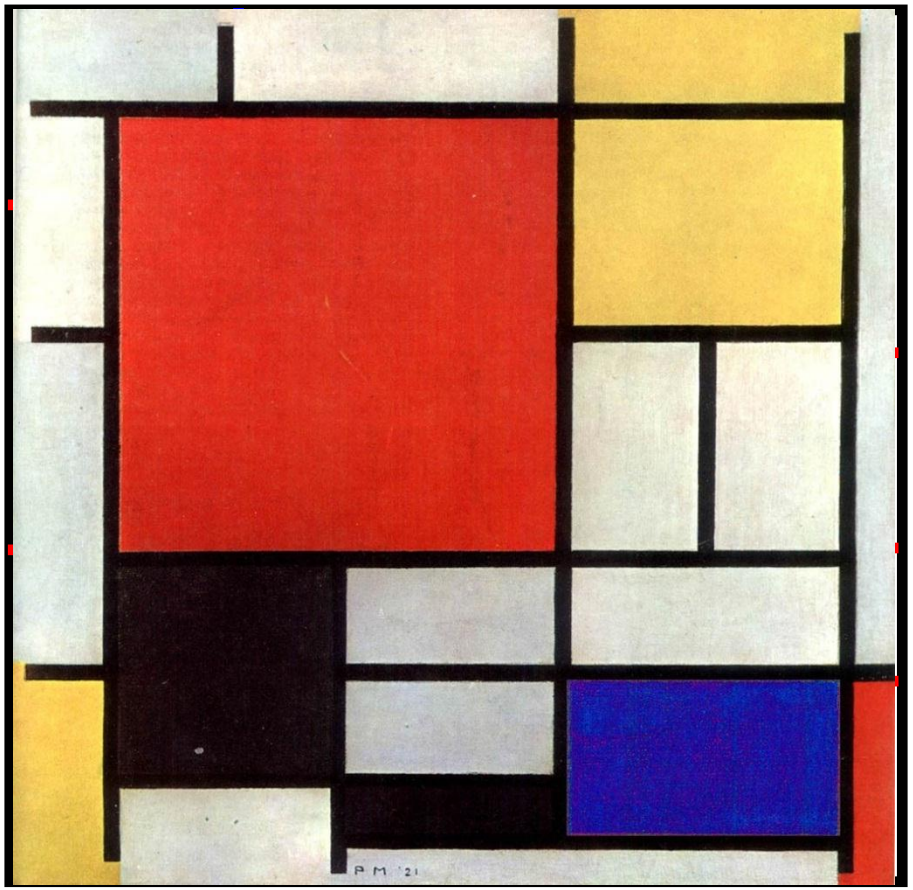
# Communication Matrix





# Communication Matrix

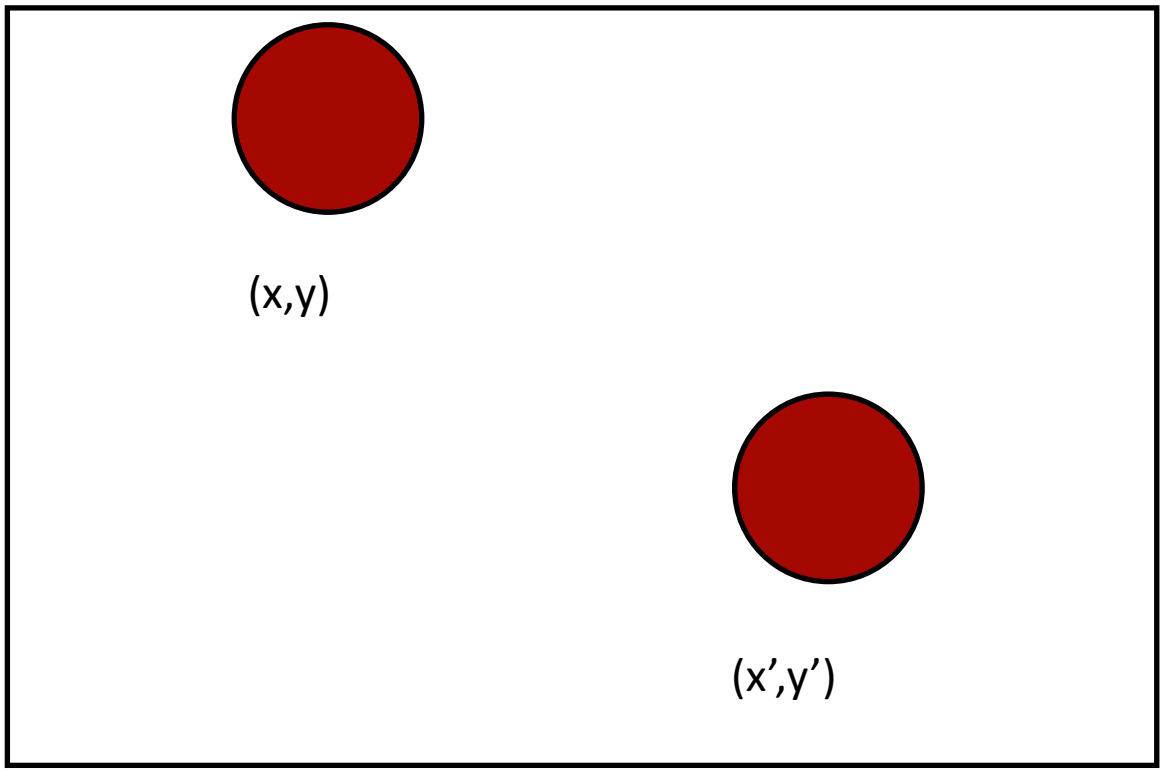




# Observations

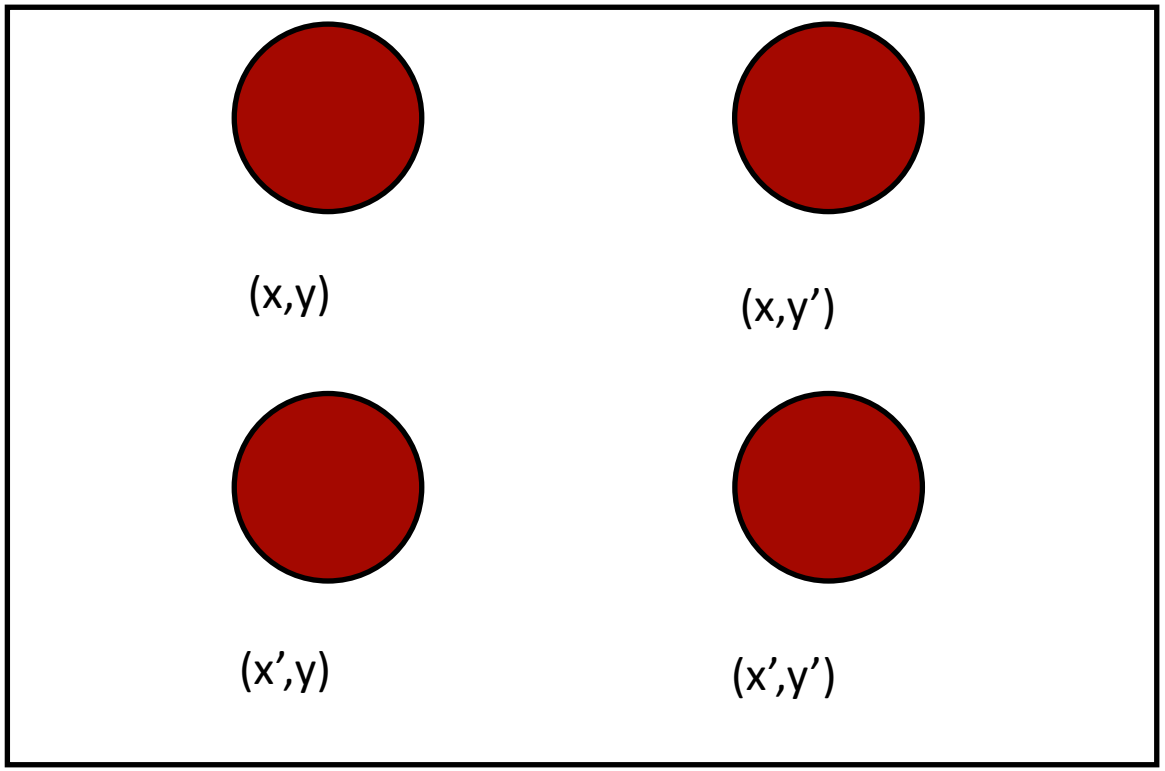
- Protocol partitions matrix into rectangles.
- With each additional bit of communication, a rectangle can be split into two. After c bits of communication, at most $2^c$ rectangles.
- For protocol to be correct, all inputs lying in the same rectangle must have the same output value. We say that such a rectangle is monochromatic.

# Diagonal property of rectangles




If (x,y) and (x’,y’) lie in the same rectangle...

# Diagonal property of rectangles




So must (x,y') and (x',y).

# Going back to SI

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 010 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 011 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 100 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 101 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 110 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 111 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

# Going back to SI

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 010 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 011 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 100 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 101 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 110 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 111 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

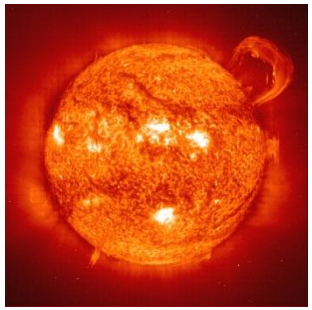
# Lower bound for SI

- Notice:
  - all anti-diagonal entries are 0.
  - all entries below anti-diagonal are 1.
- No two anti-diagonal entries can be in the same monochromatic rectangle!
- This means at least $2^n$ monochromatic rectangles are needed---n bits of communication.

# Recap

- We just showed a lower bound---any protocol with less than n bits of communication will incorrectly answer the set intersection problem on some input.
- For set intersection, the trivial protocol is optimal!

# Bob goes to the moon



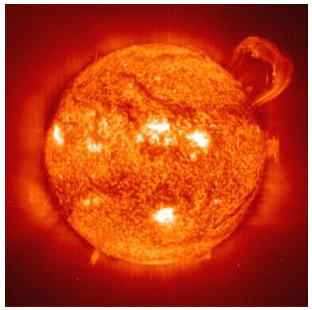
Is the file corrupted
along the way?

Alice wishes to send
a huge file M to Bob

Bob receives some file M'.

Does M=M'?

# Bob goes to the moon



M

M’

Bob could just send M’ back.

This is very costly.

# Checking Equality

More abstractly, the problem is the following:

Alice has an n bit string M, Bob has an n bit string M'.

They want to answer 1 if M=M' and 0 otherwise.

# Communication Matrix

This is known as the identity matrix

| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|---|---|---|---|---|---|---|---|---|
| 000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 001 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 010 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 011 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 100 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 101 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 110 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 111 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

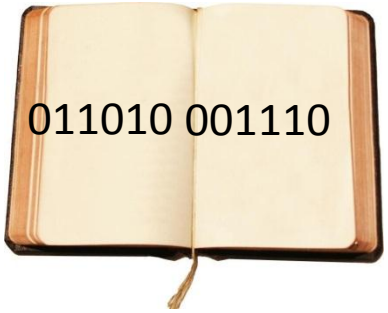
# Communication Matrix

- By the diagonal property of rectangles, no two pairs (x,x) and (y,y) can be in the same monochromatic rectangle.
- Again we need $2^n$ many monochromatic rectangles, and so n bits of communication.
- Is communication complexity always so boring?

# Randomized Model

- Not if we allow randomness!
- Alice and Bob are each given the same book of random numbers. This is known as shared randomness.
- For every input, they must output the correct answer with probability 90%.

# Randomized Protocol




Alice looks at string
R=first n bits of book.

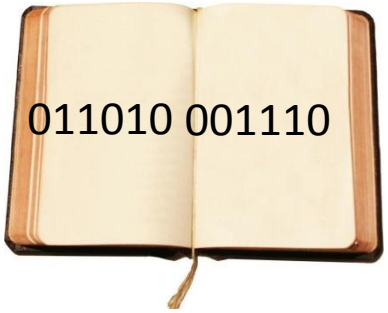
Then she computes

$$a = M \cdot R = \sum_i M_i R_i \mod 2.$$

M

M’

# Randomized Protocol




Alice looks at string
R=first n bits of book.

Bob checks if a=a', where

$$a' = M' \cdot R = \sum_i M'_i R_i \mod 2.$$

M

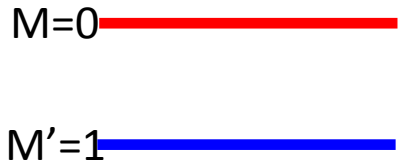
M'

If they are equal he outputs 1, otherwise outputs 0.

# Correctness of Protocol

- If M=M’ then we will always have a=a’. No mistakes.
- If $M \neq M'$ what is the probability that a=a’?
- M and M’ must differ in some position. Assume it is the first one.

# Correctness of Protocol




Consider the random strings R in pairs (0r, 1r).

Now notice that

$$M \cdot 0r = M \cdot 1r$$

$$M' \cdot 0r \neq M' \cdot 1r$$

So either

$$M \cdot 0r \neq M' \cdot 0r \quad \text{or} \quad M \cdot 1r \neq M' \cdot 1r$$

For half the random strings, we get different answers.

# Randomized Protocol

Protocol Recap:

Alice computes $$a = M \cdot R = \sum_i M_i R_i \mod 2.$$

Bob computes $$a' = M' \cdot R = \sum_i M'_i R_i \mod 2.$$

If M=M' then a=a' for any R.

If $M \neq M'$ then a=a' for at most half of the R's.

# Deterministic vs. Random

- Here we have a case where with randomness we can provably do better than without.
- Deterministically, we must send the entire input, n bits!
- But if Alice and Bob share a book of randomness, constant communication suffices.

# On the model

- Sharing randomness is a strong assumption.
- Can eliminate assumption, but communication increases to log(n) bits.
  - First “derandomize” to use log(n) bits of public randomness
  - Then can just share the random coins.

# Equality with random primes

- Idea: With her own (private) randomness Alice chooses a random prime number between 1 and $n^2$.
- Alice sends Bob p and a=M mod p, known as "fingerprint" of M.
- Again Bob checks if a=M' mod p.

# Correctness of the protocol

- How many bits does this take?
- If M=M’ then again clearly a=a’.
- Suppose that $M \neq M'$ yet a=a’:
  - M mod p = M’ mod p;
  - M-M’ = 0 mod p
- How many prime divisors can M-M’ have?
  - At most log(M)

# There are lots of primes!

- Prime number theorem (1896): Asymptotically, the number of primes less than N is $\frac{N}{\log N}$
  - a random n bit number is prime with probability < 1/n
- $\frac{N}{2 \log N}$ primes less than $n^2$
- only log(N) of them are “bad”.

# Questions?

- How do you choose a random prime efficiently?
- Can you find an n-bit prime efficiently deterministically?
- Can every protocol with shared randomness be modified to use private randomness in this way?