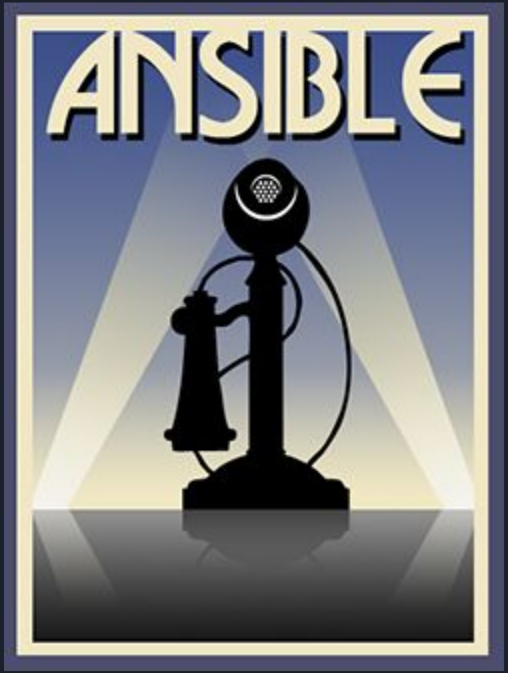
Problem Solving Without Ansibles:

An Introduction to Communication Complexity

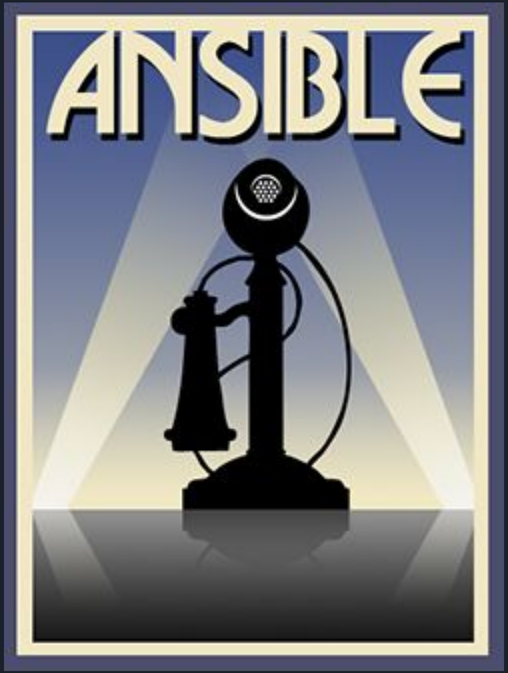
Chris Grossack (they/them)

Ansibles in Science Fiction:

Ansibles in Science Fiction:

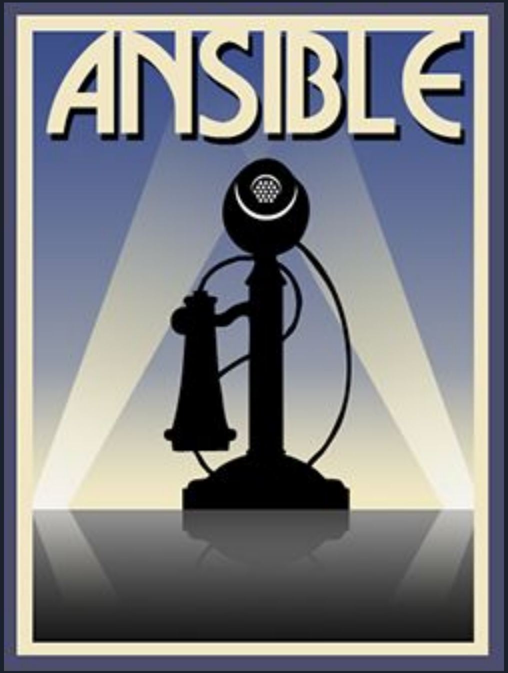


Ansibles in Science Fiction:



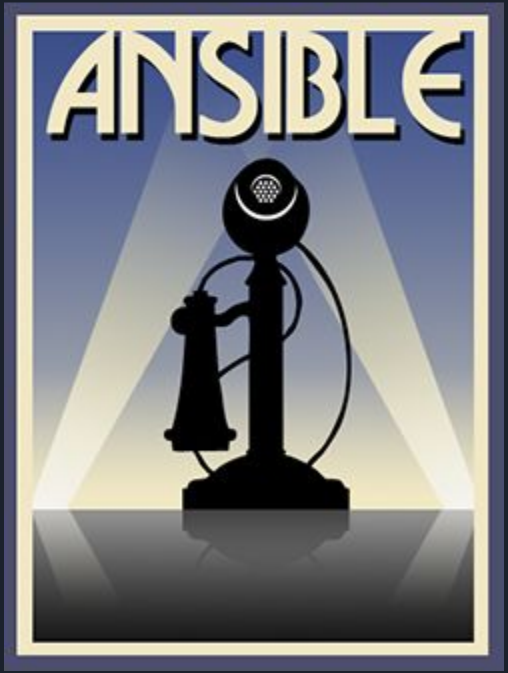
- Faster-than-light communication

Ansibles in Science Fiction:



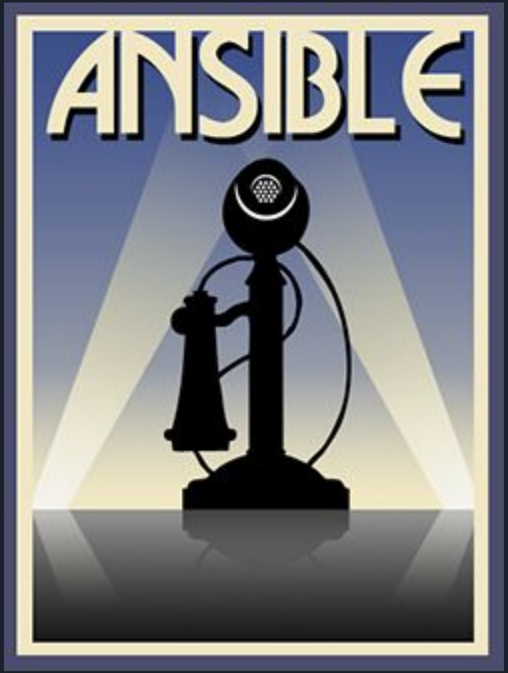
- Faster-than-light communication
- Allows colonies, etc. to speak in real time

Ansibles in Science Fiction:



- Faster-than-light communication
- Allows colonies, etc. to speak in real time
- Generally helps the plot go brrrrrr

Ansibles in Science Fiction:

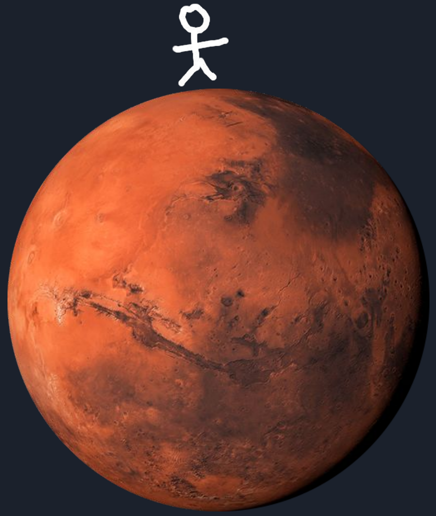


- Faster-than-light communication
- Allows colonies, etc. to speak in real time
- Generally helps the plot go brrrrrr
- Notably aphysical -- almost certainly impossible

In the real world...

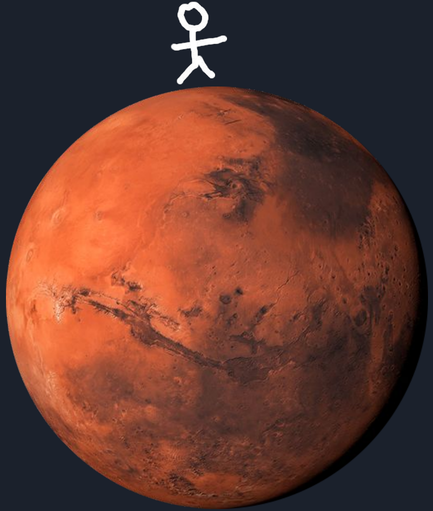
In the real world...

In the real world...



In the real world...

$a \in \{0,1\}^n$



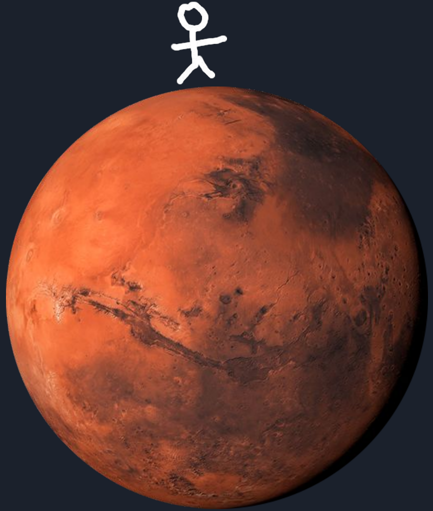
$b \in \{0,1\}^n$

In the real world...

$f(x,y) : 2^n \times 2^n \to 2$

In the real world...

$f(x,y) : 2^n \times 2^n \to 2$



$f(a,b)$

$a \in \{0,1\}^n$

$b \in \{0,1\}^n$

We only penalize **Communication** between Alyss and Bob

Some remarks:

- We never need $> n+1$ messages

Some remarks:

- We never need $> n+1$ messages
  - Alyss sends a to Bob (n messages)
  - Bob computes $f(a,b)$
  - Bob sends result to Alyss (1 message)
-

Some remarks:

- We never need $> n+1$ messages
  - Alyss sends a to Bob (n messages)
  - Bob computes $f(a,b)$
  - Bob sends result to Alyss (1 message)
- So “efficient” means Polylog(n) messages

Some remarks:

- We never need $> n+1$ messages
  - Alyss sends a to Bob (n messages)
  - Bob computes $f(a,b)$
  - Bob sends result to Alyss (1 message)

- So “efficient” means Polylog(n) messages
  - log(n)

Some remarks:

- We never need $> n+1$ messages
  - Alyss sends a to Bob (n messages)
  - Bob computes $f(a,b)$
  - Bob sends result to Alyss (1 message)

- So “efficient” means Polylog(n) messages
  - $\log(n)$
  - $(\log(n))^k$

Some remarks:

- We never need $> n+1$ messages
  - Alyss sends a to Bob (n messages)
  - Bob computes $f(a,b)$
  - Bob sends result to Alyss (1 message)

- So “efficient” means Polylog(n) messages
  - $\log(n)$
  - $(\log(n))^k$
  - $\log(n) \log(\log(n))$

Some remarks:

- We never need $> n+1$ messages
  - Alyss sends a to Bob (n messages)
  - Bob computes $f(a,b)$
  - Bob sends result to Alyss (1 message)

- So “efficient” means Polylog(n) messages
  - $\log(n)$
  - $(\log(n))^k$
  - $\log(n) \log(\log(n))$

- In general, we want #messages $< \log(n)^k$ for some constant k

A simple example:

Write $D(f) \triangleq \min_{\text{protocol } \underline{P}} \max_{(a,b)}$ #messages sent using $\underline{P}$ to compute $f(a,b)$

A simple example:

Write $D(f) \triangleq \min_{\text{protocol } P} \max_{(a,b)}$ #messages sent using P to compute $f(a,b)$

What is $D(Eq_n)$,

where $Eq_n(x,y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$?

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Idea:

$$\begin{array}{cc} & b \\ 2^n \; a & \begin{pmatrix} & \vdots & \\ \cdots & f(a,b) & \\ & & \end{pmatrix} \\ & 2^n \end{array}$$

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Idea:

$$
A \quad \begin{array}{c c}
\begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 & \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\
 & B
\end{array}
$$

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Let’s see a sample computation with some (suboptimal) protocol:

Idea:

$$A \quad \begin{array}{c|cccc} & \\ 00 & 1 & 0 & 0 & 0 \\ 01 & 0 & 1 & 0 & 0 \\ 10 & 0 & 0 & 1 & 0 \\ 11 & 0 & 0 & 0 & 1 \\ \hline & 00 & 01 & 10 & 11 \end{array}$$

$$B$$

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Let's see a sample computation with some (suboptimal) protocol:

Idea:

$$
A \quad \begin{array}{c c}
00 & \\
01 & \\
10 & \\
11 &
\end{array}
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

$$
\begin{array}{cccc}
00 & 01 & 10 & 11
\end{array}
$$

$$B$$

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Idea:

$$A \quad \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{array}{cccc} 00 & 01 & 10 & 11 \end{array}$$

$$B$$

Let's see a sample computation with some (suboptimal) protocol:

1. Alyss sends her first bit (0) to Bob
   This restricts the region of the matrix Bob is interested in

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Idea:

$$
\begin{array}{cc}
A \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \!\!\!\! & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 & \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\
 & B
\end{array}
$$

Let's see a sample computation with some (suboptimal) protocol:

1. Alyss sends her first bit (0) to Bob
   This restricts the region of the matrix Bob is interested in

2. Maybe Bob sends his second bit (0) to Alyss
   This restricts the region of interest again

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

Idea:

$$
A \quad
\begin{array}{c}
00 \\ 01 \\ 10 \\ 11
\end{array}
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

$$
\begin{array}{cccc}
00 & 01 & 10 & 11
\end{array}
$$

$$B$$

Let's see a sample computation with some (suboptimal) protocol:

1. Alyss sends her first bit (0) to Bob

    This restricts the region of the matrix Bob is interested in

2. Maybe Bob sends his second bit (0) to Alyss

    This restricts the region of interest again

3. Then Alyss sends her second bit (1) to Bob

    This restricts the region again, and now the only option is 0! So we know f(a,b) = 0

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

- In general, when we follow some protocol, each message sent restricts the region of interest in this matrix.

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

- In general, when we follow some protocol, each message sent restricts the region of interest in this matrix.
- But each region we restrict to is a combinatorial rectangle

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

- In general, when we follow some protocol, each message sent restricts the region of interest in this matrix.
- But each region we restrict to is a combinatorial rectangle
- That is, a region of the form $A_0 \times B_0$, for $A_0 \subseteq A$ and $B_0 \subseteq B$

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

- In general, when we follow some protocol, each message sent restricts the region of interest in this matrix.
- But each region we restrict to is a combinatorial rectangle
- That is, a region of the form $A_0 \times B_0$, for $A_0 \subseteq A$ and $B_0 \subseteq B$
- So! Any protocol correctly computing f must partition our matrix into combinatorial rectangles, each of which only has 0s or 1s inside it.

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

- In general, when we follow some protocol, each message sent restricts the region of interest in this matrix.
- But each region we restrict to is a combinatorial rectangle
- That is, a region of the form $A_0 \times B_0$, for $A_0 \subseteq A$ and $B_0 \subseteq B$
- So! Any protocol correctly computing f must partition our matrix into combinatorial rectangles, each of which only has 0s or 1s inside it.
- In ~fancy~ lingo: Any protocol partitions the matrix into monochromatic rectangles

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

$$A \quad \begin{array}{cc}
\begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
 & \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\
 & B
\end{array}$$

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

$$A\;\begin{array}{cc} \begin{array}{c}00\\01\\10\\11\end{array} & \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} \\ & \begin{array}{cccc}00&01&10&11\end{array} \\ & B \end{array}$$

This entry MUST be a 1x1 rectangle,
as any rectangle containing 2 rows
(resp. columns) must contain an off diagonal
entry.

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

$$
A \quad \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array}
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
$$

$$
\begin{array}{cccc} 00 & 01 & 10 & 11 \end{array}
$$

$$
B
$$

This entry MUST be a 1x1 rectangle,
as any rectangle containing 2 rows
(resp. columns) must contain an off diagonal
entry.

Similarly, each of these must be 1x1 rectangles.

This means any protocol solving Eq_n has at
least 2^n rectangles.

Theorem: Eq_n is maximally hard

(That is, $D(Eq_n) = n+1$)

$$
A \quad \begin{array}{cc}
\begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} & \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \begin{array}{cccc} 00 & 01 & 10 & 11 \end{array} \\
& B
\end{array}
$$

This entry MUST be a 1x1 rectangle, as any rectangle containing 2 rows (resp. columns) must contain an off diagonal entry.

Similarly, each of these must be 1x1 rectangles.

This means any protocol solving Eq_n has at least 2^n rectangles.

But each additional message sent splits a rectangle into 2 pieces.

So Eq_n requires at least $\log_2(2^n) = n$ many messages.

Ok... What if we only want to be correct with high probability?

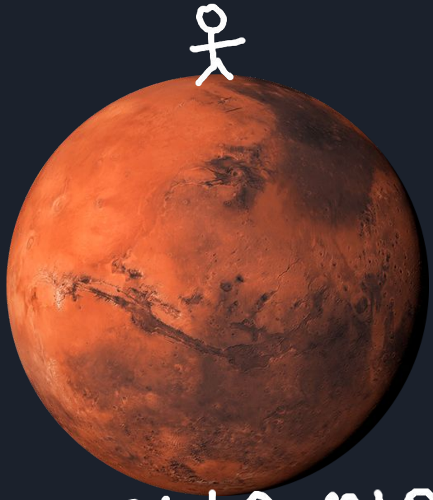
Theorem:
Alyss and Bob can communicate $O(\log n)$ bits, so that

1. If $a = b$, then we always correctly say “yes, they’re equal”

2. If $a \neq b$, then we incorrectly say “yes they’re equal” with probability $< 1/n$

(We say such an algorithm has “one-sided error”)

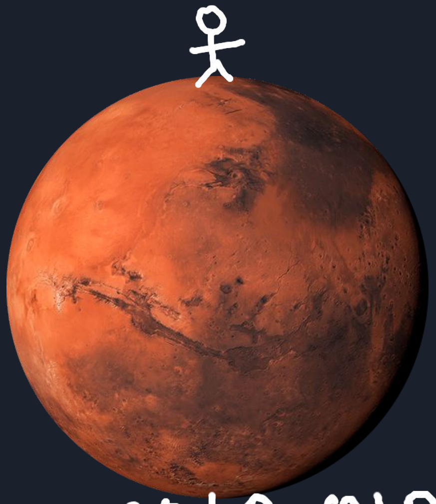
$a = 1010\ 0010$



$b = 0010\ 0101$

$$a = 1010\ 0010$$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$



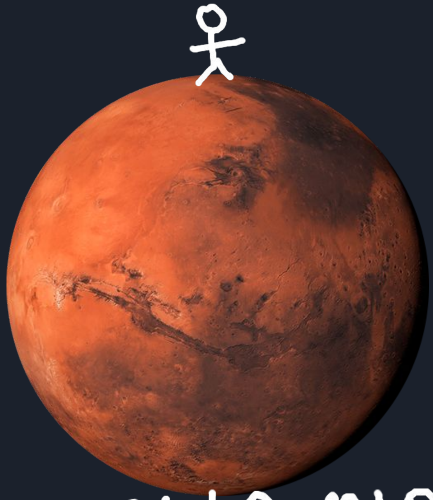
$$b = 0010\ 0101$$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

$n^2 < q < 2n^2$

$a = 1010\ 0010$

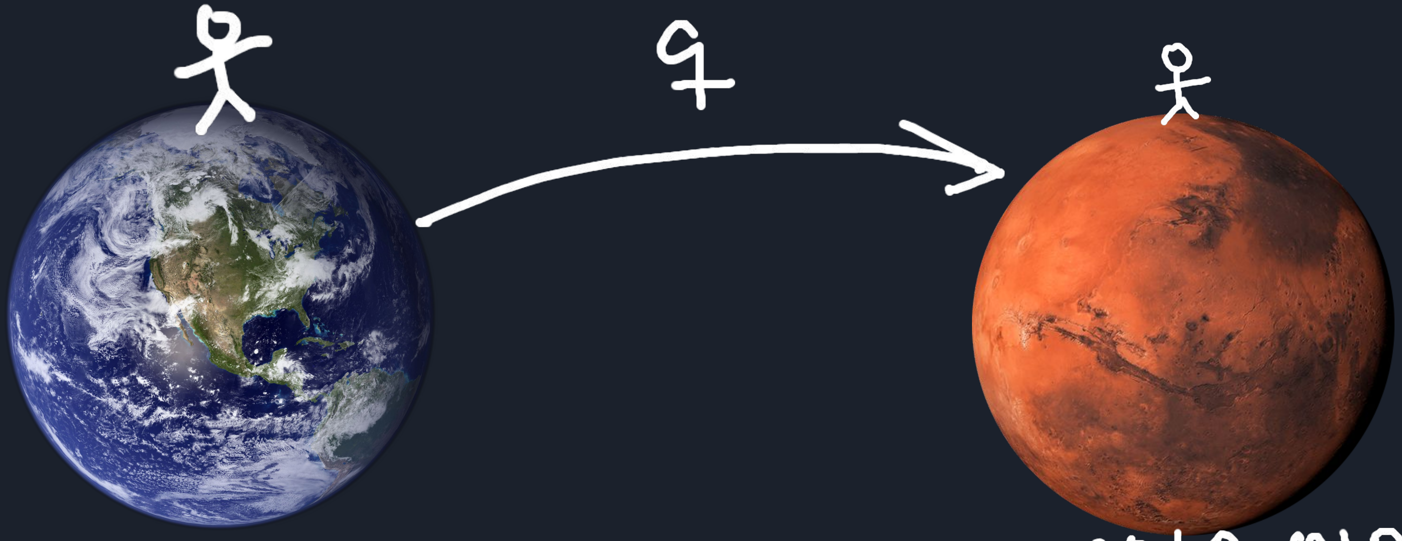
$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$



$b = 0010\ 0101$

$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$

Budget:
O(log(q))



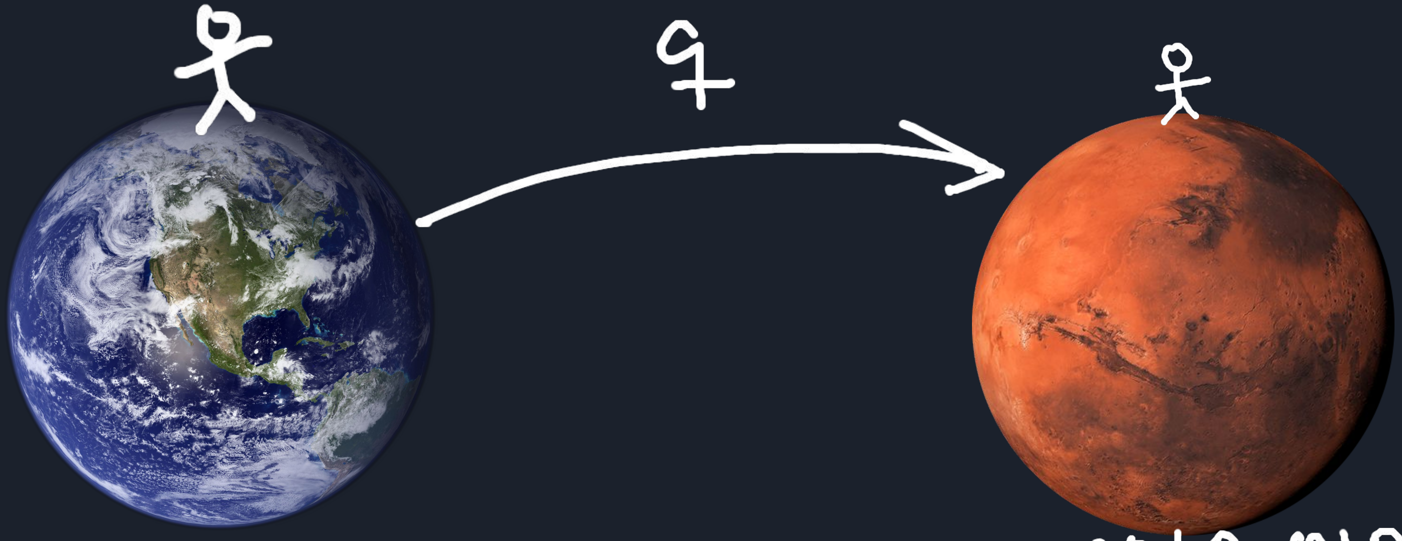
$$a = 1010\ 0010$$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$$b = 0010\ 0101$$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
$O(\log(n^2))$



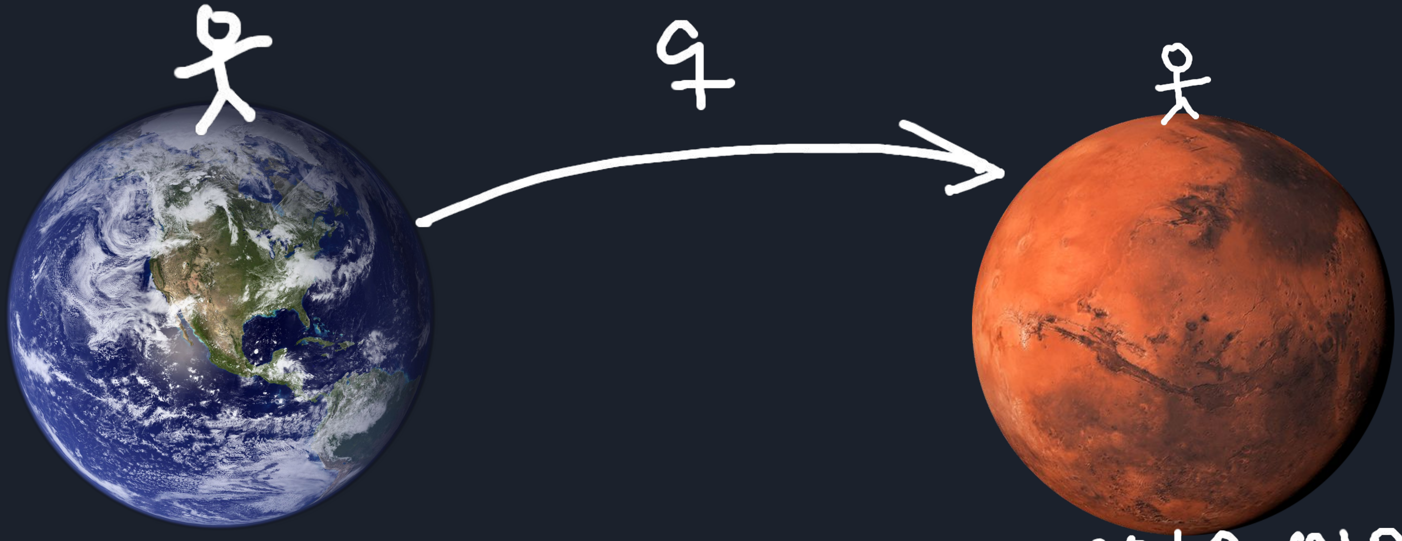
$a = 1010\ 0010$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$b = 0010\ 0101$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))



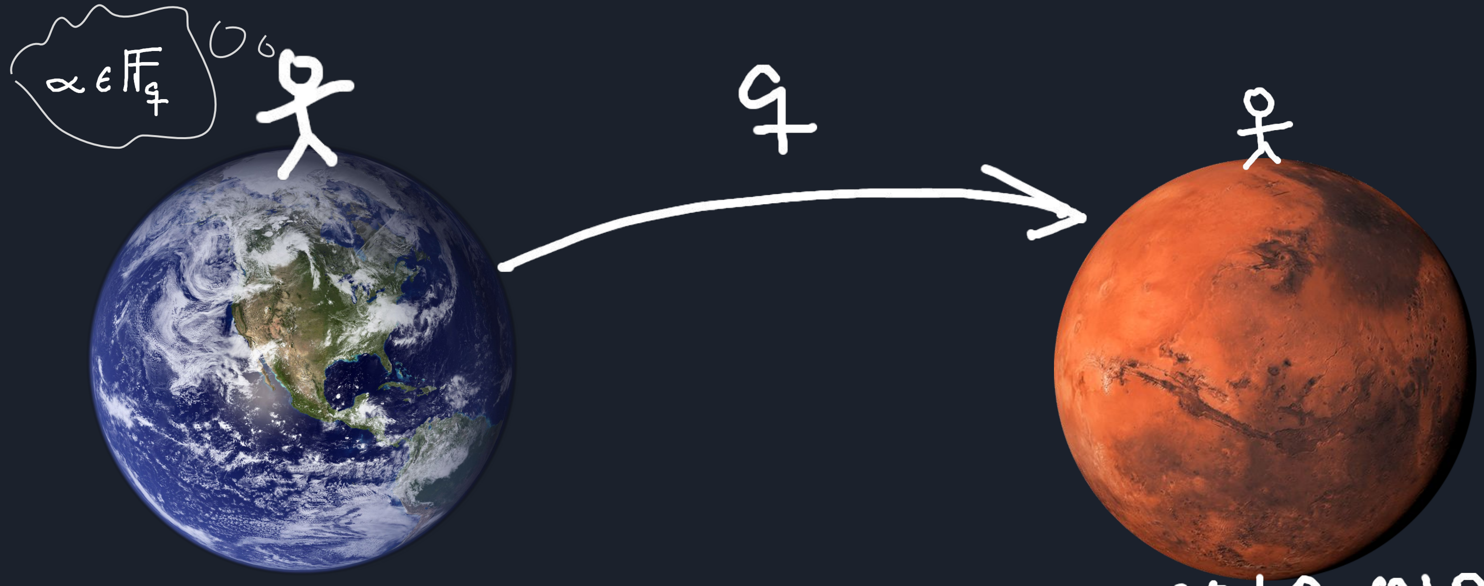
$a = 1010\ 0010$

$b = 0010\ 0101$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))



$$a = 1010\ 0010$$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

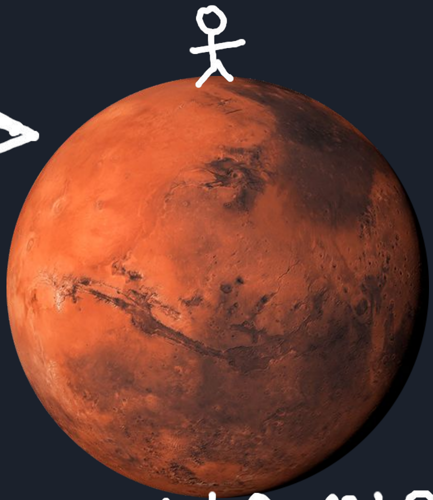
$$b = 0010\ 0101$$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))
O(log(α))

q

α



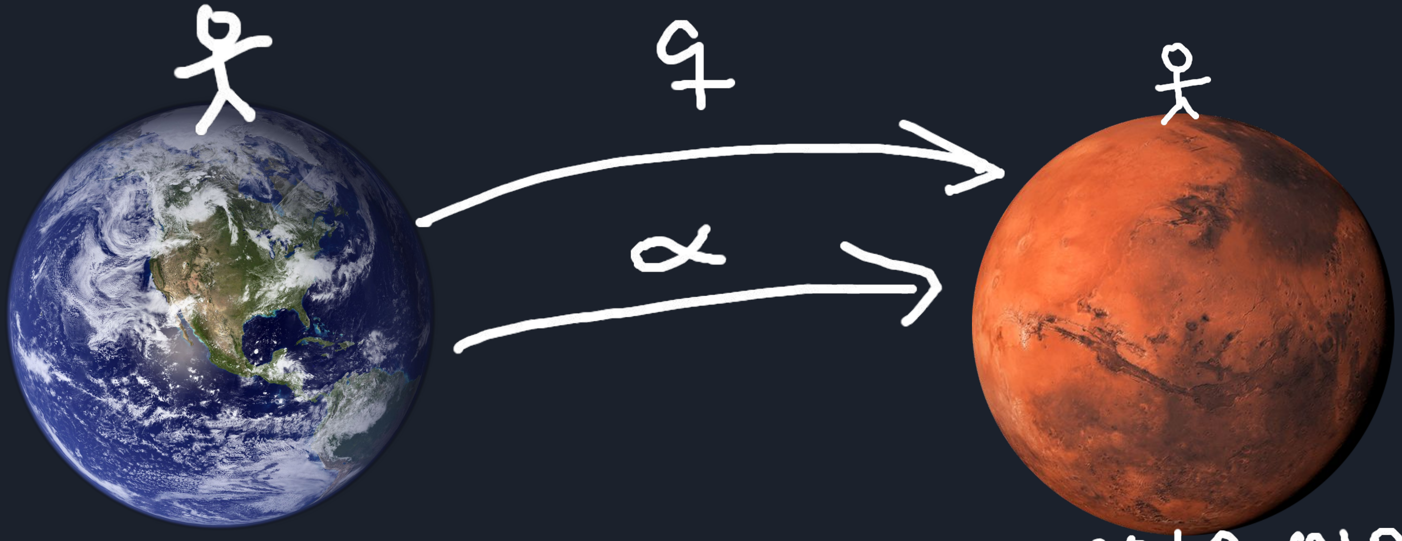
$a = 1010\ 0010$

$b = 0010\ 0101$

$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$

$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$

Budget:
O(log(n))
O(log(n))



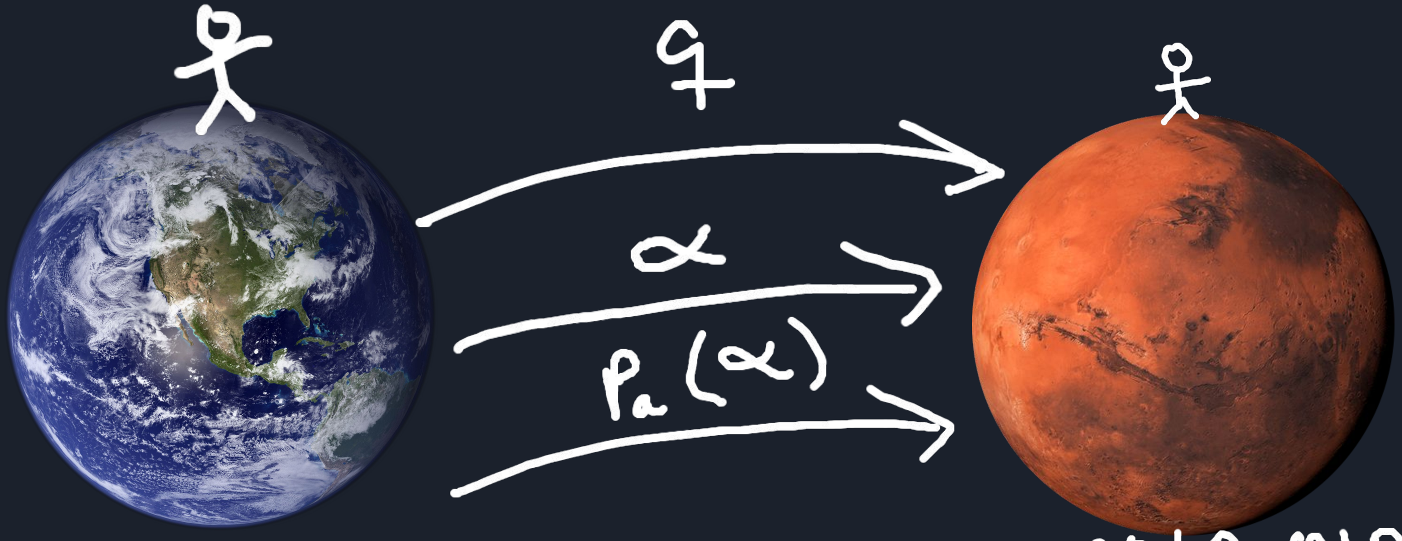
$$a = 1010\ 0010$$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$$b = 0010\ 0101$$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))
O(log(n))
O(log(n))



$a = 1010\ 0010$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$b = 0010\ 0101$

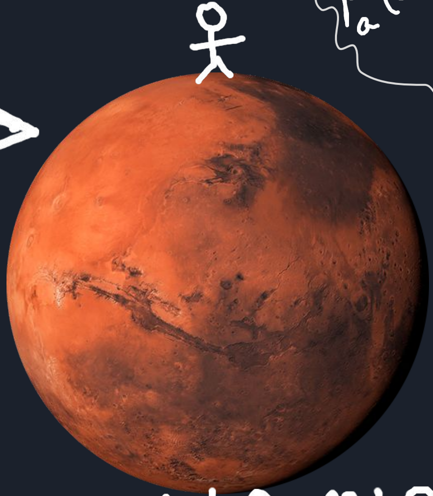
$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))
O(log(n))
O(log(n))

$\xrightarrow{q}$

$\xrightarrow{\alpha}$

$\xrightarrow{p_a(\alpha)}$



$p_a(\alpha) \stackrel{?}{=} p_b(\alpha)$

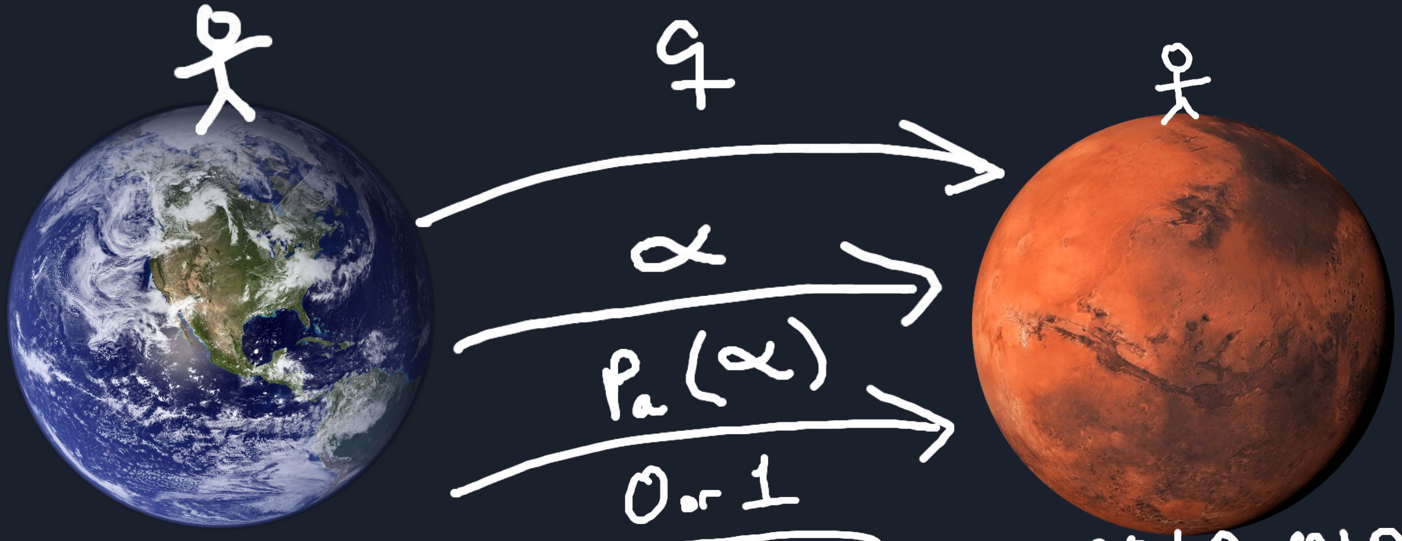
$a = 1010\ 0010$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$b = 0010\ 0101$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))
O(log(n))
O(log(n))
O(1)



$a = 1010\ 0010$

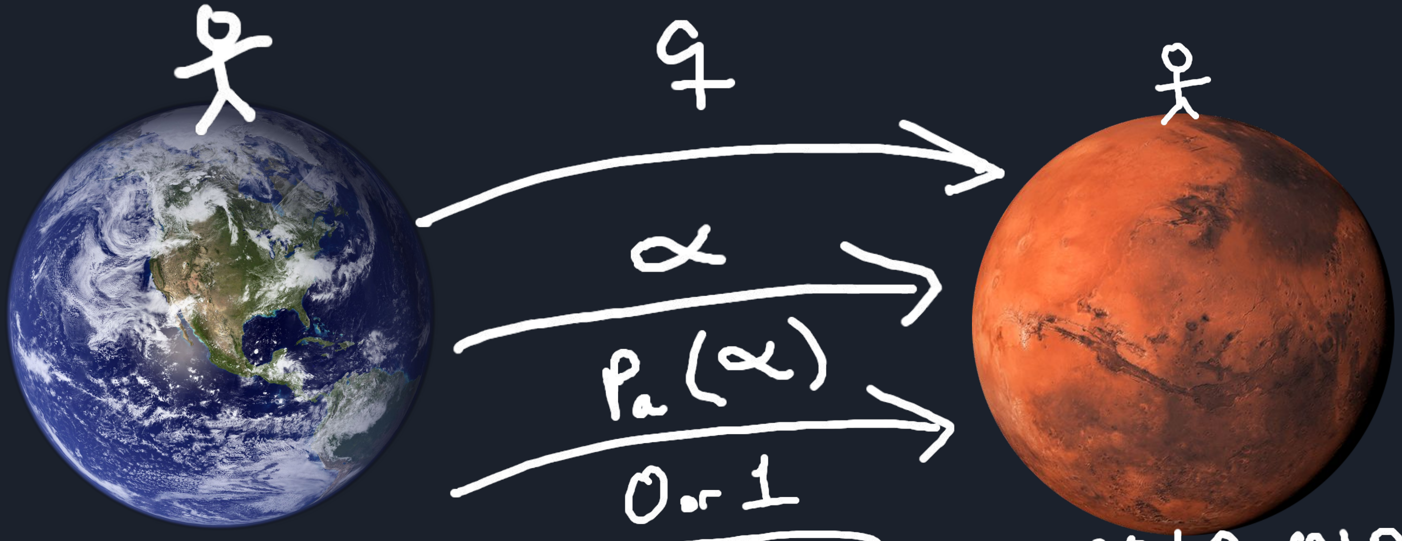
$b = 0010\ 0101$

$$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$$

$$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$$

Budget:
O(log(n))
O(log(n))
O(log(n))
O(1)

O(log(n))



$a = 1010\ 0010$

$b = 0010\ 0101$

$p_a = 1x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 0x^6 + 1x^7 + 0x^8$

$p_b = 0x + 0x^2 + 1x^3 + 0x^4 + 0x^5 + 1x^6 + 0x^7 + 1x^8$

if $a = b$, $\rho_a = \rho_b$

So $\forall \alpha .\ \rho_a \alpha = \rho_b \alpha$

So 100% of the time,

Bob tells Alyss

YES

if $a = b$, $\rho_a = \rho_b$

So $\forall \alpha$. $\rho_a \alpha = \rho_b \alpha$

So 100% of the time,

Bob tells Alyss

YES

If $a \neq b$, $\rho_a \neq \rho_b$

now $\rho_a \alpha = \rho_b \alpha$

$\iff (\rho_a - \rho_b)(\alpha) = 0$

if $a = b$, $p_a = p_b$

So $\forall \alpha$. $p_a \alpha = p_b \alpha$

So 100% of the time,
Bob tells Alyss
YES

If $a \neq b$, $p_a \neq p_b$

now $p_a \alpha = p_b \alpha$

$\iff (p_a - p_b)(\alpha) = 0$

but $\leqslant n$ α are roots of a polynomial of degree n.

if $a = b$, $P_a = P_b$

So $\forall \alpha.\ P_a \alpha = P_b \alpha$

So 100% of the time, Bob tells Alyss YES

If $a \neq b$, $P_a \neq P_b$

now $P_a \alpha = P_b \alpha$

$\iff (P_a - P_b)(\alpha) = 0$

but $\leqslant n$ α are roots of a polynomial of degree n.

so $P_r[\text{false "yes"}]$

$= P_r[\alpha \text{ root of } P_a - P_b]$

$\leqslant \dfrac{\deg(P_a - P_b)}{q} \leqslant \dfrac{n}{n^2} = \dfrac{1}{n}$

if $a = b$, $p_a = p_b$

So $\forall \alpha .\ p_a \alpha = p_b \alpha$

So 100% of the time, Bob tells Alyss YES

If $a \neq b$, $p_a \neq p_b$

now $p_a \alpha = p_b \alpha$

$$\iff (p_a - p_b)(\alpha) = 0$$

but $\leq n$ α are roots of a polynomial of degree n.

so $\Pr[\text{false "yes"}]$

$$= \Pr[\alpha \text{ root of } p_a - p_b]$$

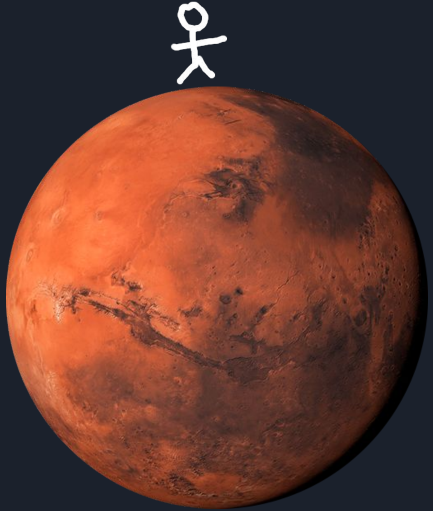
$$\leq \frac{\deg(p_a - p_b)}{q} \leq \frac{n}{n^2} = \frac{1}{n}$$

Can we do better?

Can we do better?

Yes! (If we cheat a little)

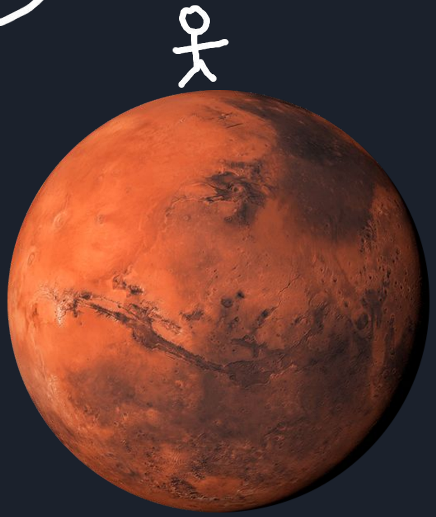
$a \in \{0,1\}^n$



$b \in \{0,1\}^n$

Random Bits In the Sky.

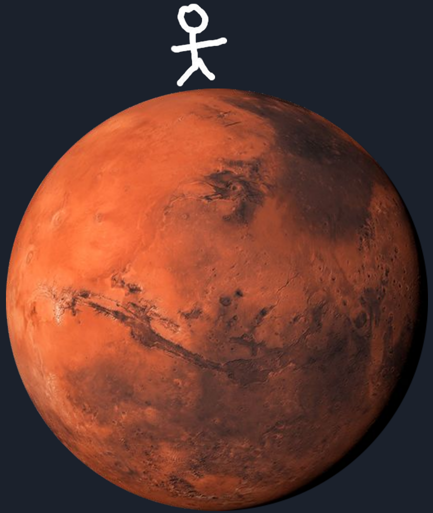
$a \in \{0,1\}^n$



$b \in \{0,1\}^n$

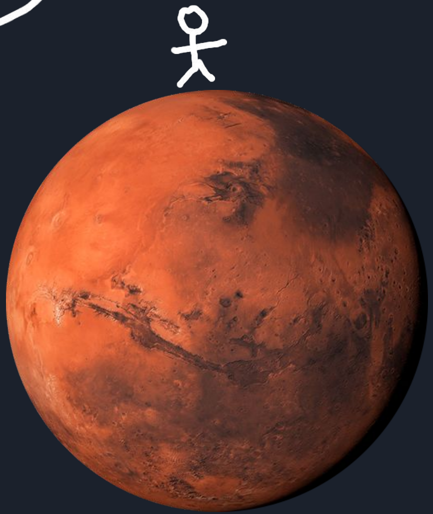
5,778.029364.... K

$a \in \{0,1\}^n$



$b \in \{0,1\}^n$

Random Bits In the Sky.



$a \in \{0,1\}^n$

$b \in \{0,1\}^n$

Theorem:

In the “public randomness” model, Alyss and Bob can solve Eq_n with probability of a false positive < 25% using…

~ Audience Participation ~

Theorem:

In the “public randomness” model, Alyss and Bob can solve Eq_n with probability of a false positive < 25% using…

3 bits of communication!

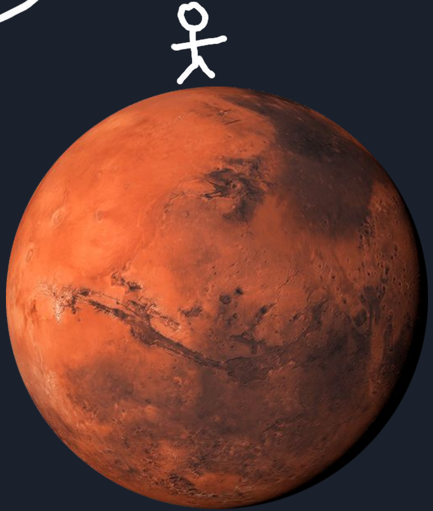
Theorem:

In the “public randomness” model, Alyss and Bob can solve Eq_n with probability of a false positive $< \epsilon$ using $O(\log(1/\epsilon))$ bits of communication

Uniform in n!

Random Bits In the Sky.

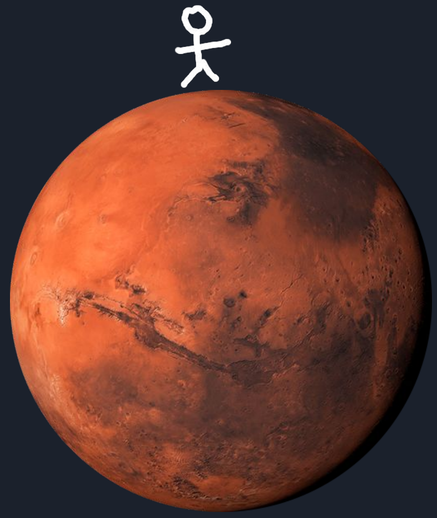
$a \in \{0,1\}^n$



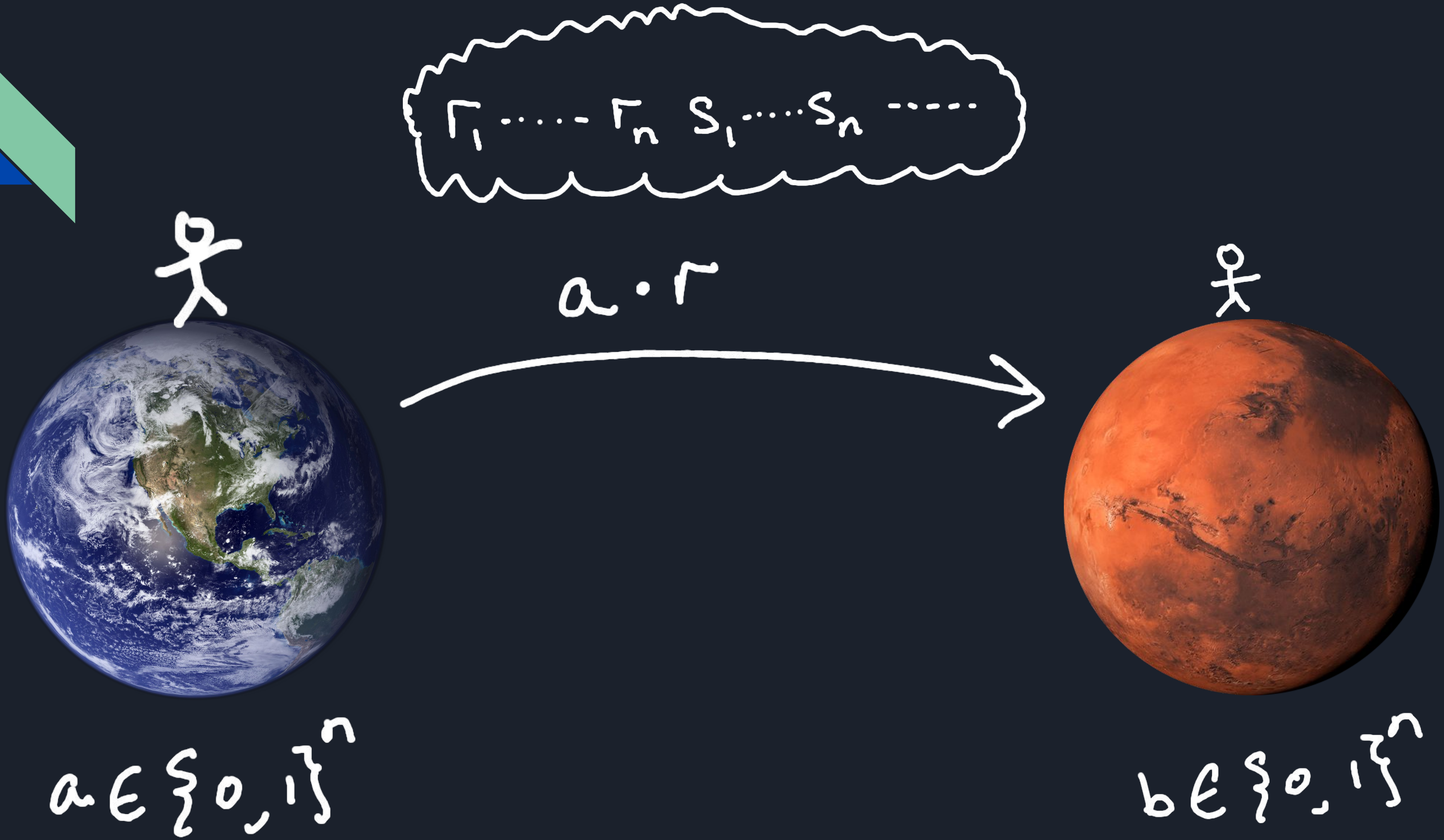
$b \in \{0,1\}^n$

$r_1 \cdots r_n \quad s_1 \cdots s_n \cdots$

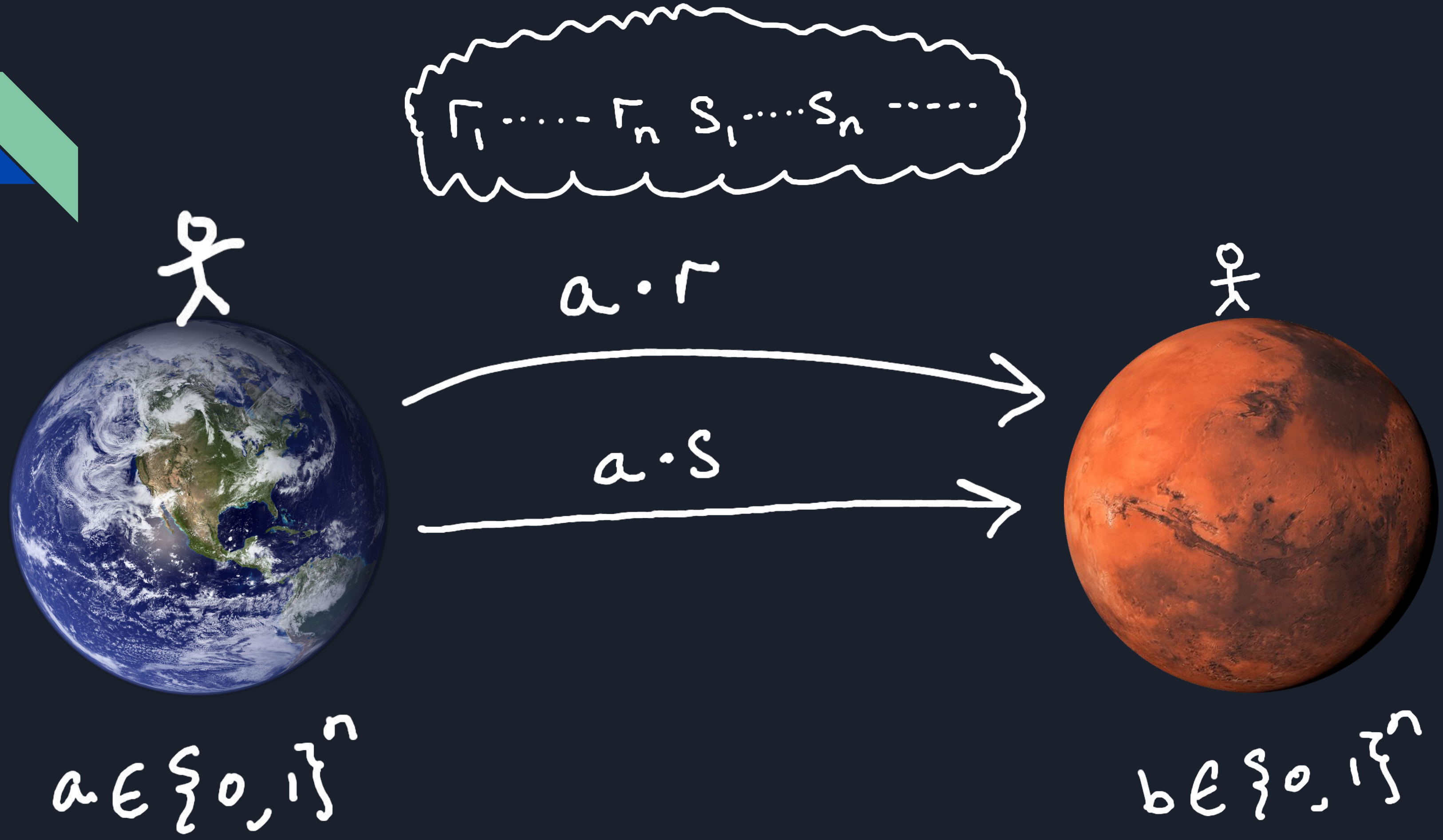
$a \in \{0,1\}^n$

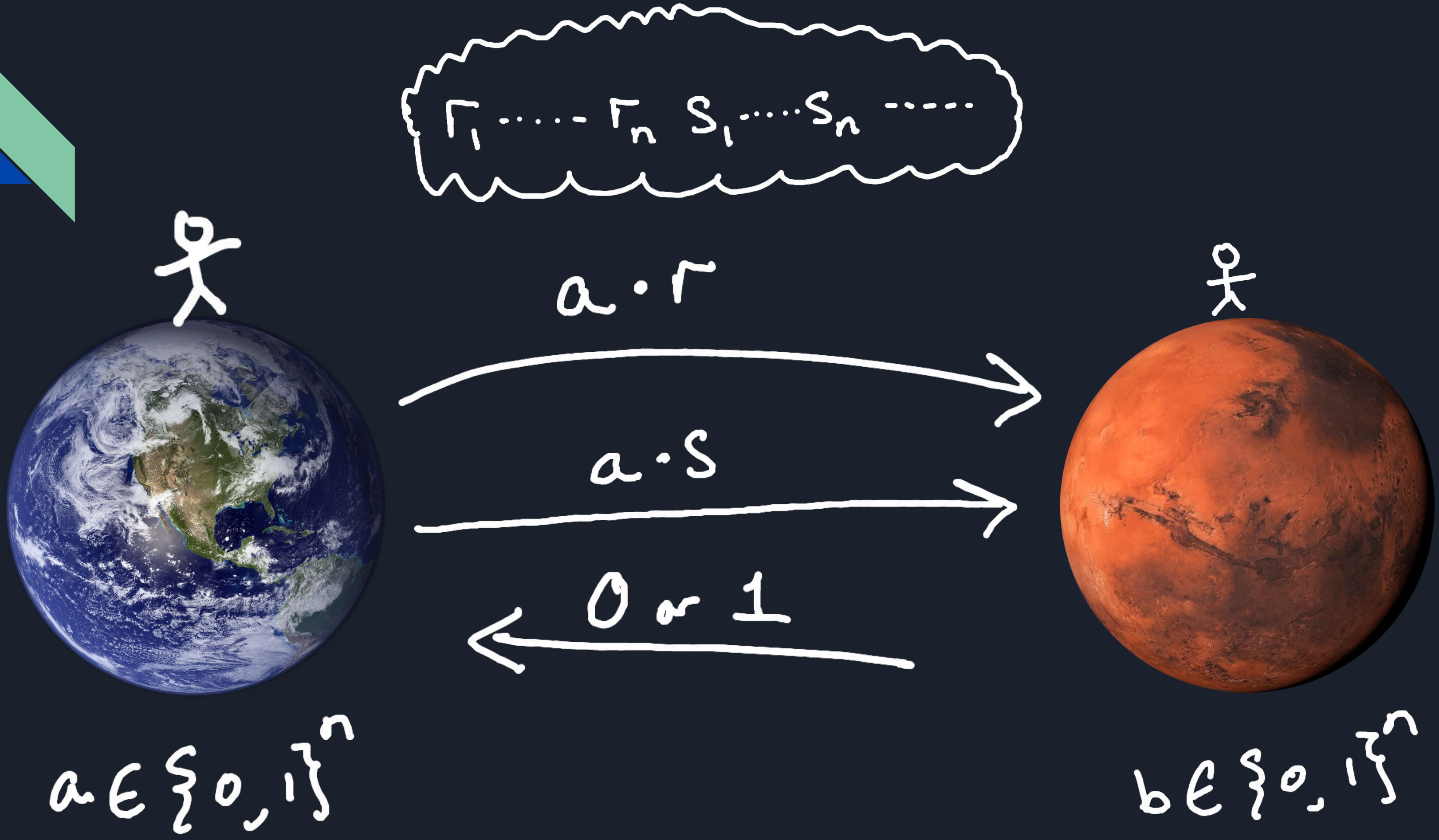


$b \in \{0,1\}^n$



$r_1 \cdots\cdots r_n \; s_1 \cdots\cdots s_n \cdots\cdots$
$a \cdot r$
$a \in \{0,1\}^n$
$b \in \{0,1\}^n$


$r_1 \cdots\cdots r_n \; s_1 \cdots\cdots s_n$ -----
$a \cdot r$
$a \cdot s$
$a \in \{0,1\}^n$
$b \in \{0,1\}^n$


$r_1 \cdots\cdots r_n \; s_1 \cdots\cdots s_n \cdots\cdots$
$a \cdot r$
$a \cdot s$
0 or 1
$a \in \{0,1\}^n$
$b \in \{0,1\}^n$

If $a = b$, clearly $\forall r.\ a \cdot r = b \cdot r$

Otherwise, note

$$\Pr[a \cdot r = 1] = \frac{1}{2}.$$

So $\Pr[a \cdot r = b \cdot r] = \frac{1}{2}$

So $\Pr[a \cdot r = b \cdot r \ \& \ a \cdot s = b \cdot s] = \frac{1}{4}$.

As we said, though:
25% isn't special.

If you want Pr[☹] < 0.01,
then sending 7 bits
is always enough:

$$\frac{1}{2^7} = \frac{1}{128} < \frac{1}{100} = 0.01$$

In general, to get

$\Pr[\text{☹}] < \varepsilon$, we need

$$2^{-\#\text{messages}} < \varepsilon$$

In general, to get

$\Pr[☹] < \varepsilon$, we need

$$2^{-\#\text{messages}} < \varepsilon$$

$$\left(\iff \#\text{messages} > \log(1/\varepsilon)\right)$$

Ok… But how much are we cheating by?

What about private coin complexity?

Newman’s Theorem

- Before we wrote D(f) for the **D**eterministic communication complexity of f

Newman’s Theorem

- Before we wrote D(f) for the **D**eterministic communication complexity of f
- Now let’s write $R_{\epsilon}^{pub}(f)$ for the **R**andomized (public) communication complexity

Newman's Theorem

- Before we wrote D(f) for the **D**eterministic communication complexity of f
- Now let's write $R_{\epsilon}^{pub}(f)$ for the **R**andomized (public) communication complexity
- Similarly, we write $R_{\epsilon}^{priv}(f)$ for the **R**andomized (private) communication complexity

Newman's Theorem

- Before we wrote D(f) for the **D**eterministic communication complexity of f
- Now let's write $R_{\epsilon}^{pub}(f)$ for the **R**andomized (public) communication complexity
- Similarly, we write $R_{\epsilon}^{priv}(f)$ for the **R**andomized (private) communication complexity
- (In both, the subscript ϵ indicates the tolerance for errors)

Theorem (Newman):

$$R_{\epsilon+\delta}^{priv}(f) \leq R_{\epsilon}^{pub}(f) + O(\log(n) + \log(1/\delta))$$

Theorem (Newman): $R_{\epsilon+\delta}^{priv}(f) \leq R_{\epsilon}^{pub}(f) + O(\log(n) + \log(1/\delta))$

Says, at the cost of a little bit more error, and log(n) extra messages, we can turn a public randomness protocol into a private coin protocol.

Theorem (Newman):
$R_{\epsilon+\delta}^{priv}(f) \leq R_{\epsilon}^{pub}(f) + O(\log(n) + \log(1/\delta))$

Says, at the cost of a little bit more error, and log(n) extra messages, we can turn a public randomness protocol into a private coin protocol.

This agrees with our result from earlier:
In the "polynomial protocol" only Alyss needed randomness, and we needed O(log(n)) many messages.

Theorem (Newman):
$R_{\epsilon+\delta}^{priv}(f) \leq R_{\epsilon}^{pub}(f) + O(\log(n) + \log(1/\delta))$

Says, at the cost of a little bit more error, and log(n) extra messages, we can turn a public randomness protocol into a private coin protocol.

This agrees with our result from earlier:
In the “polynomial protocol” only Alyss needed randomness, and we needed O(log(n)) many messages.

In the public protocol we just discussed, though, we got it down to O(1) messages. So we have a disparity of O(log(n)).

Theorem (Newman):
$R_{\epsilon+\delta}^{priv}(f) \le R_{\epsilon}^{pub}(f) + O(\log(n) + \log(1/\delta))$

Says, at the cost of a little bit more error, and log(n) extra messages, we can turn a public randomness protocol into a private coin protocol.

This agrees with our result from earlier:
In the “polynomial protocol” only Alyss needed randomness, and we needed O(log(n)) many messages.

In the public protocol we just discussed, though, we got it down to O(1) messages. So we have a disparity of O(log(n)).

Newman’s Theorem says we will never do worse than this.

For more information, see Ryan O’Donnell’s “CS Theory Toolkit” on Youtube.

I’ll link the playlist on my blog post for this talk at grossack.site

Thank You! ^_^