How to find the limits of algorithms?

Anup Rao

Computers

Computers

are really awesome,

Computers

are really awesome,

but is there something they cannot do?

Yes! [Godel,Turing 1930’s]

Yes! [Godel,Turing 1930's]

<u>Halting problem</u>

$$\underset{\text{program code}}{\mathrm{Halt}(\,P}, \underset{\text{input}}{x}) = \begin{cases} 1 & \text{if } P(x) \text{ halts} \\ 0 & \text{if } P(x) \text{ runs forever} \end{cases}$$

Yes! [Godel,Turing 1930's]

Halting problem

$$\text{Halt}(\underset{\text{program code}}{P}, \underset{\text{input}}{x}) = \begin{cases} 1 & \text{if } P(x) \text{ halts} \\ 0 & \text{if } P(x) \text{ runs forever} \end{cases}$$

Thm: No program can compute Halt(P,x)

Yes! [Godel,Turing 1930's]

<u>Halting problem</u>

$$\underset{\substack{\text{program}\\\text{code}}}{\text{Halt}(\ P,}\ \underset{\text{input}}{x)} = \begin{cases} 1 & \text{if } P(x) \text{ halts} \\ 0 & \text{if } P(x) \text{ runs forever} \end{cases}$$

Thm: No program can compute Halt(P,x)

Pf: Suppose program H computes Halt. Define program G:

$$G(P) = \begin{cases} 0 & \text{if } H(P,P) \text{ outputs } 0 \\ \text{loop forever} & \text{if } H(P,P) \text{ outputs } 1 \end{cases}$$

Yes! [Godel,Turing 1930's]

Halting problem

$$\mathrm{Halt}(\underset{\text{program code}}{P}, \underset{\text{input}}{x}) = \begin{cases} 1 & \text{if } P(x) \text{ halts} \\ 0 & \text{if } P(x) \text{ runs forever} \end{cases}$$

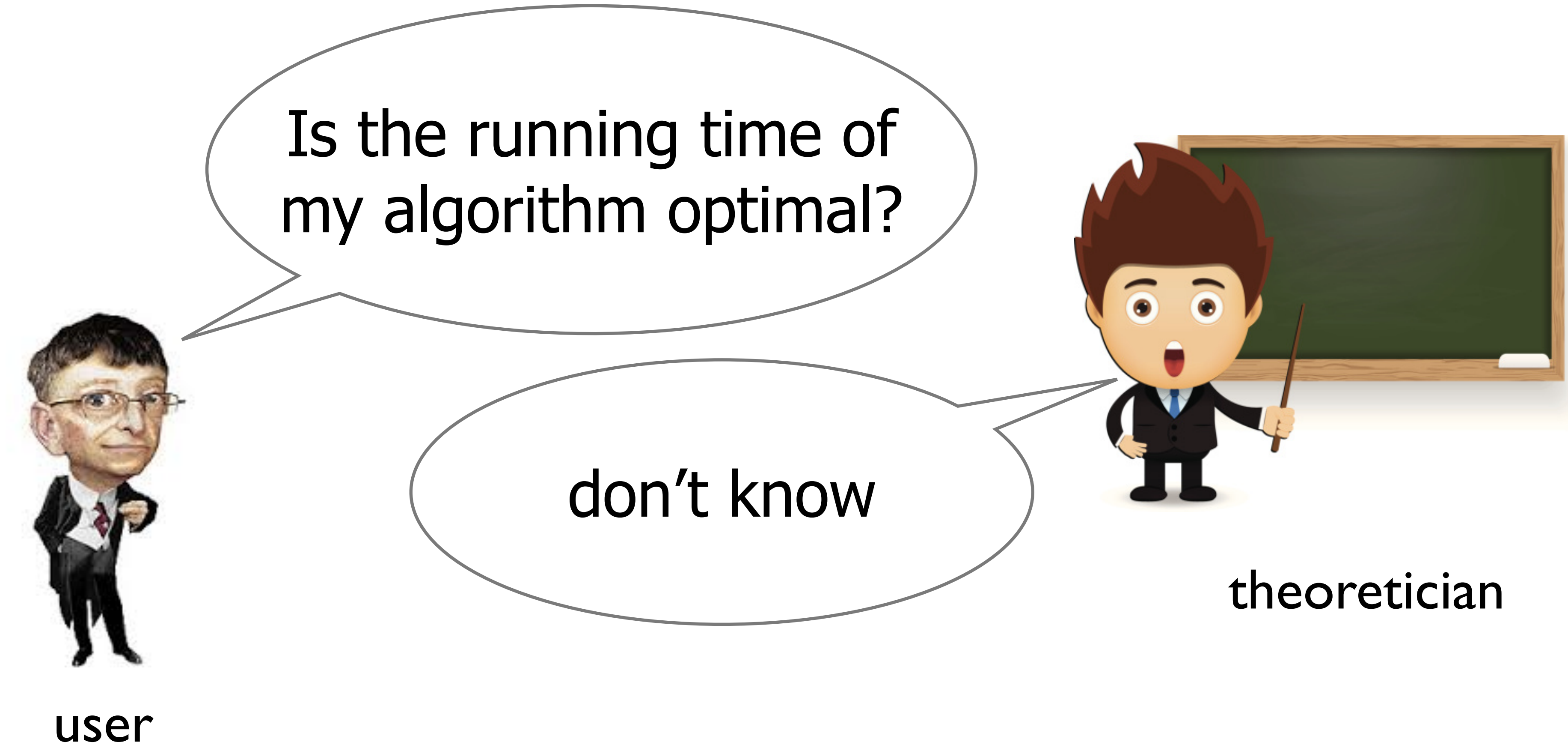
Thm: No program can compute Halt(P,x)

Pf: Suppose program H(P,x) computes Halt. Let

$$G(P) = \begin{cases} 0 & \text{if } H(P,P) \text{ outputs } 0 \\ \text{loop forever} & \text{if } H(P,P) \text{ outputs } 1 \end{cases}$$

If G(G) = 0, then H(G,G)=0, so H has a bug.
If G(G) loops forever, then H(G,G)=1, so H has a bug.

The biggest gap in our understanding of algorithms



The biggest gap in our understanding of algorithms

Is my algorithm for matrix multiplication optimal?

user

don't know

theoretician

The biggest gap in our understanding of algorithms

Is my algorithm for multiplying two numbers optimal?

don't know

theoretician

user

The biggest gap in our understanding of algorithms

Is my
algorithm for SAT
optimal?

user

That's the famous P vs
NP question. We've given it
a lot of thought, and
we don't know.

theoretician

What we do know

linear time algorithms are optimal

you have to read all the input

diagonalization (like halting)

does a given program stop in T steps? you need T steps to answer this

Common misconceptions

Sorting requires at least nlog(n) time

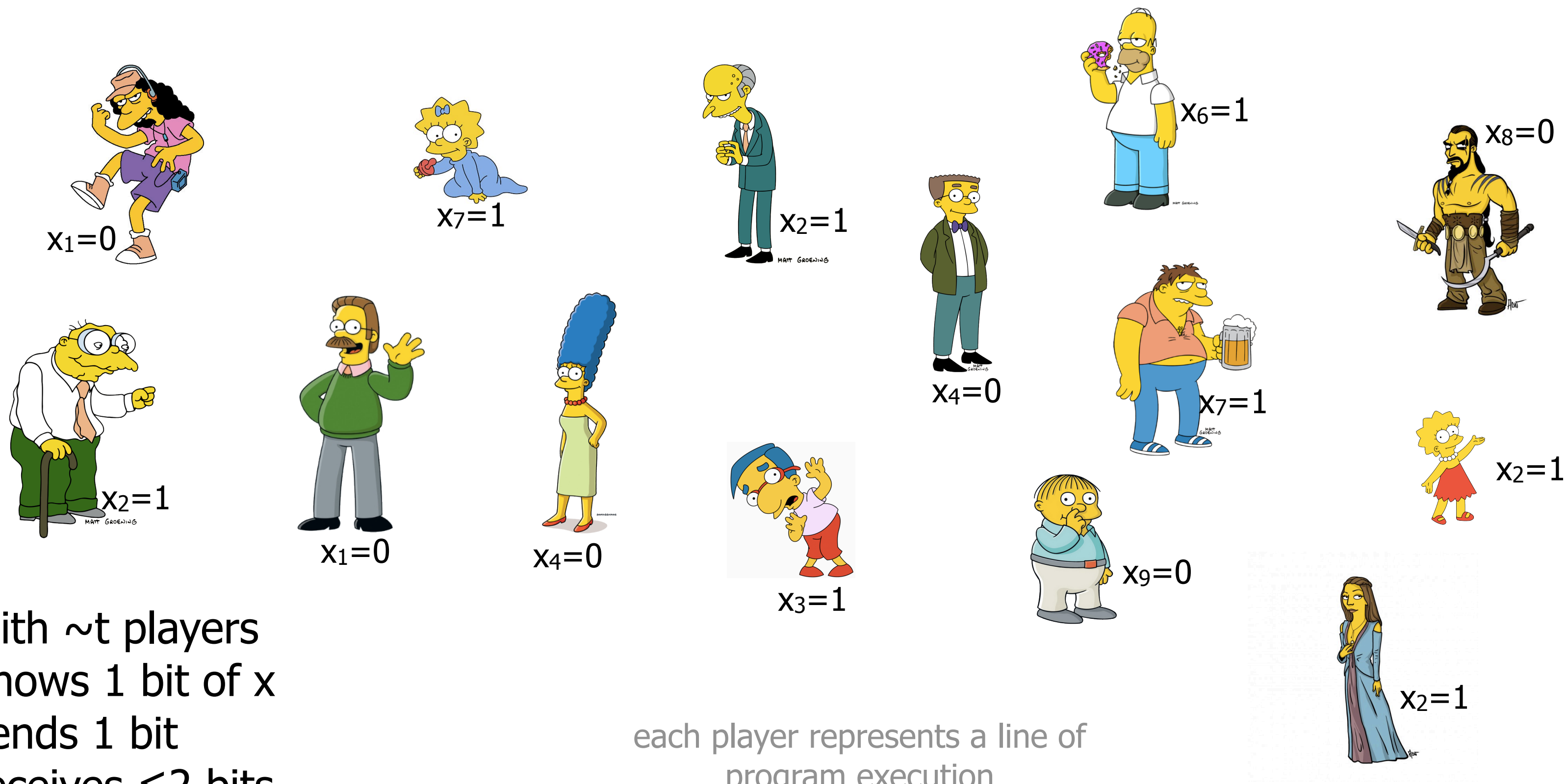
we don’t know this

How can we prove lowerbounds on running time?

Multiparty communication complexity

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,

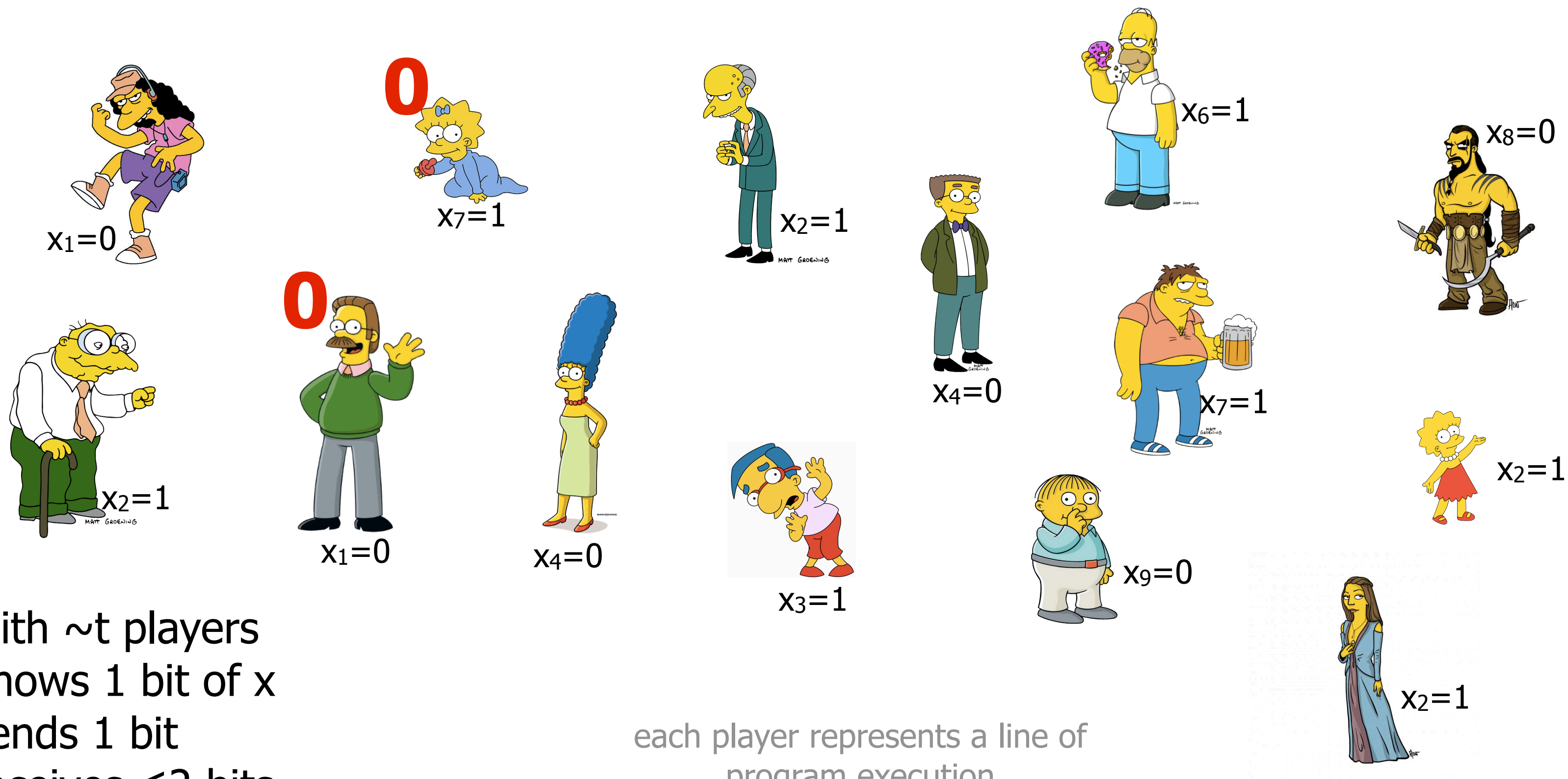


protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,



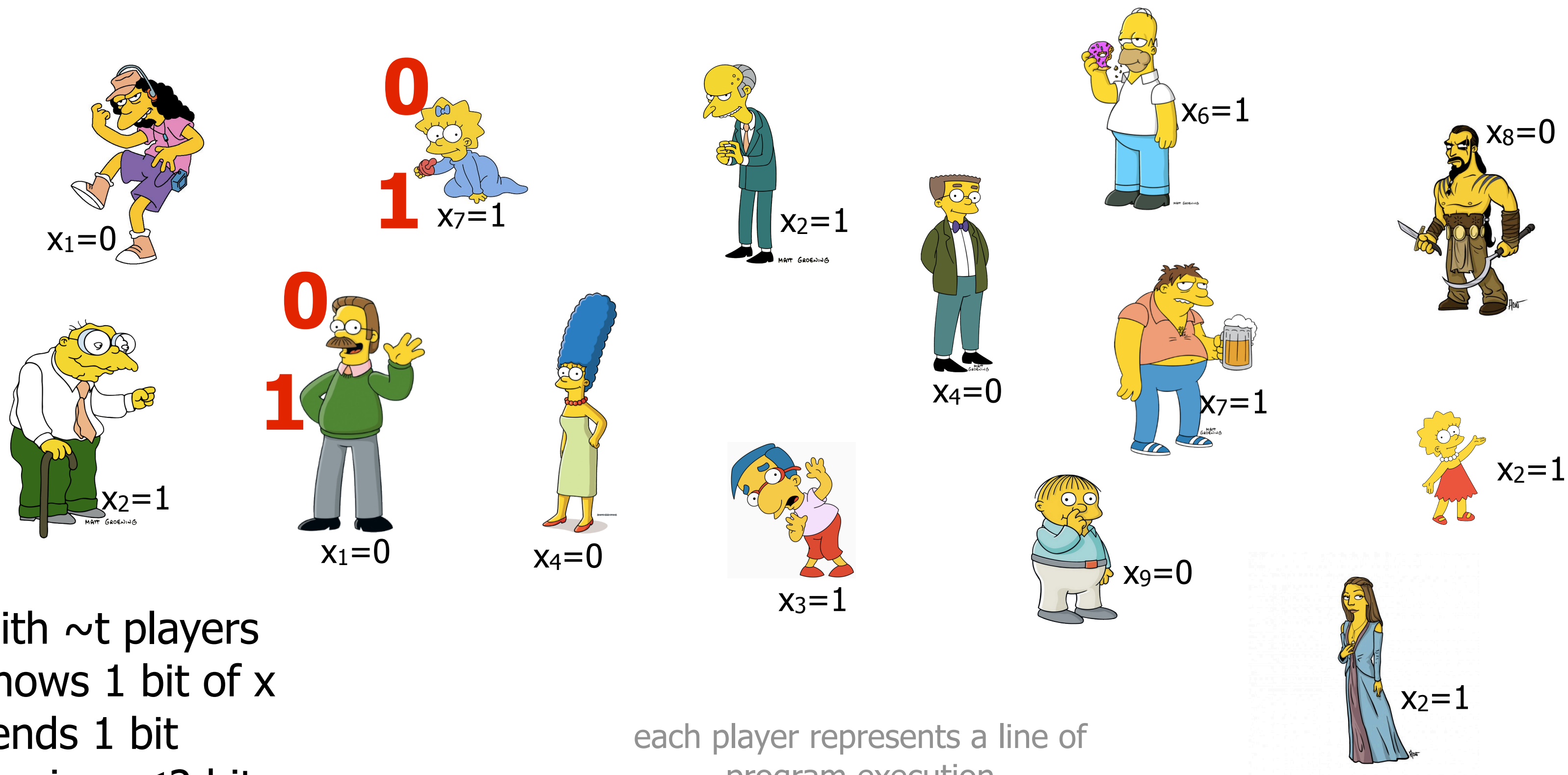
protocol with ~t players

- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,

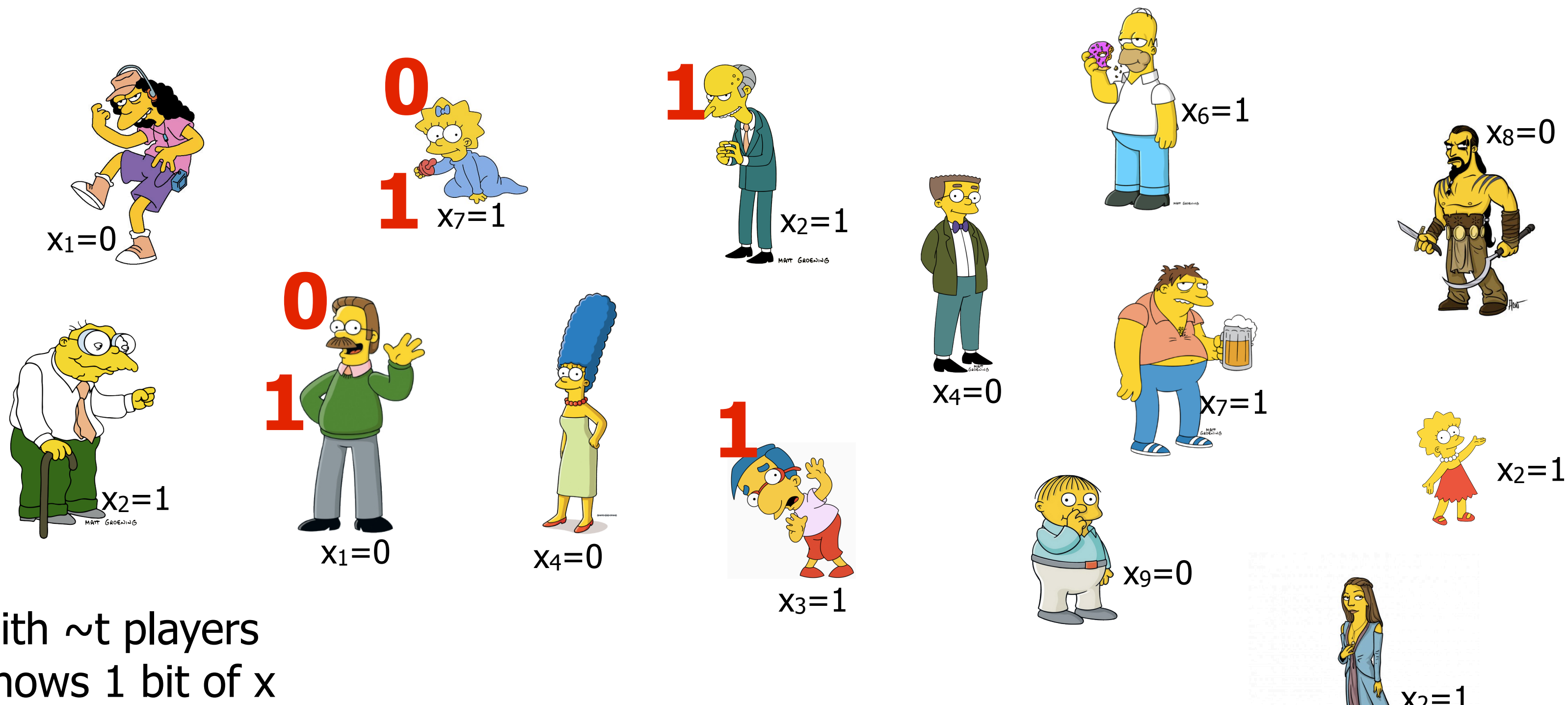


protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,

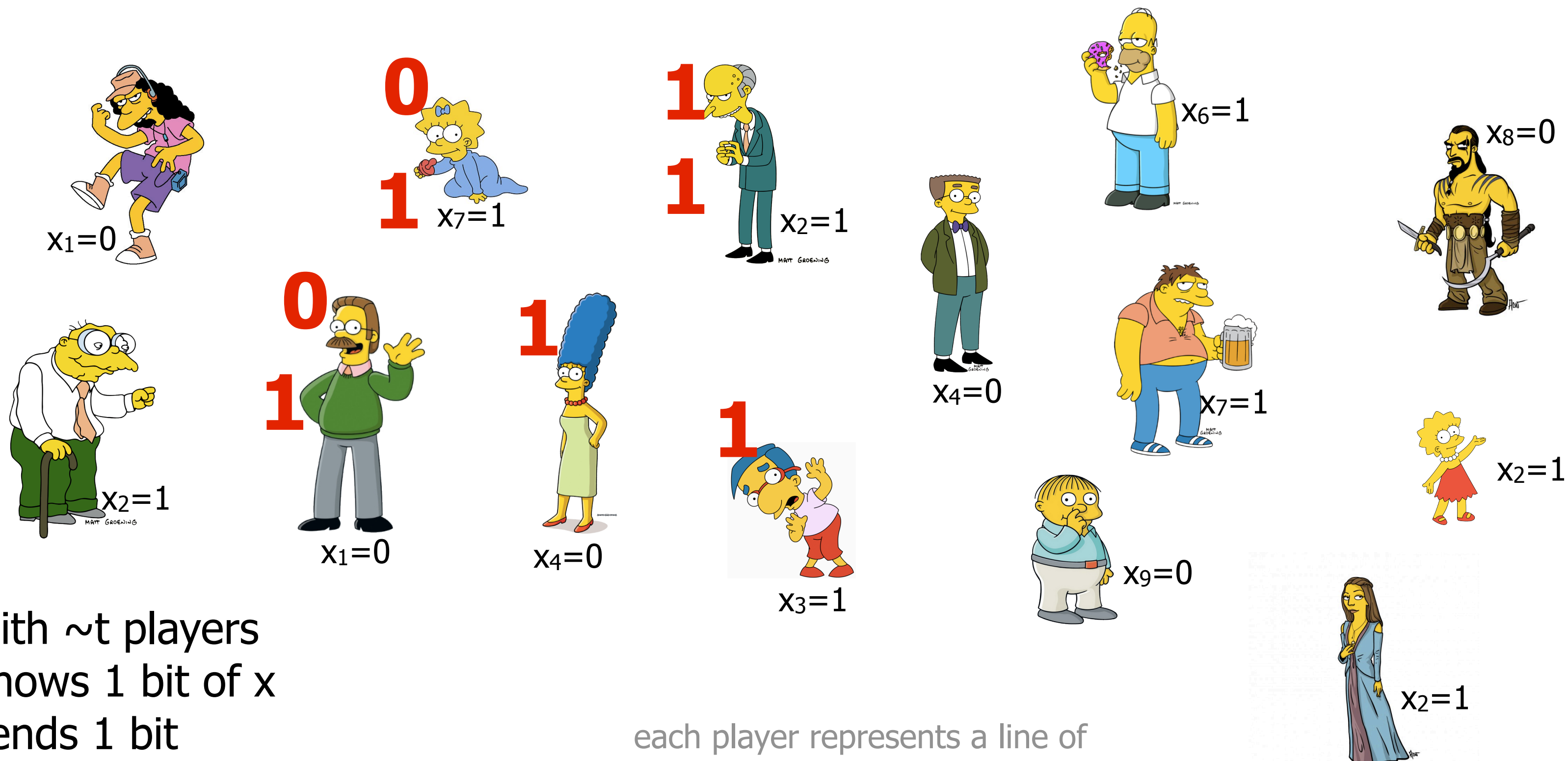


protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \to \{0,1\}$ can be computed in time t,



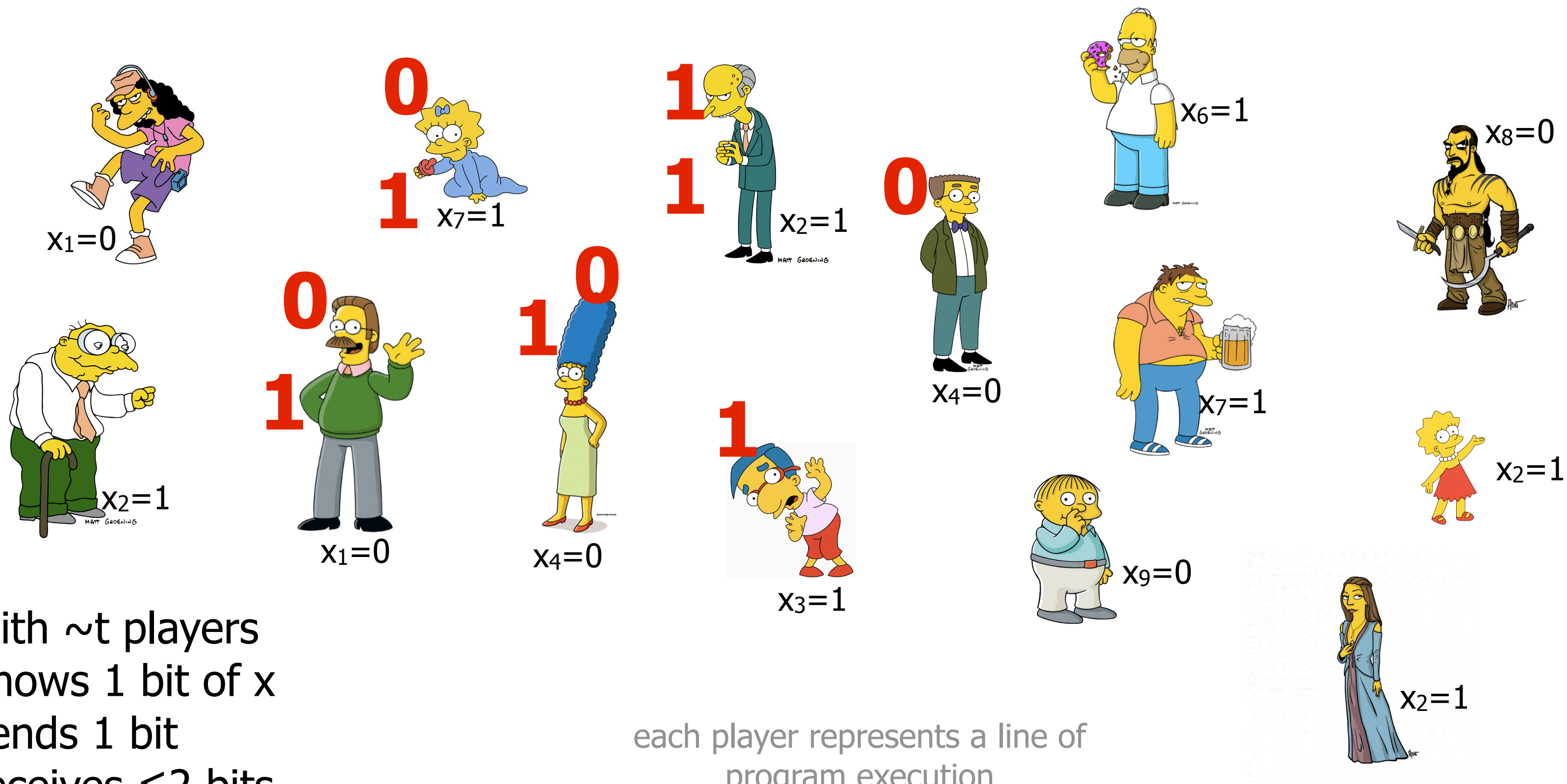
each player represents a line of program execution

protocol with ~t players

- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,

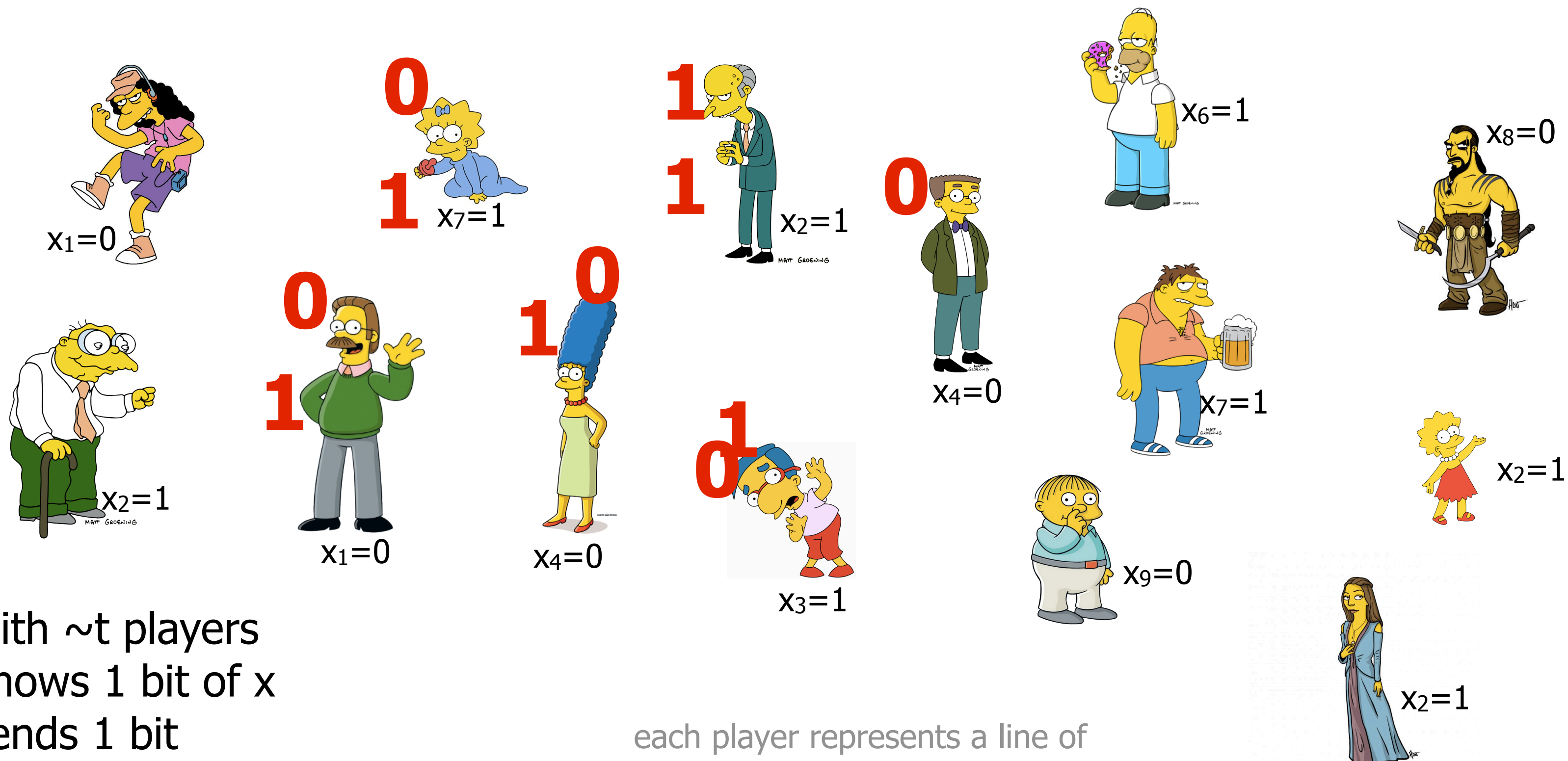


protocol with ~t players

- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,



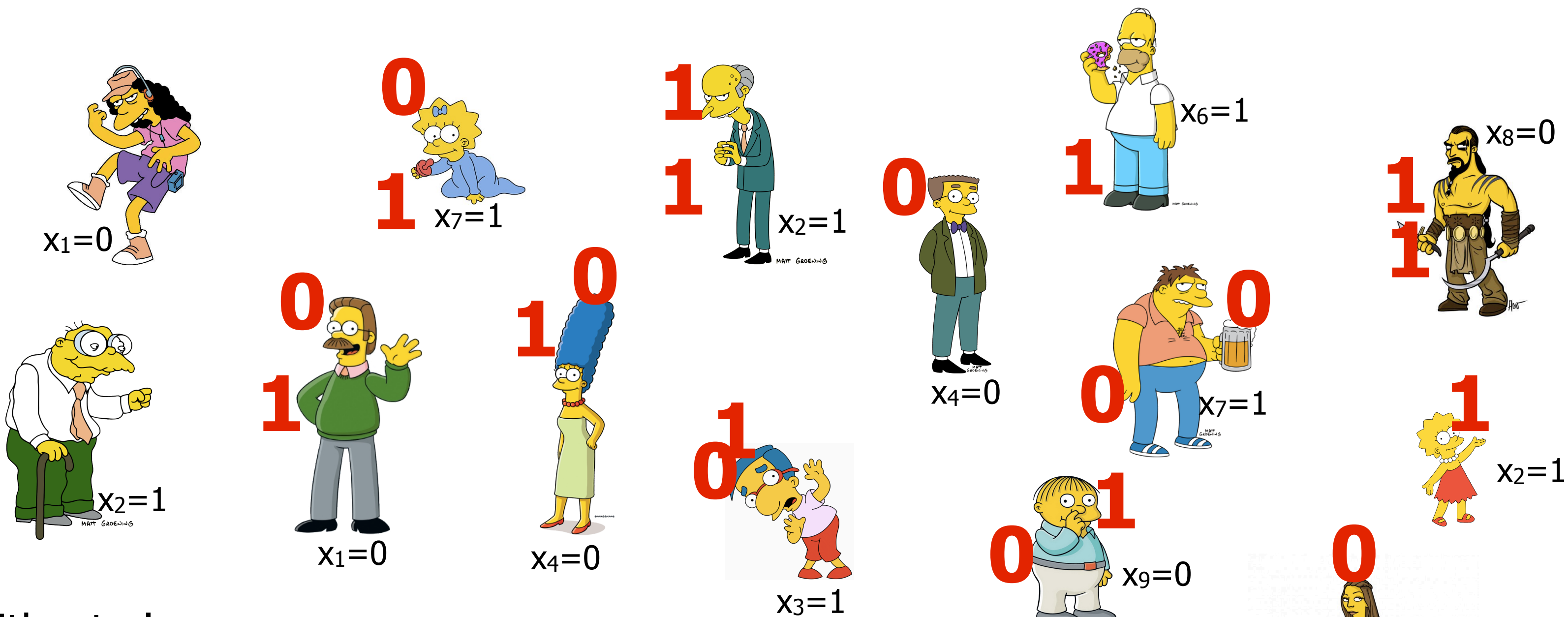
protocol with ~t players

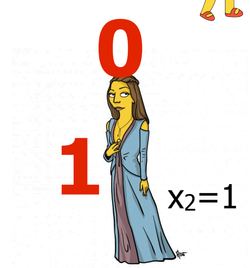
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,





protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

f(x)=1

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,

Remarks

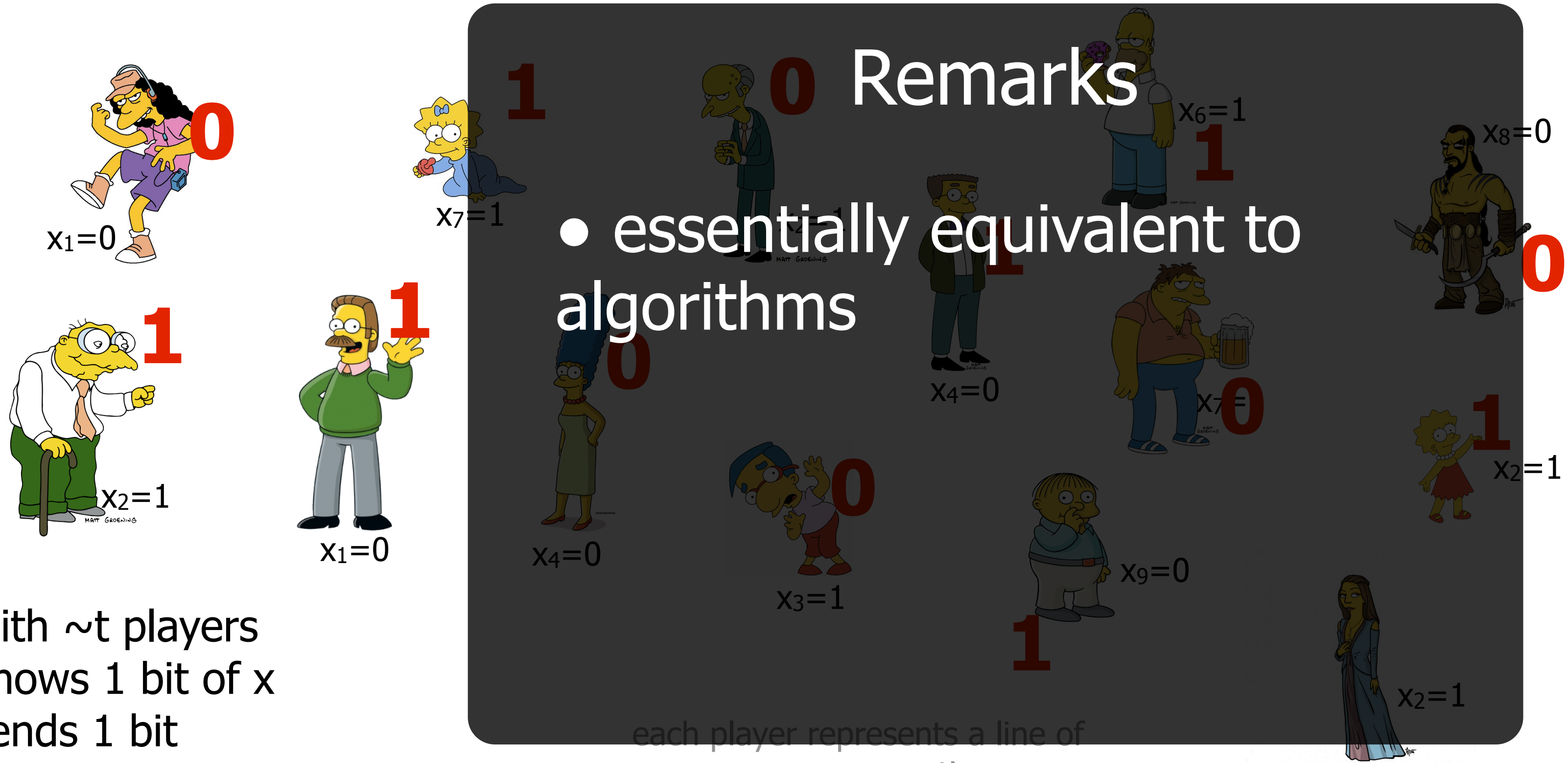
protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

each player represents a line of program execution

f(x)=1

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,



Remarks

- essentially equivalent to algorithms

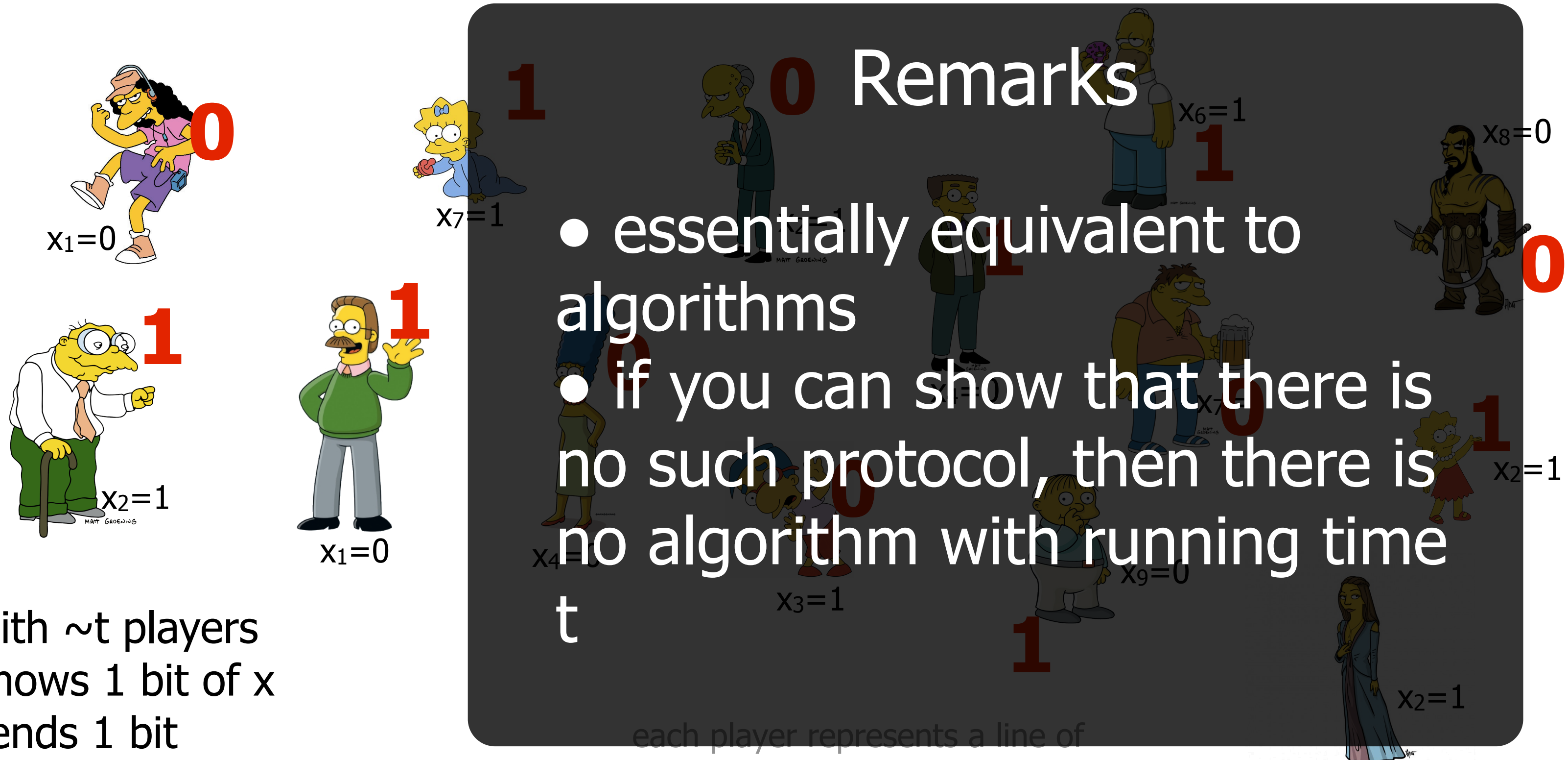
each player represents a line of program execution

protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

f(x)=1

Algorithms vs Multiparty Communication

If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in time t,



Remarks

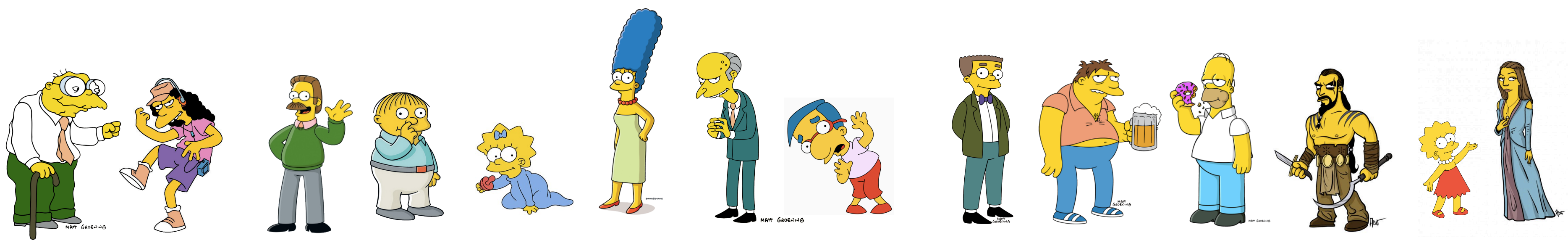
- essentially equivalent to algorithms
- if you can show that there is no such protocol, then there is no algorithm with running time t

protocol with ~t players
- player knows 1 bit of x
- player sends 1 bit
- player receives ≤2 bits
- at end, someone knows f(x)

f(x)=1

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,



$x=00100111010100100010111100010100 1$

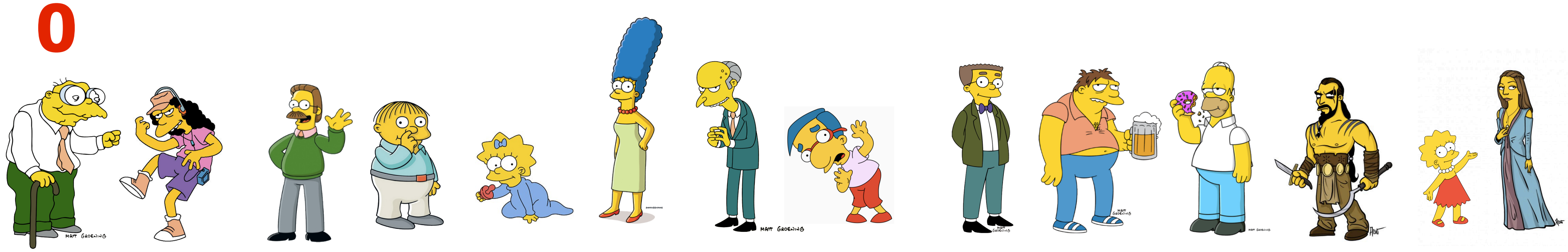
protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,

0



x=00100111$\boxed{0101001000101}$111000101001

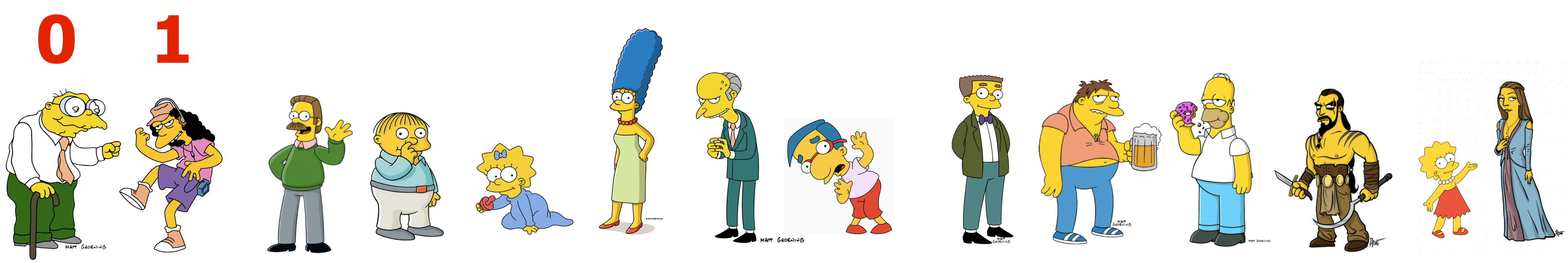
protocol with n/loglog(n) players

- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,



x=00100111010100100010111100010100 1

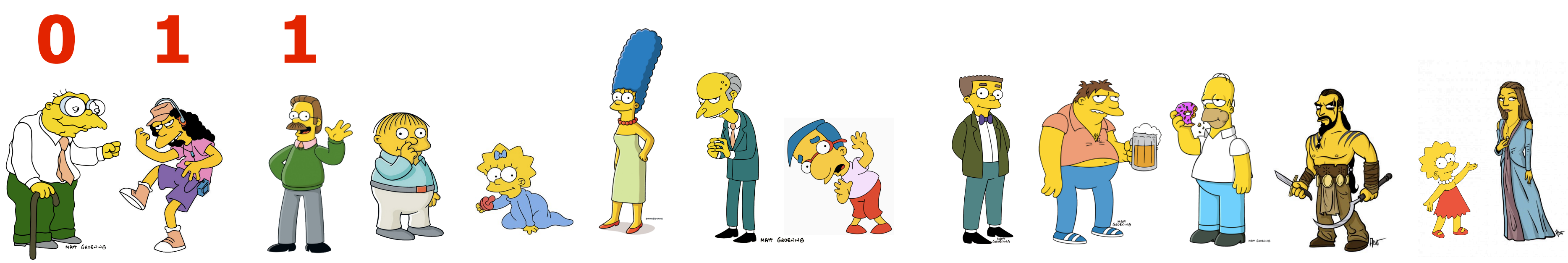
protocol with n/loglog(n) players

- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,



x=001001110101001000101111000101001

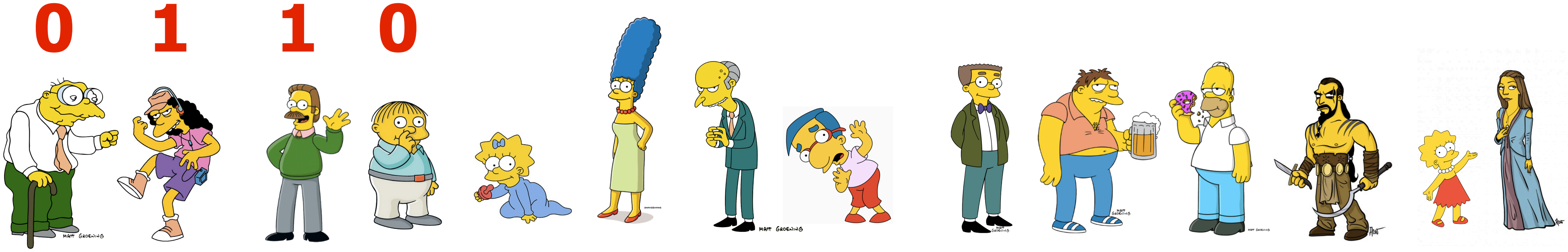
protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,

0 1 1 0



x=$\boxed{0010011101}$0100100010111100010100 1

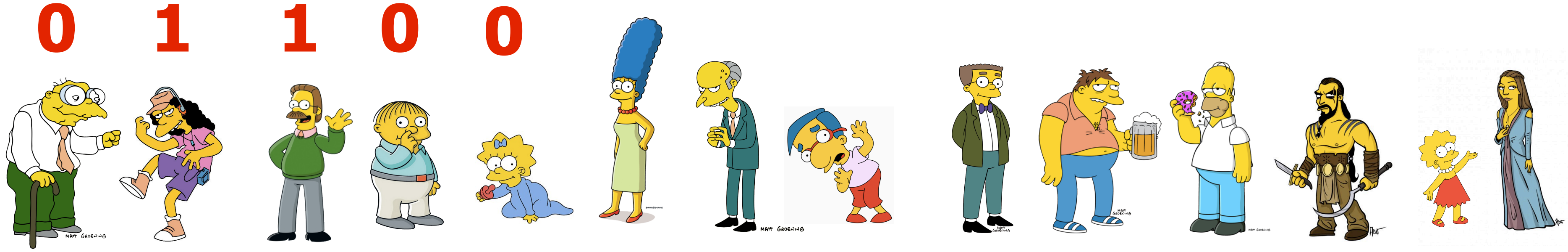
protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x){:}\{0,1\}^n \to \{0,1\}$ can be computed in parallel time O(log n), with total work O(n),

0 1 1 0 0



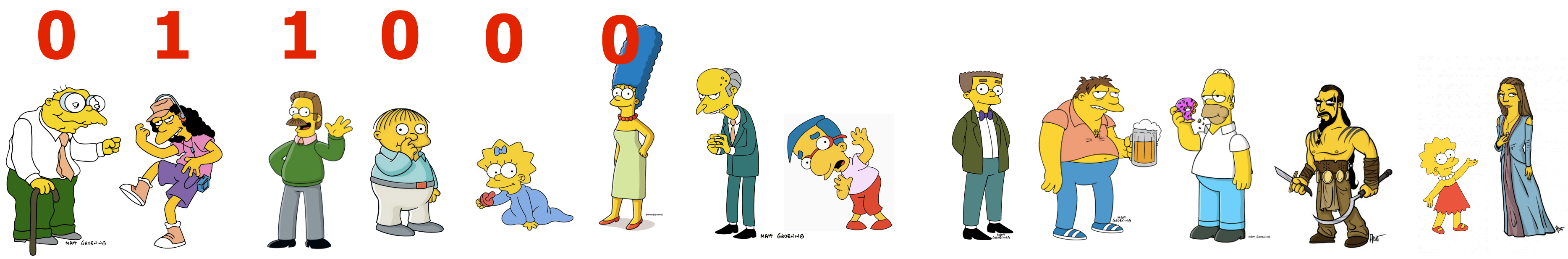
x=00100111010100100010111 1000101001

protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,



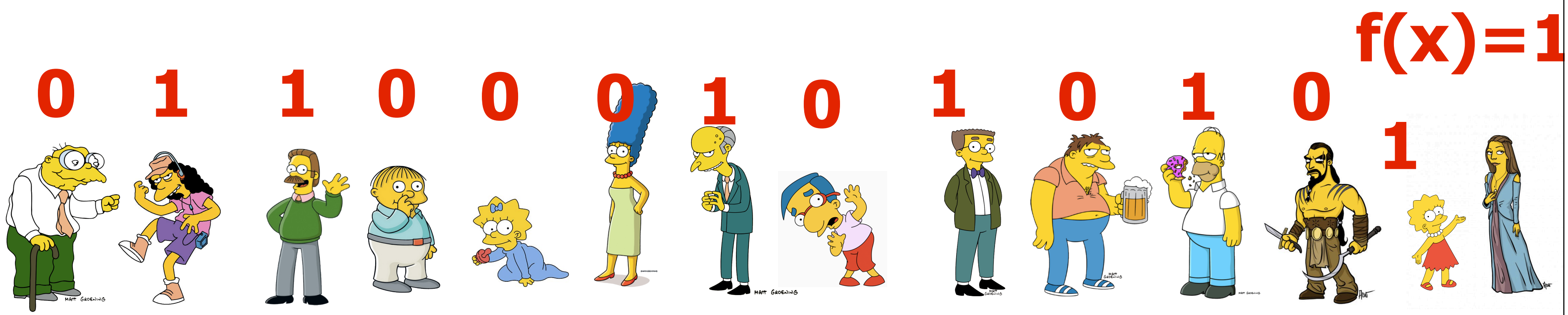
x=0010011101010010001011110001010 01

protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Algorithms vs Multiparty Communication

[V77, HR14] If $f(x):\{0,1\}^n \to \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,



x=001001110101001000101111000101001

protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

2 party communication

What is $f(X,Y)$?

X

n bits

Y

n bits

2 party communication

What is f(X,Y) ?

$$\xrightarrow{m_1(X)}$$

X

n bits

Y

n bits

2 party communication

What is $f(X,Y)$?

$m_1(X)$ →

$m_2(m_1,Y)$ ←

X

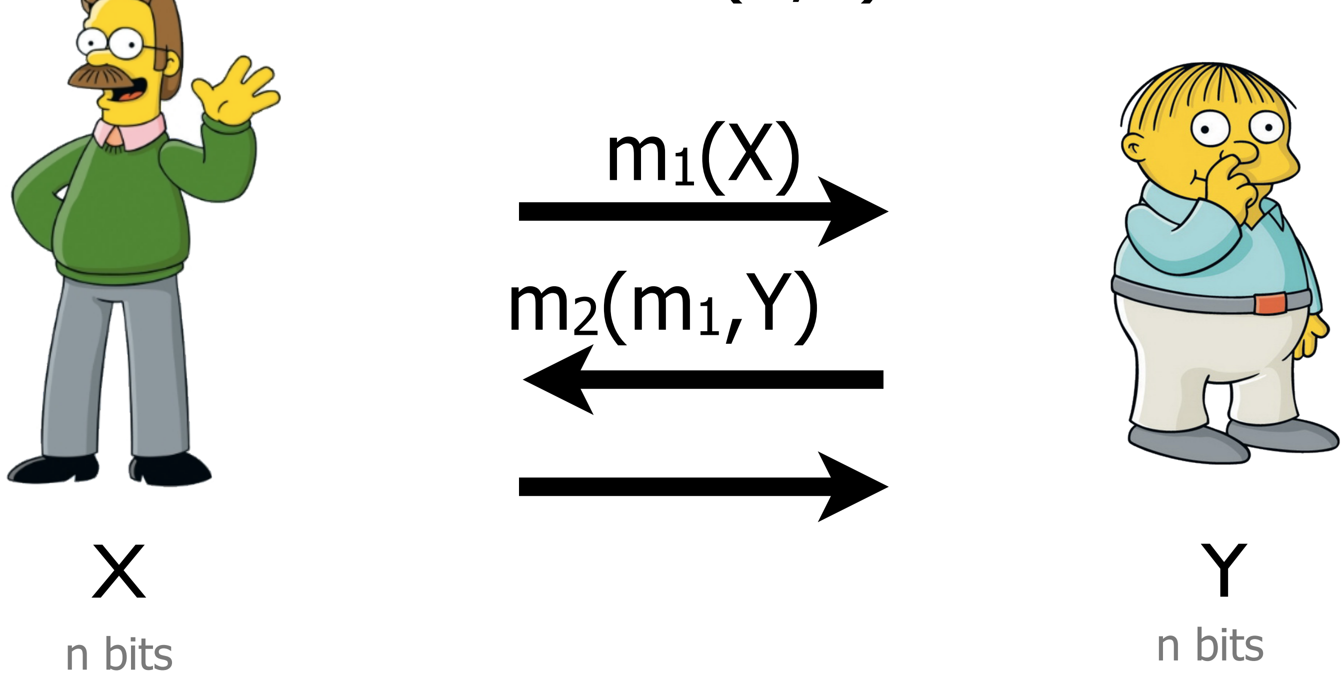
n bits

Y

n bits

2 party communication

What is f(X,Y) ?



2 party communication

What is $f(X,Y)$?

X

n bits

$m_1(X)$ →

← $m_2(m_1,Y)$

→

← $f(X,Y)$

Y

n bits

What we know about communication

Is X=Y ?

X

n bits

Y

n bits

What we know about communication

Is X=Y ?

Requires n bits communication

X

n bits

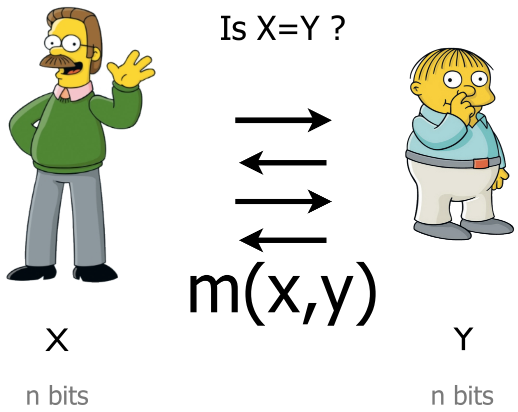
Y

n bits

Pigeonhole principle

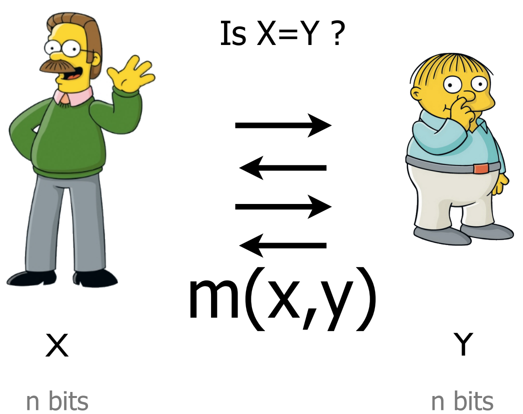
n+1 pigeons cannot fit in n holes

What we know about communication



Thm: Checking if X=Y requires n bits of communication.

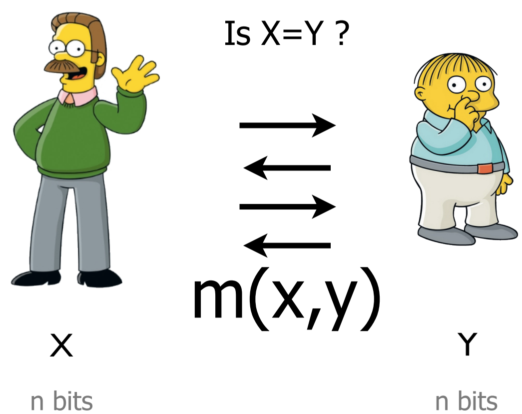
What we know about communication



Thm: Checking if X=Y requires n bits of communication.

Pf: Suppose there is protocol with < n bits of communication.

What we know about communication

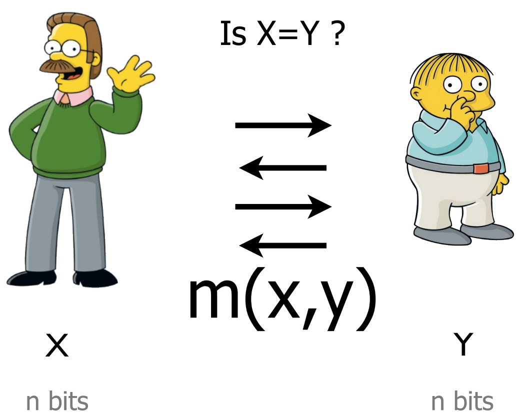


Thm: Checking if X=Y requires n bits of communication.

Pf: Suppose there is protocol with < n bits of communication.

There are 2^n inputs (x,y) with x=y pigeons

What we know about communication



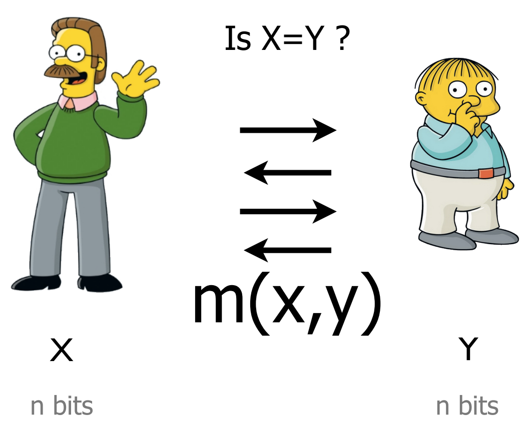
Thm: Checking if X=Y requires n bits of communication.

Pf: Suppose there is protocol with < n bits of communication.

There are 2^n inputs (x,y) with x=y — pigeons

There are < 2^n possible transcripts m — holes

What we know about communication



Thm: Checking if X=Y requires n bits of communication.

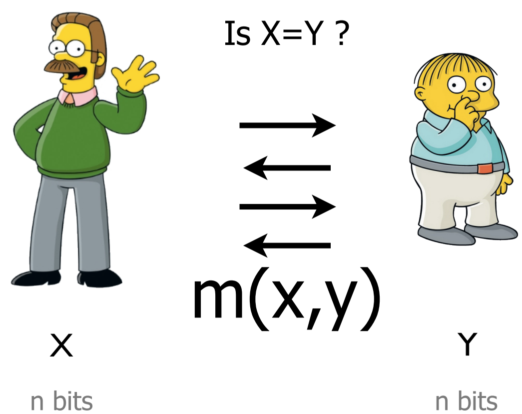
Pf: Suppose there is protocol with < n bits of communication.

There are 2^n inputs (x,y) with x=y — pigeons

There are < 2^n possible transcripts m — holes

There must be x=y, x'=y', with x≠x', m(x,y) = m(x',y')

What we know about communication



Thm: Checking if X=Y requires n bits of communication.

Pf: Suppose there is protocol with < n bits of communication.

There are 2^n inputs (x,y) with x=y — pigeons

There are < 2^n possible transcripts m — holes

There must be x=y, x'=y', with x≠x', m(x,y) = m(x',y')

But then m(x,y') = m(x,y)! The protocol has a bug.

What we know about communication

Is $|X \cap Y| = 0$?

Requires n bits communication

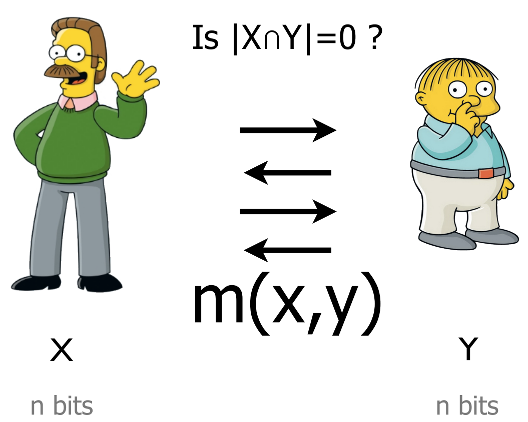
$X \subseteq \{1,2,\ldots,n\}$

n bits

$Y \subseteq \{1,2,\ldots,n\}$

n bits

What we know about communication



Thm: Checking if $|X \cap Y|=0$ requires n bits of communication.

Pf: Suppose there is protocol with < n bits of communication.

There are 2^n inputs (x,y) with y=complement(x) pigeons

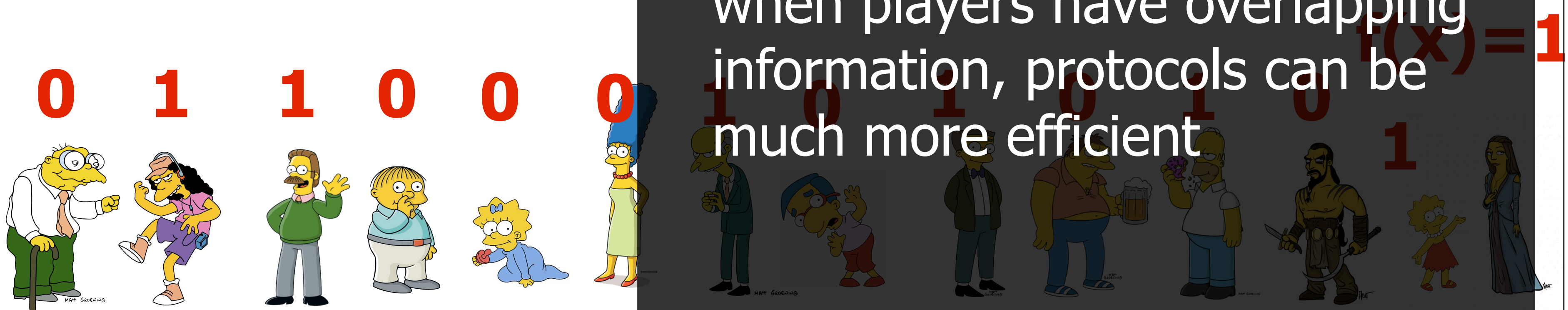
There are < 2^n possible transcripts m holes

There must be y=complement(x), y′=complement(x′), with $x \neq x'$, m(x,y) = m(x′,y′).

But then m(x,y′) = m(x,y), but $|x \cap y'|>0$.

Algorithms vs Multiparty Communication

[Valiant] If $f(x):\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time $O(\log n)$, with total work $O(n)$,



x=0010011101010010001011110001 01001

protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

each player represents a "critical" line of program execution

Remarks

when players have overlapping information, protocols can be much more efficient

Overlapping information

X

n bits

Is X=Y=Z ?

Requires n bits communication

Y

n bits

Z

n bits

Overlapping information

Is X=Y=Z ?

X,Y

n bits

Y,Z

n bits

Z,X

n bits

Overlapping information

Is X=Y=Z ?

2 bits of
communication suffice!

X=Y? Y=Z? determine answer

X,Y

n bits

Y,Z

n bits

Z,X

n bits

Overlapping information

Is $|X \cap Y \cap Z|$ even?

[BNS,1990] n bits of communication required

$Y,Z \subseteq \{1,2,\ldots,n\}$

n bits

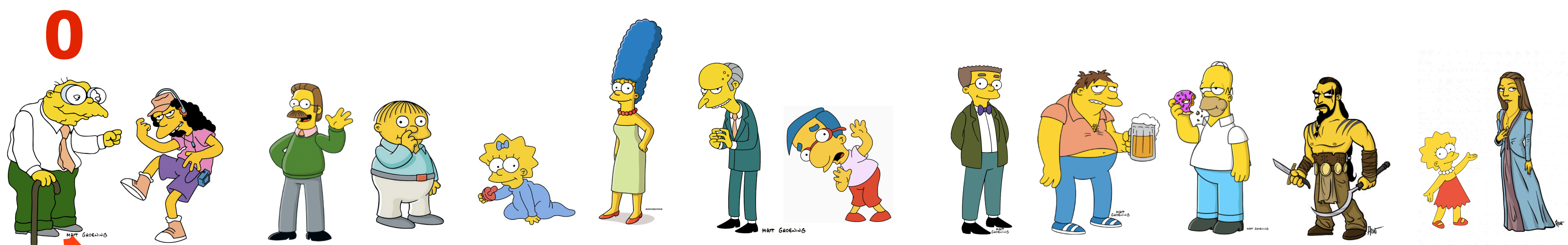
$X,Y \subseteq \{1,2,\ldots,n\}$

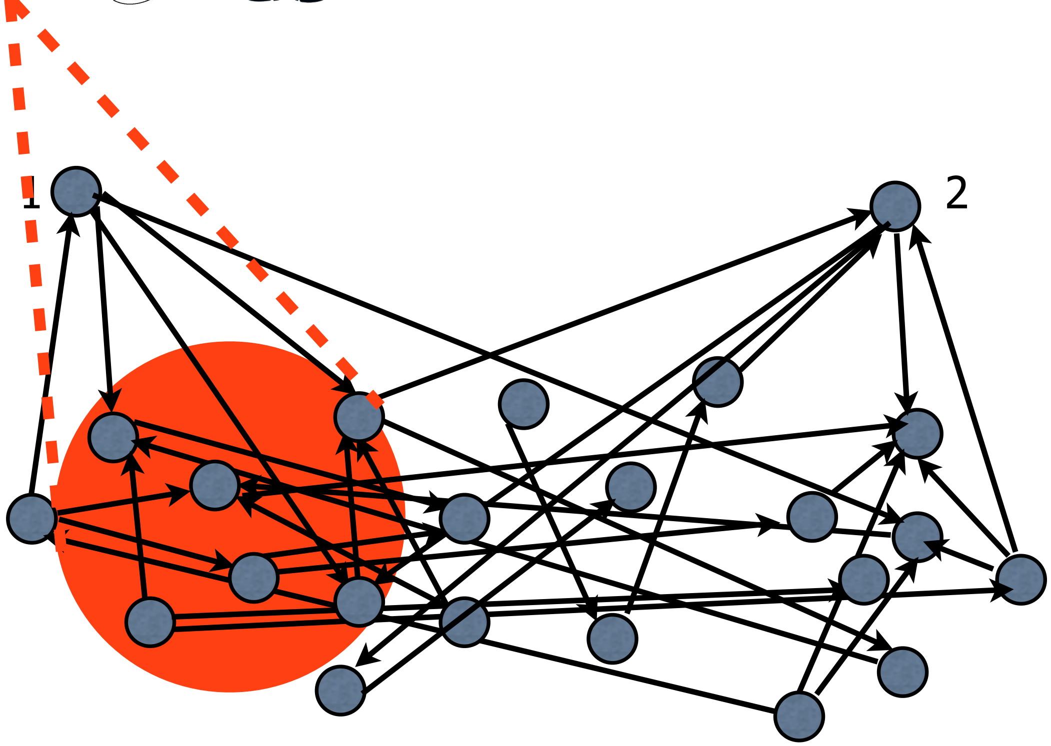
n bits

$Z,X \subseteq \{1,2,\ldots,n\}$

n bits

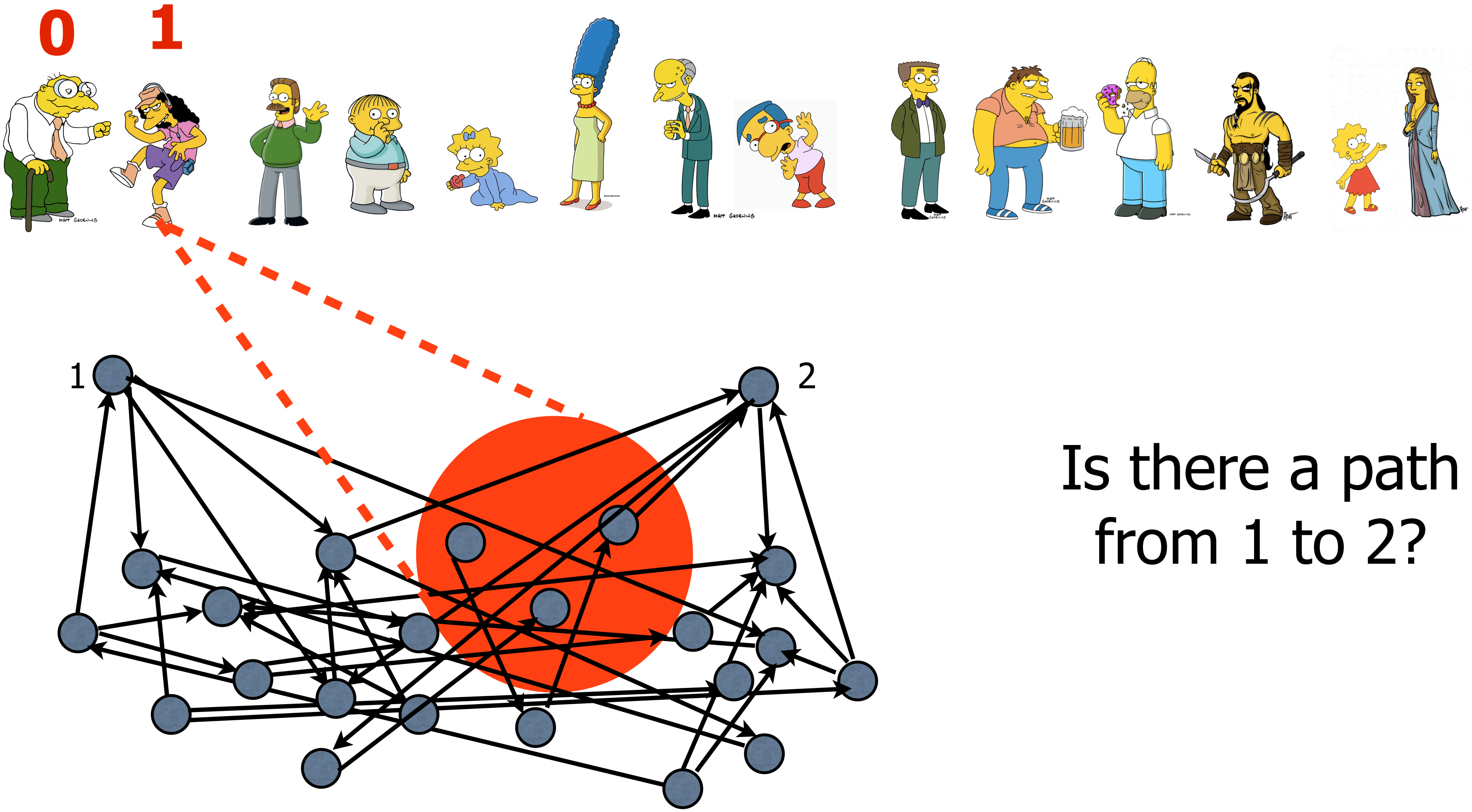
Algorithms vs Multiparty Communication





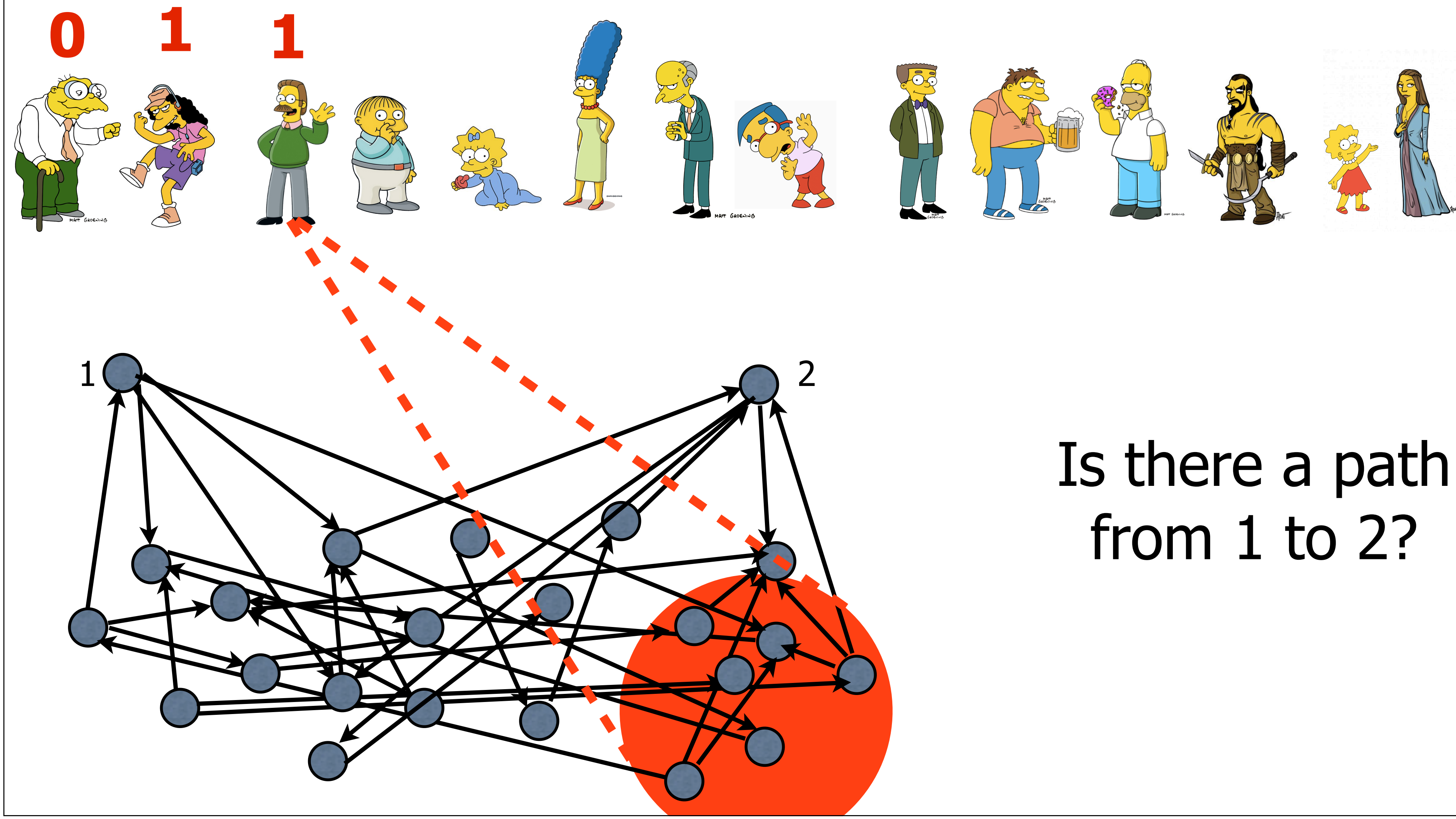
Is there a path from 1 to 2?

Algorithms vs Multiparty Communication



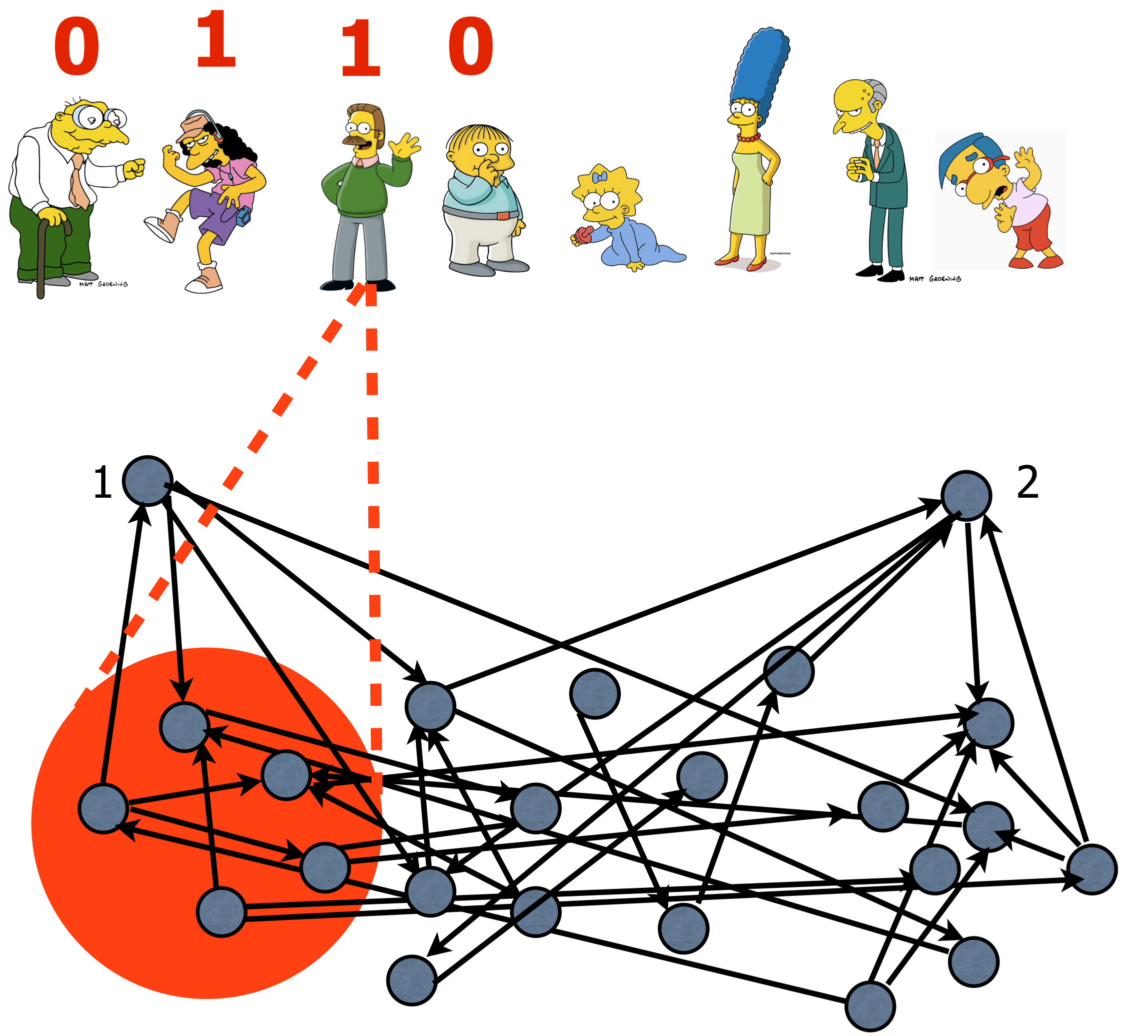
Is there a path from 1 to 2?

Algorithms vs Multiparty Communication



Is there a path from 1 to 2?

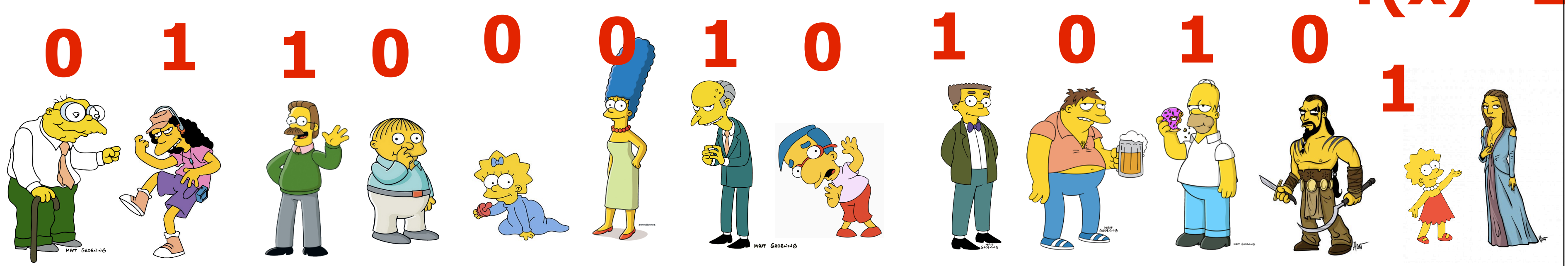
Algorithms vs Multiparty Communication

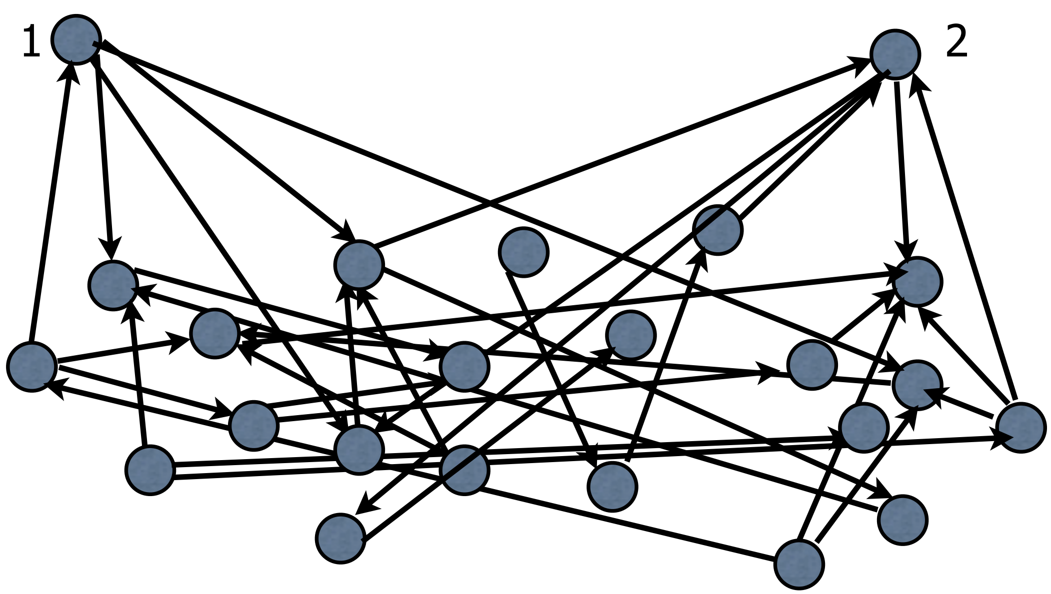


Is there a path from 1 to 2?

Algorithms vs Multiparty Communication

f(x)=1





Is there a path from 1 to 2?

Conjecture: there is no way to do this with a short protocol

Conclusions

- Lots of cool combinatorial problems that we don't know how to solve
- Lots of room for interesting math

Thanks

Overlapping information

Is $|X \cap Y \cap Z| = 0$?

Open for a long time

$X, Y \subseteq \{1, 2, \ldots, n\}$

n bits

$Y, Z \subseteq \{1, 2, \ldots, n\}$

n bits

$Z, X \subseteq \{1, 2, \ldots, n\}$

n bits

Overlapping information

Is $|X \cap Y \cap Z| = 0$?

Open for a long time

$Y,Z \subseteq \{1,2,\ldots,n\}$

n bits

$X,Y \subseteq \{1,2,\ldots,n\}$

n bits

$Z,X \subseteq \{1,2,\ldots,n\}$

n bits

Other applications

- separations between circuit classes
- proof complexity lower bounds
- oracle separations between complexity classes

Prior work

k players, k sets, each knows k-1 sets

$|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Reference	communication reqd.
[T03], [BPSW06]	$\log(n)/k$
[LS09], [CA08]	$n^{1/(k+1)}/2^{2^{O(k)}}$
[BH09]	$2^{\Omega(\sqrt{\log(n)/k})}/2^k$
[S12]	$n^{1/4}/2^{k/2}$
[S13]	$\sqrt{n}/k2^k$
[This work]	$n/4^k$

[G92]: $k^2n/2^k$ bits suffice

Prior work

k players, k sets, each knows k-1 sets

$|X_1 \cap X_2 \cap \ldots \cap X_k| = 0?$

Reference	communication reqd.
[T03], [BPSW06]	$\log(n)/k$
[LS09], [CA08]	$n^{1/(k+1)}/2^{2^{O(k)}}$
[BH09]	$2^{\Omega(\sqrt{\log(n)/k})}/2^k$
[S12]	$n^{1/4}/2^{k/2}$
[S13]	$\sqrt{n}/k2^k$
[This work]	$n/4^k$

Our bound

- the simplest proof: 3 pages
- we also simplify Sherstov's randomized lower bound, skipping several steps

[G92]: $k^2 n/2^k$ bits suffice

Proof Outline

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Partition universe into $m = n/16(4^k)$ parts, each of size $16(4^k)$

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Partition universe into $m = n/16(4^k)$ parts, each of size $16(4^k)$

In each part, $X_1, \ldots, X_{k-1}$: random sets intersecting in one point, X_k : independent uniformly random set. (Note: these sets almost always intersect!)

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Partition universe into $m=n/16(4^k)$ parts, each of size $16(4^k)$

In each part, $X_1, \ldots, X_{k-1}$: random sets intersecting in one point, X_k : independent uniformly random set. (Note: these sets almost always intersect!)

D_i: indicator variable for no intersection in i'th part

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Partition universe into $m = n/16(4^k)$ parts, each of size $16(4^k)$

In each part, $X_1, \ldots, X_{k-1}$: random sets intersecting in one point, X_k : independent uniformly random set. (Note: these sets almost always intersect!)

D_i: indicator variable for no intersection in i'th part

Thm [implicit in S12]: If π is computed in communication c

$$\left| \mathbb{E}\left[\pi(X_1, \ldots, X_k) \cdot (-1)^{\sum_{i=1}^{m} D_i} \right] \right| \leq 2^{c-2m}$$

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Partition universe into $m=n/16(4^k)$ parts, each of size $16(4^k)$

In each part, $X_1,\ldots,X_{k-1}$: random sets intersecting in one point, X_k : independent uniformly random set. (Note: these sets almost always intersect!)

D_i: indicator variable for no intersection in i'th part

Thm [implicit in S12]: If π is computed in communication c

$$\left|\mathbb{E}\left[\pi(X_1,\ldots,X_k)\cdot(-1)^{\sum_{i=1}^m D_i}\right]\right| \leq 2^{c-2m}$$

If π computes disjointness, when $\pi = 1, \sum_{i=1}^{m} D_i = m$.

Proof Outline

k players, k sets, each knows k-1 sets. $|X_1 \cap X_2 \cap \ldots \cap X_k| = 0$?

Partition universe into $m = n/16(4^k)$ parts, each of size $16(4^k)$

In each part, $X_1, \ldots, X_{k-1}$: random sets intersecting in one point, X_k : independent uniformly random set. (Note: these sets almost always intersect!)

D_i: indicator variable for no intersection in i'th part

Thm [implicit in S12]: If π is computed in communication c

$$\left| \mathbb{E}\left[\pi(X_1, \ldots, X_k) \cdot (-1)^{\sum_{i=1}^{m} D_i} \right] \right| \leq 2^{c-2m}$$

If π computes disjointness, when $\pi = 1, \sum_{i=1}^{m} D_i = m$.

Thus LHS= $2^{-m} \leq 2^{c-2m} \Rightarrow c \geq m$

Randomized lower bound

Randomized lower bound

Define $f(j)$ probability protocol outputs 1 when $\sum_{i=1}^{m} D_i = j$.

Randomized lower bound

Define $f(j)$ probability protocol outputs 1 when $\sum_{i=1}^{m} D_i = j$.

Thm: If $f(j)$ defined using a protocol, $\forall r > c$

$$\left|\mathbb{E}\left[f((D_1 + \ldots + D_r)m/r) \cdot (-1)^{D_1 + \ldots + D_r}\right]\right| \leq 2^{-12r}.$$

Randomized lower bound

Define $f(j)$ probability protocol outputs 1 when $\sum_{i=1}^{m} D_i = j$.

Thm: If $f(j)$ defined using a protocol, $\forall r > c$

$$\left|\mathbb{E}\left[f((D_1 + \ldots + D_r)m/r) \cdot (-1)^{D_1 + \ldots + D_r}\right]\right| \leq 2^{-12r}.$$

Thm: f can be approximated by a degree c polynomial.

Randomized lower bound

Define $f(j)$ probability protocol outputs 1 when $\sum_{i=1}^{m} D_i = j$.

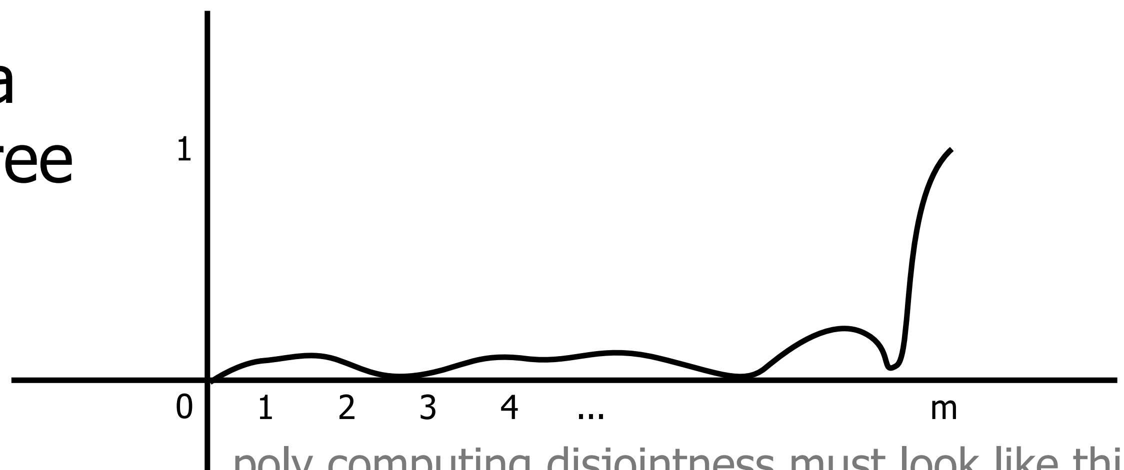
Thm: If $f(j)$ defined using a protocol, $\forall r > c$

$$\left|\mathbb{E}\left[f((D_1 + \ldots + D_r)m/r) \cdot (-1)^{D_1+\ldots+D_r}\right]\right| \leq 2^{-12r}.$$

Thm: f can be approximated by a degree c polynomial.

Thm [NS]: such a poly must have degree

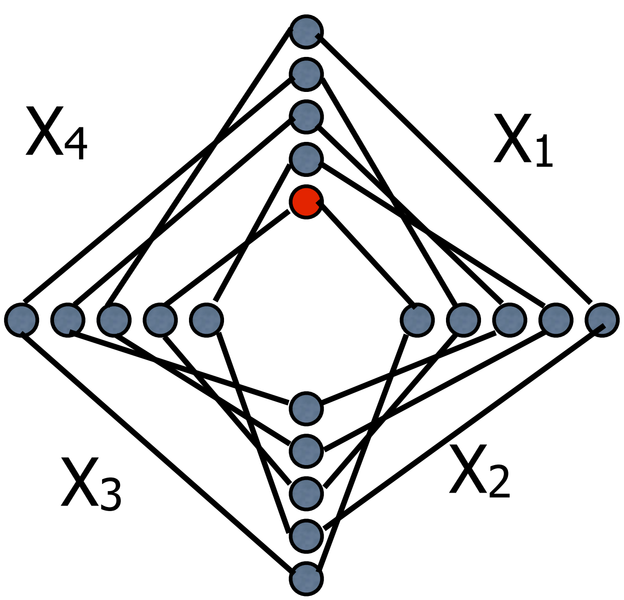
$\sqrt{m}$



poly computing disjointness must look like this

Open problems

- Prove lower bounds of n for these communication models. (Anything better than $n/2^k$).
- Candidate problem:



- Input: k matchings of size n
- Start at red, walk clockwise for n/100 steps, do we end at even vertex?
- Careful: When $k = 3$, there is a protocol that can compute 3rd step in o(n) communication! [PRS97]

Algorithms vs Multiparty Communication

[Valiant] If f(x):$\{0,1\}^n \rightarrow \{0,1\}$ can be computed in parallel time O(log n), with total work O(n),

x=0010011101010010001011111000101001

protocol with n/loglog(n) players
- player knows some $n^{0.1}$ bits of x
- player broadcasts 1 bit
- at end, someone knows f(x)

Candidate Hard Function

Given a graph on n vertices, where every vertex has out-degree 1, is 1 connected to 2?

each player represents a "critical" line of program execution