Superlinear Lower Bounds for Multipass Graph Processing

Krzysztof Onak
IBM T.J. Watson Research Center

Joint work with **Venkat Guruswami** (CMU)

Streaming Algorithms for Graphs

Model:

- Input: large stream of edges
- Goal: minimize the amount of space
  and processing time per edge
- Allowed: randomization and small error probability

Algorithm $\longleftarrow$ $\mathbf{(5,4)\ (1,2)\ (4,3)\ (2,5)\ (3,1)\ \ldots}$

Streaming Algorithms for Graphs

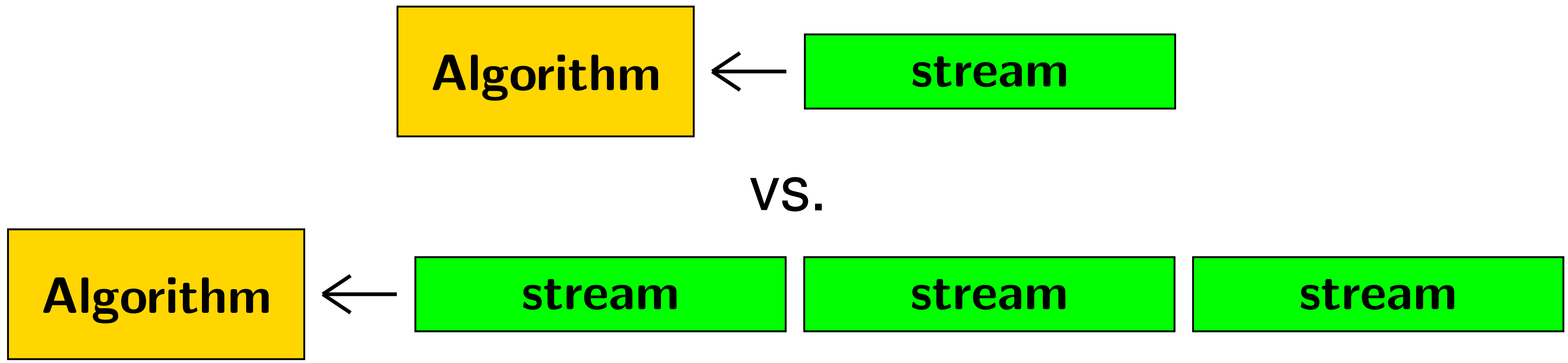
Model:

- Input: large stream of edges
- Goal: minimize the amount of space
  and processing time per edge
- Allowed: randomization and small error probability

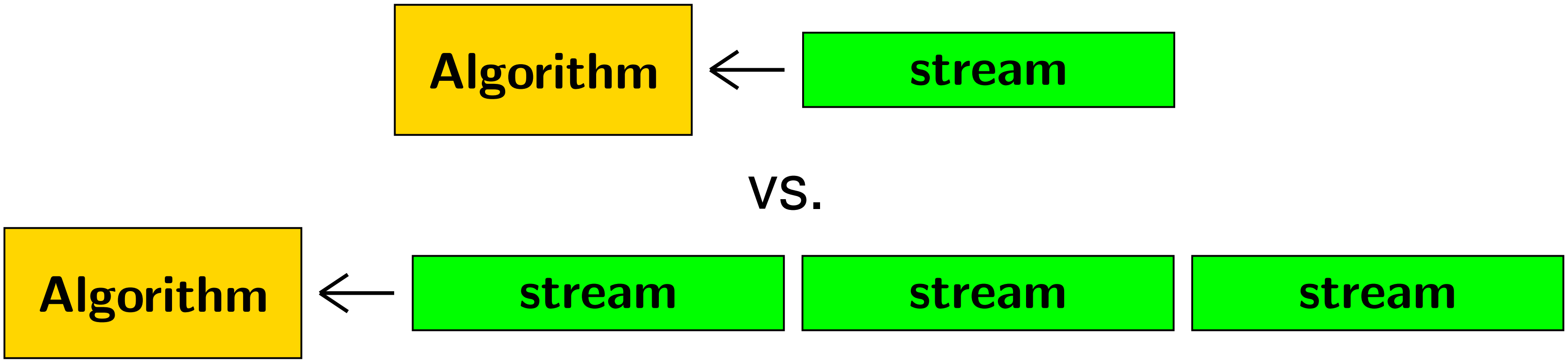
Algorithm $\longleftarrow$ $\mathbf{(5{,}4)\ (1{,}2)\ (4{,}3)\ (2{,}5)\ (3{,}1)\ \ldots}$

- Worst-case ordering of edges (as opposed to random)
  - The adversary knows the algorithm
    but not its random bits

One Pass vs. Multiple Passes

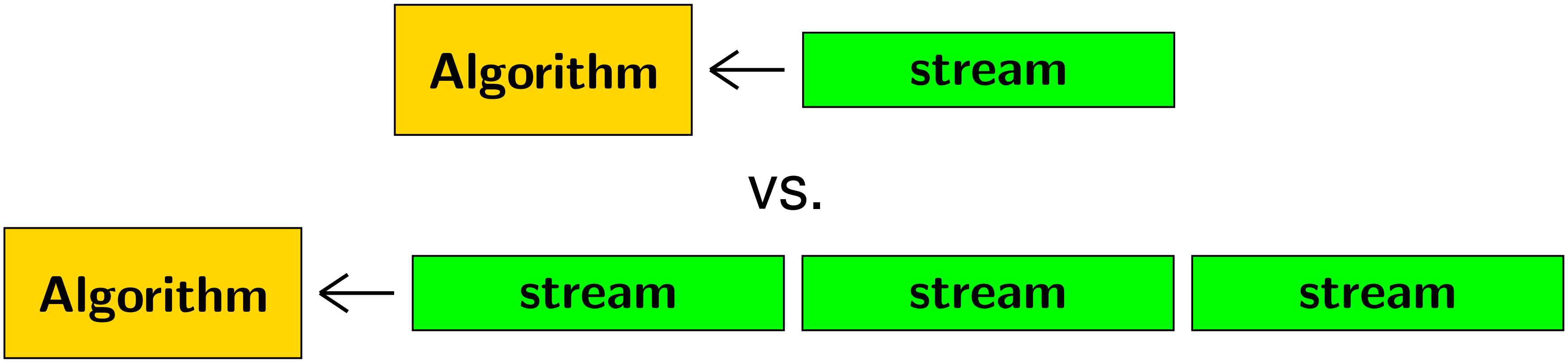


One Pass vs. Multiple Passes



Do multiple passes make sense?

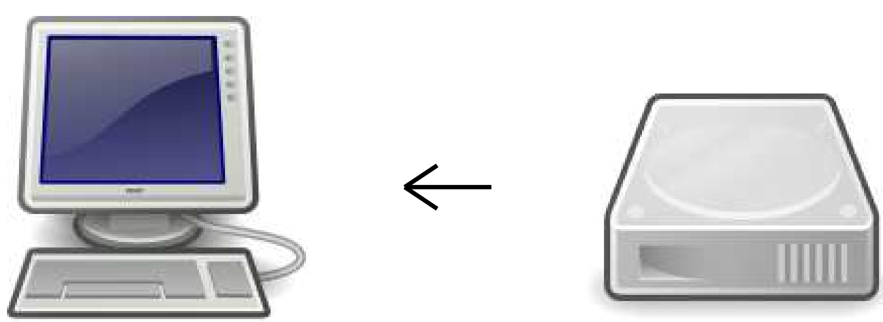
One Pass vs. Multiple Passes



Do multiple passes make sense?

YES:

- Data on a large external storage device
- Sequential access often maximizes throughput



Graph Streaming

"Sweet-spot" for graph streaming: Semi-streaming model
[Muthukrishnan'03]

- Allow $n \cdot \mathrm{poly}(\log n)$ space
- Enough space to store vertices, but not edges

Graph Streaming

"Sweet-spot" for graph streaming: Semi-streaming model [Muthukrishnan'03]

- Allow $n \cdot \mathrm{poly}(\log n)$ space
- Enough space to store vertices, but not edges

General challenge: Which graph-theoretic problems admit $n \cdot \mathrm{poly}(\log n)$ space streaming algorithms in one or a few passes?

Graph Streaming

“Sweet-spot” for graph streaming: Semi-streaming model [Muthukrishnan’03]

- Allow $n \cdot \mathrm{poly}(\log n)$ space
- Enough space to store vertices, but not edges

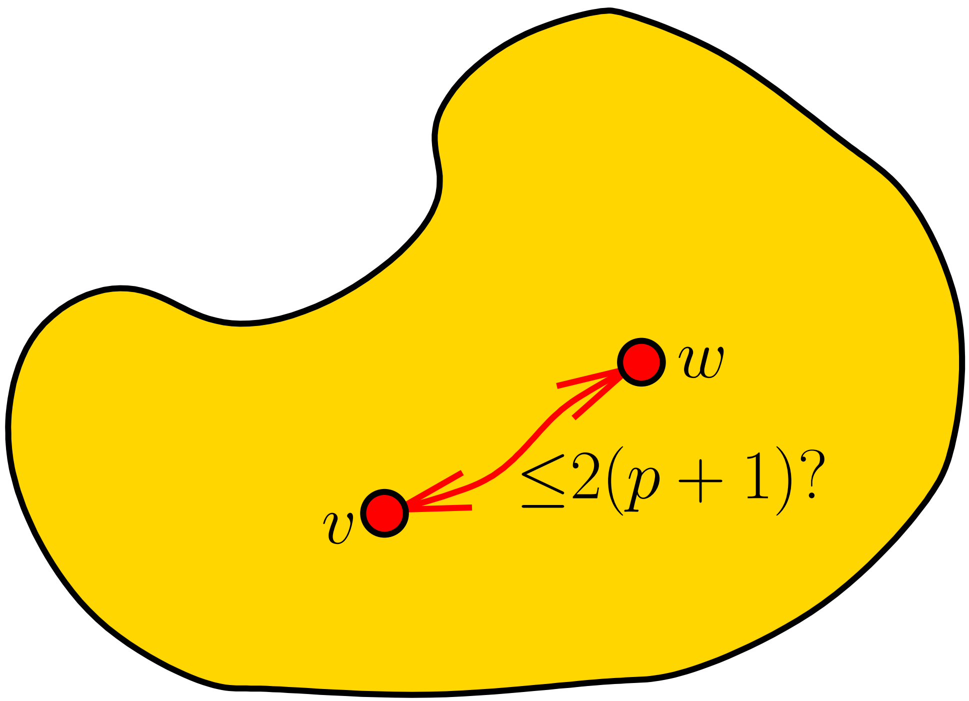
General challenge: Which graph-theoretic problems admit $n \cdot \mathrm{poly}(\log n)$ space streaming algorithms in one or a few passes?

This Work: Rule out such algorithms for some basic graph theory problems

Our Results

Undirected graphs:

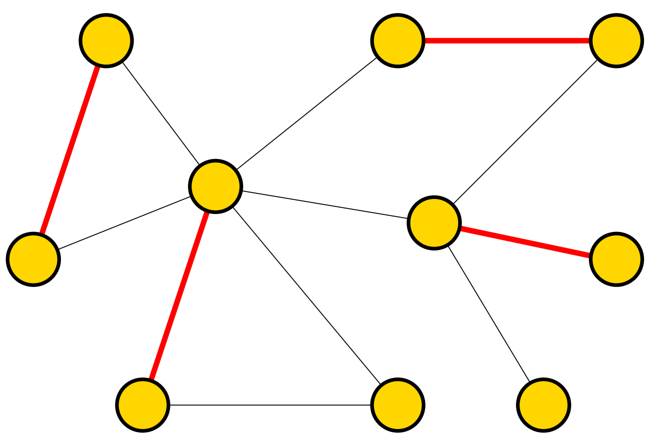
- **Problem 1:** Are v and w at distance at most $2(p+1)$?



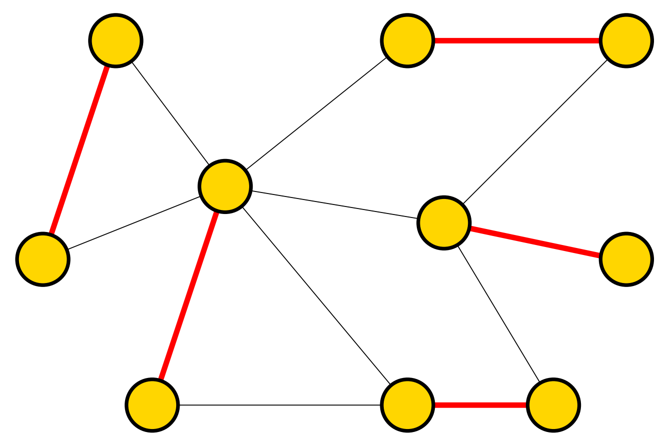
Our Results

Undirected graphs:

- **Problem 1:** Are v and w at distance at most $2(p+1)$?
- **Problem 2:** Is there a perfect matching?



vs.



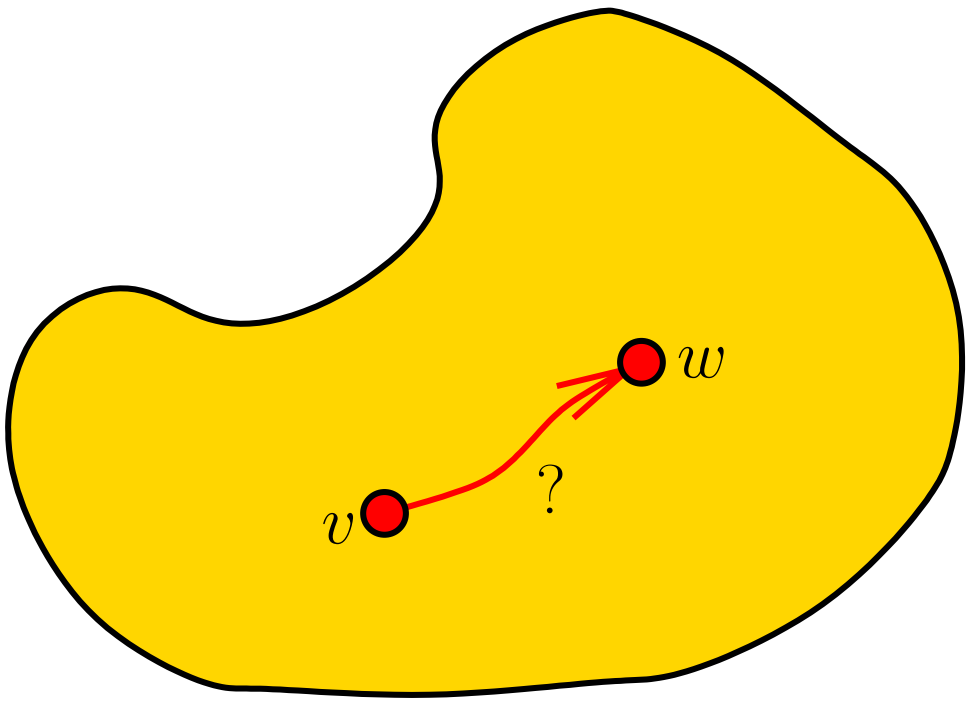
Our Results

Undirected graphs:

- **Problem 1:** Are v and w at distance at most $2(p+1)$?
- **Problem 2:** Is there a perfect matching?

Directed graphs:

- **Problem 3:** Is there a directed path from v to w?



Our Results

Undirected graphs:

- **Problem 1:** Are v and w at distance at most $2(p+1)$?
- **Problem 2:** Is there a perfect matching?

Directed graphs:

- **Problem 3:** Is there a directed path from v to w?

Solving these graph problems in p passes requires

$$\Omega\left(\frac{n^{1+1/(2p+2)}}{p^{20}\log^{3/2} n}\right) = \frac{n^{1+\Omega(1/p)}}{p^{O(1)}}$$

bits of space ($n =$ #vertices)

Comparison to Previous Results

- Known to require $\Omega(n^2)$ bits in one pass
  [Feigenbaum, Kannan, McGregor, Suri, Zhang 2004]

Comparison to Previous Results

- Known to require $\Omega(n^2)$ bits in one pass [Feigenbaum, Kannan, McGregor, Suri, Zhang 2004]

- Easy to prove $\Omega(n/p)$ for p passes via set disjointness

Comparison to Previous Results

- Known to require $\Omega(n^2)$ bits in one pass
[Feigenbaum, Kannan, McGregor, Suri, Zhang 2004]

- Easy to prove $\Omega(n/p)$ for p passes via set disjointness

- We want $n^{1+\Omega(1)}$ lower bounds

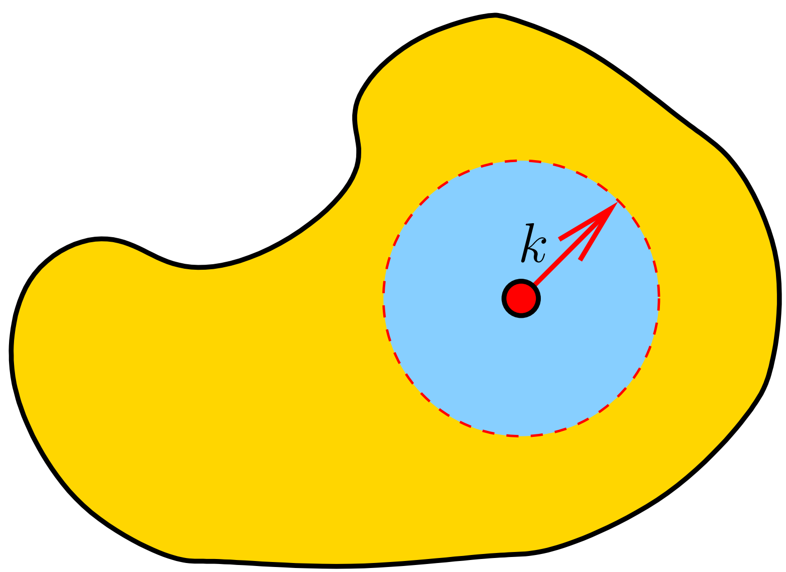
Comparison to Previous Results

- Known to require $\Omega(n^2)$ bits in one pass [Feigenbaum, Kannan, McGregor, Suri, Zhang 2004]
- Easy to prove $\Omega(n/p)$ for p passes via set disjointness
- We want $n^{1+\Omega(1)}$ lower bounds
- Main challenge: embed hard problems into
  the "space of edges"
  not just vertices

Related Results: Shortest Path(s)

Feigenbaum, Kannan, McGregor, Suri, Zhang (2005):

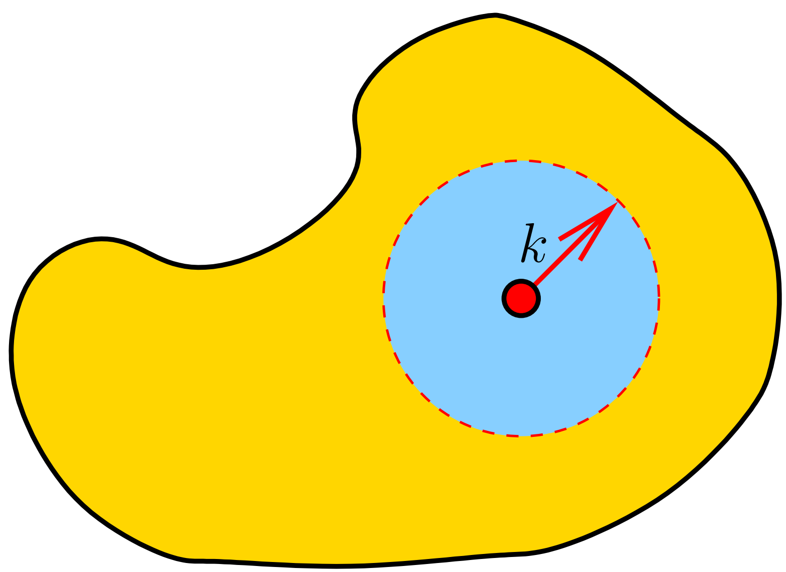
Computing the first k layers of BFS tree in $<k/2$ passes requires $\Omega(n^{1+1/k}/k^{O(1)}(\log n)^{1/k})$ space



Related Results: Shortest Path(s)

Feigenbaum, Kannan, McGregor, Suri, Zhang (2005):
Computing the first k layers of BFS tree in $<k/2$ passes requires $\Omega(n^{1+1/k}/k^{O(1)}(\log n)^{1/k})$ space

- Can be improved to $< k$ passes using [Guha, McGregor 2007]

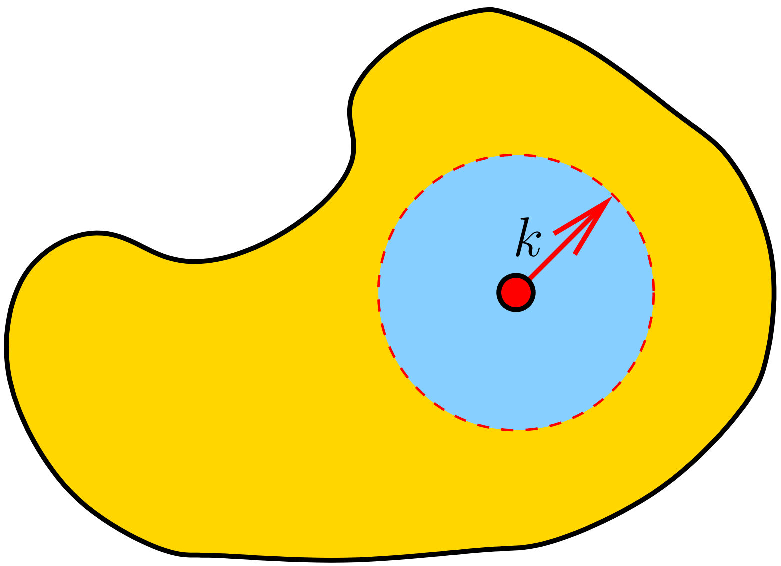


Related Results: Shortest Path(s)

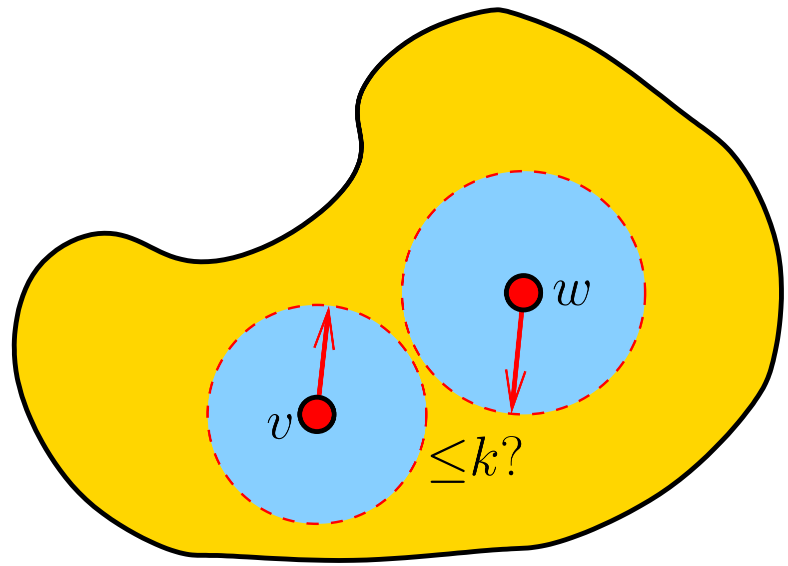
Feigenbaum, Kannan, McGregor, Suri, Zhang (2005):
Computing the first k layers of BFS tree in $<k/2$ passes requires $\Omega(n^{1+1/k}/k^{O(1)}(\log n)^{1/k})$ space

- Can be improved to $< k$ passes using [Guha, McGregor 2007]

Our problem: Fewer passes suffice



[FKMSZ'05]



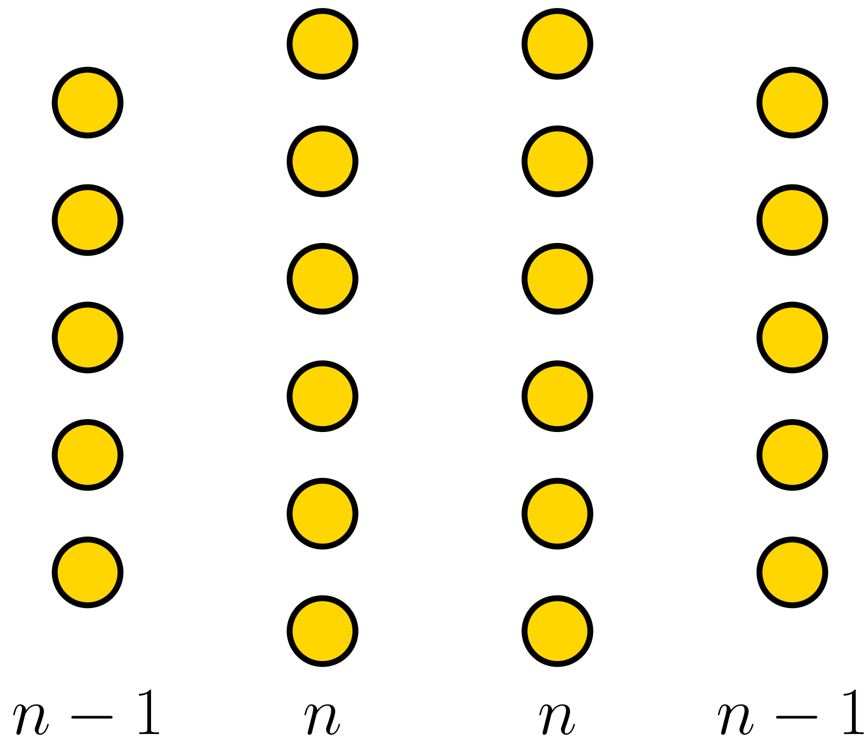
Here

Warmup:

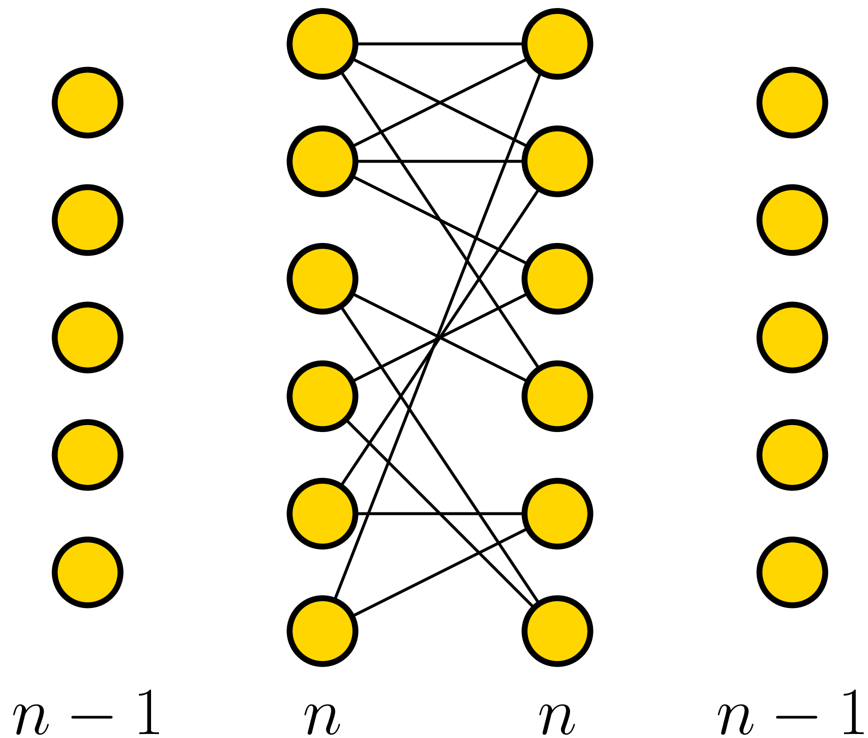
One-Pass Lower Bound

[Feigenbaum et al. 2004]

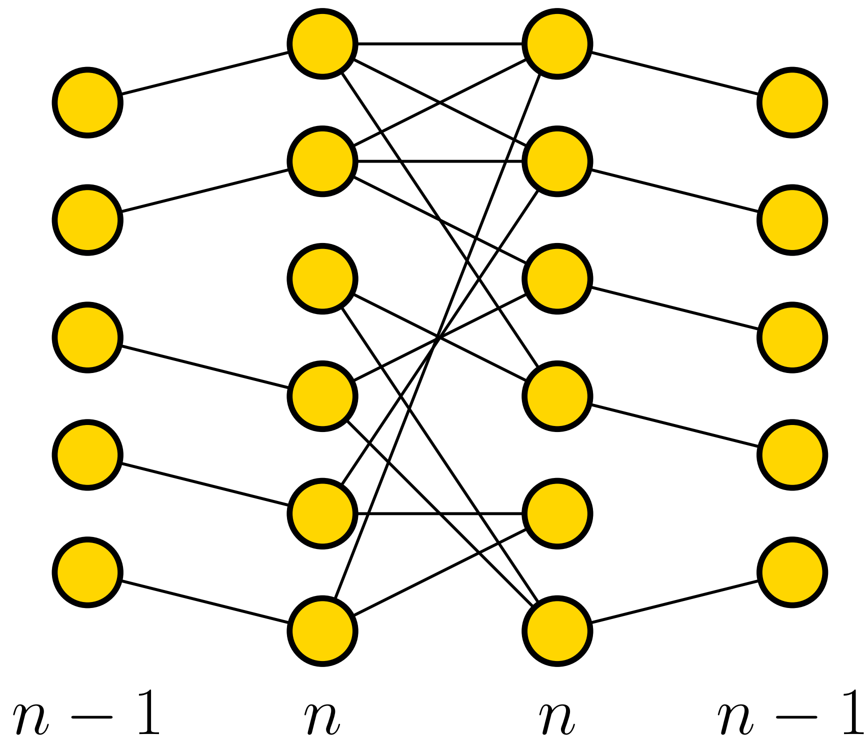
Construction for Perfect Matching



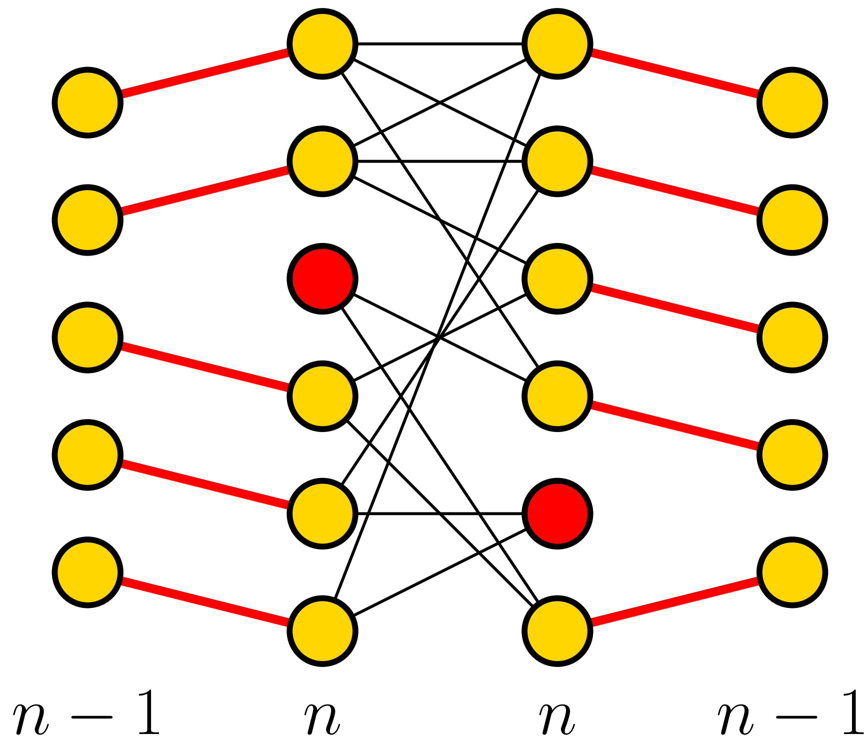
Construction for Perfect Matching



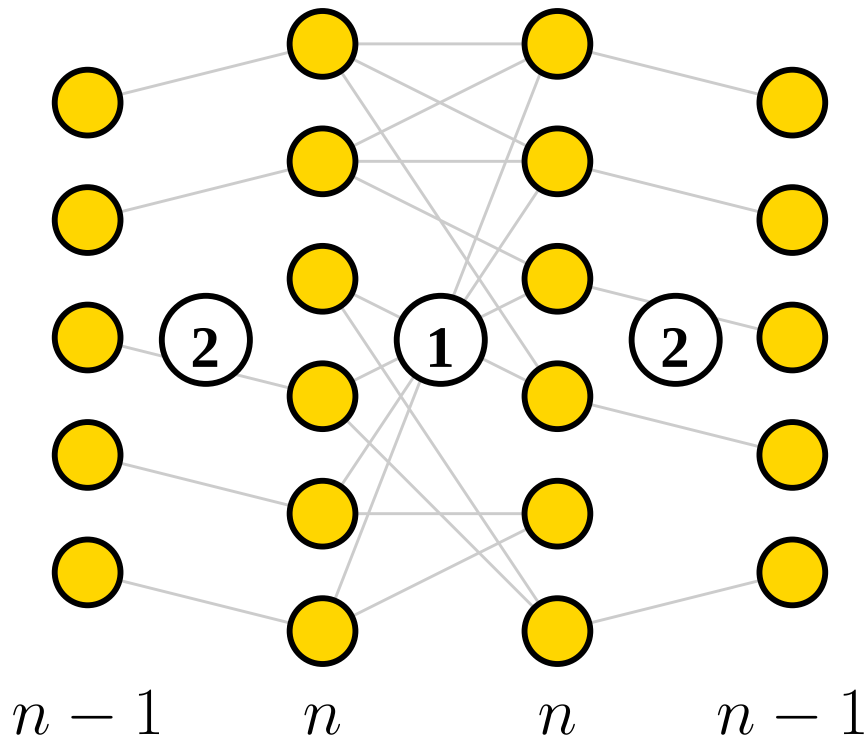
Construction for Perfect Matching



Construction for Perfect Matching

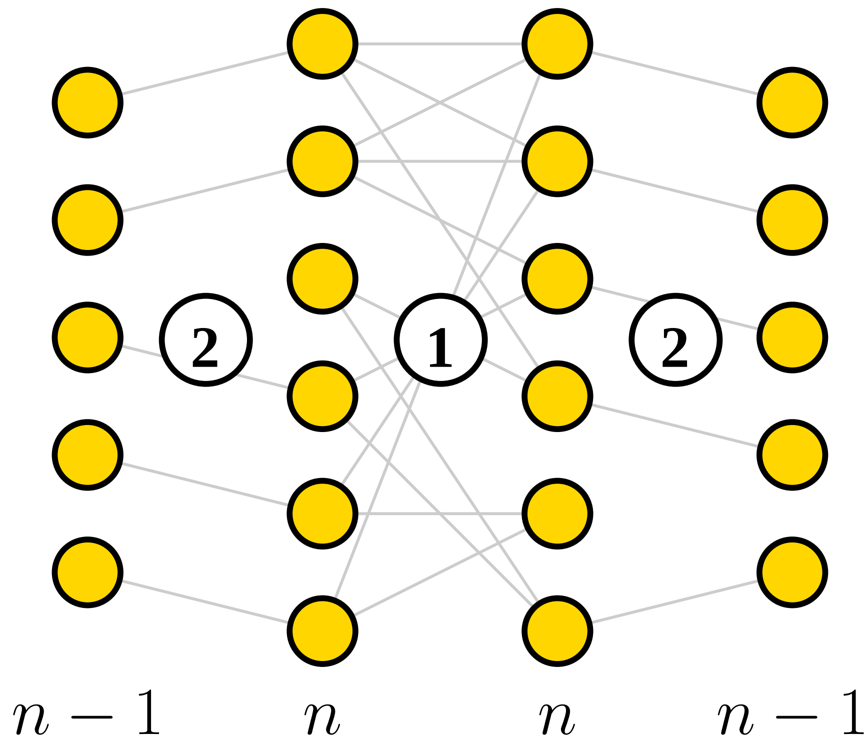


Construction for Perfect Matching



Stream = (1) (2)

Construction for Perfect Matching



Stream = ① ②

Lower bound of $\Omega(n^2)$ via indexing

Alice		Bob
$A[1 \ldots n^2]$	$\Rightarrow$	x

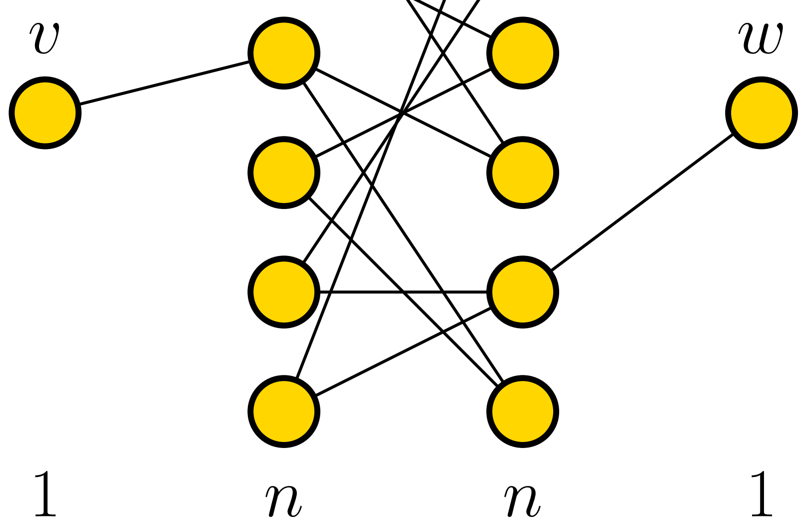
Bob's task: output $A[x]$

Construction for Shortest Path

Approximation better then $5/3$ requires $\Omega(n^2)$ space

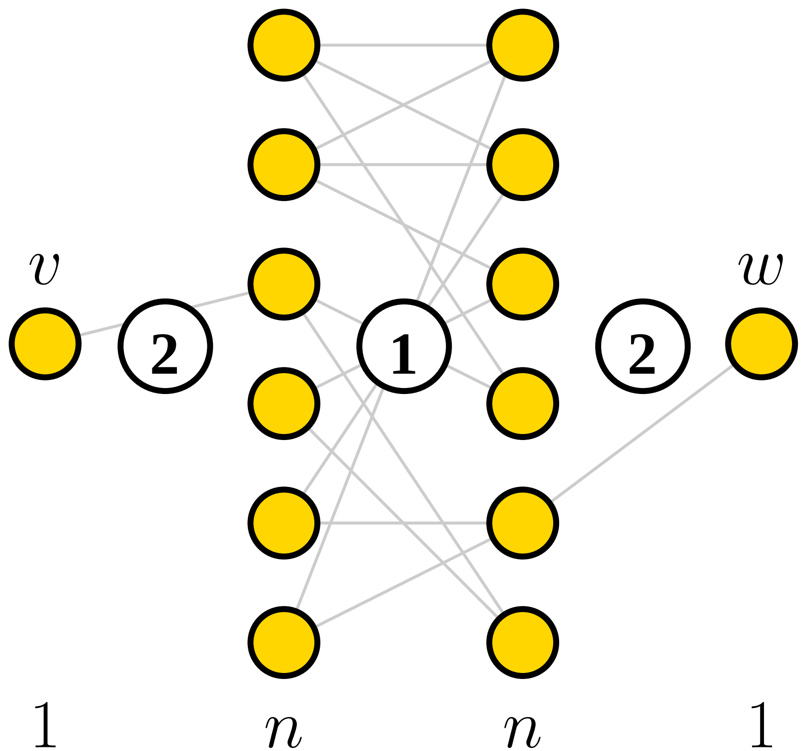
Construction for Shortest Path

Approximation better then $5/3$ requires $\Omega(n^2)$ space



Construction for Shortest Path

Approximation better then $5/3$ requires $\Omega(n^2)$ space



Stream = ① ②

Stream Ordering

- How do we order edges in the stream?

Stream Ordering

- How do we order edges in the stream?
- Graphs $=$ vertices $+$ relations between them

Stream Ordering

- How do we order edges in the stream?
- Graphs $=$ vertices $+$ relations between them
- Solving most graph problems requires exploration

Stream Ordering

- How do we order edges in the stream?
- Graphs $=$ vertices $+$ relations between them
- Solving most graph problems requires exploration
- To prove lower bounds, create obstacles for exploration

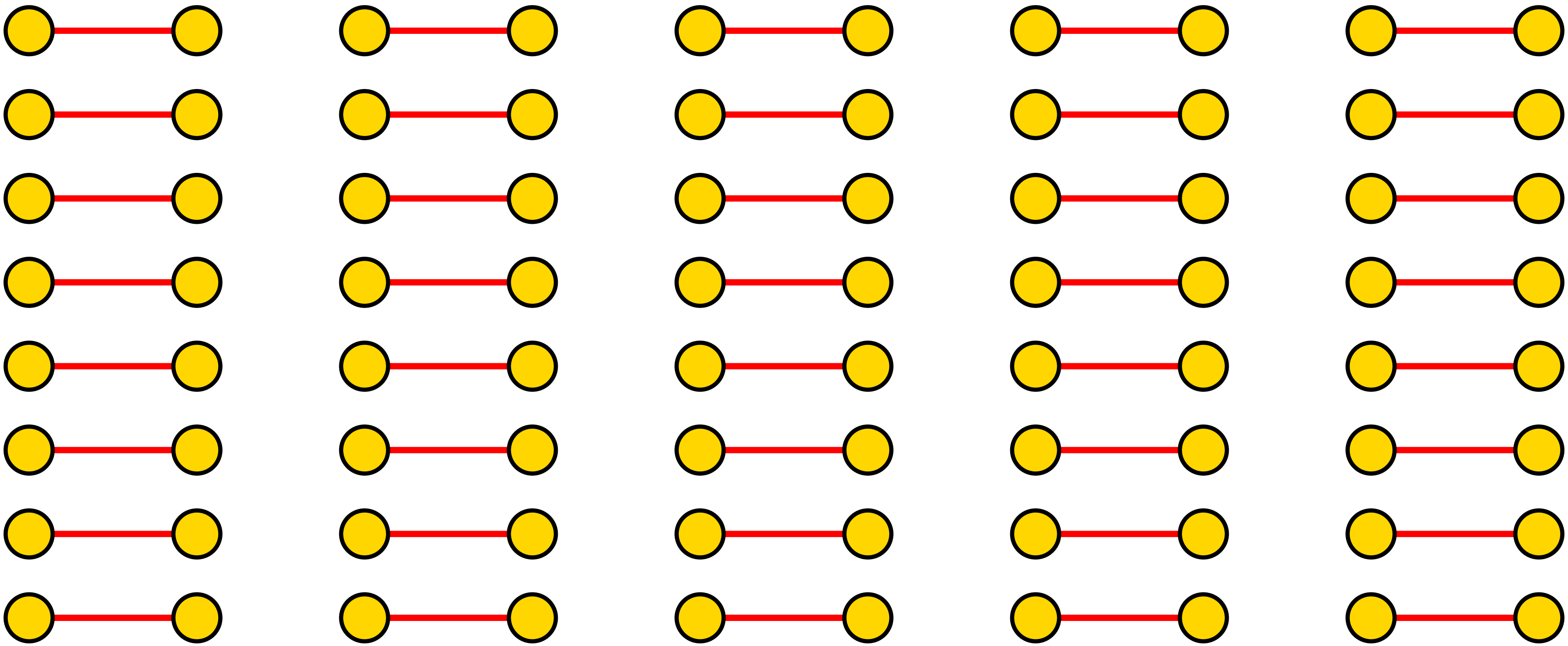
Stream Ordering

- How do we order edges in the stream?
- Graphs $=$ vertices $+$ relations between them
- Solving most graph problems requires exploration
- To prove lower bounds, create obstacles for exploration
- One possibility: present edges in order opposite to what is suitable for exploration

Hard Instance for Multiple Passes

Construction for Perfect Matching

Is there a perfect matching?

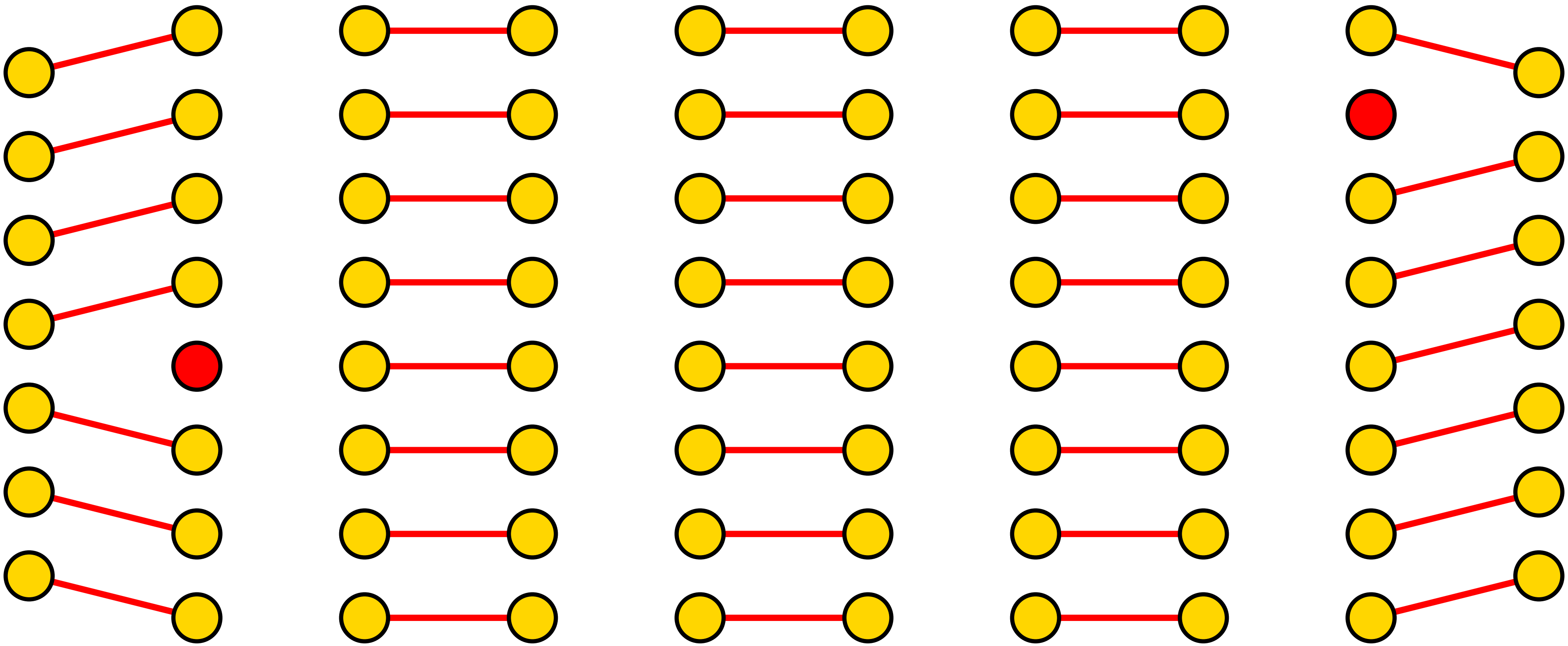


$\Theta(1)$ columns

Each column $\Theta(n)$ rows

Construction for Perfect Matching

Is there a perfect matching?

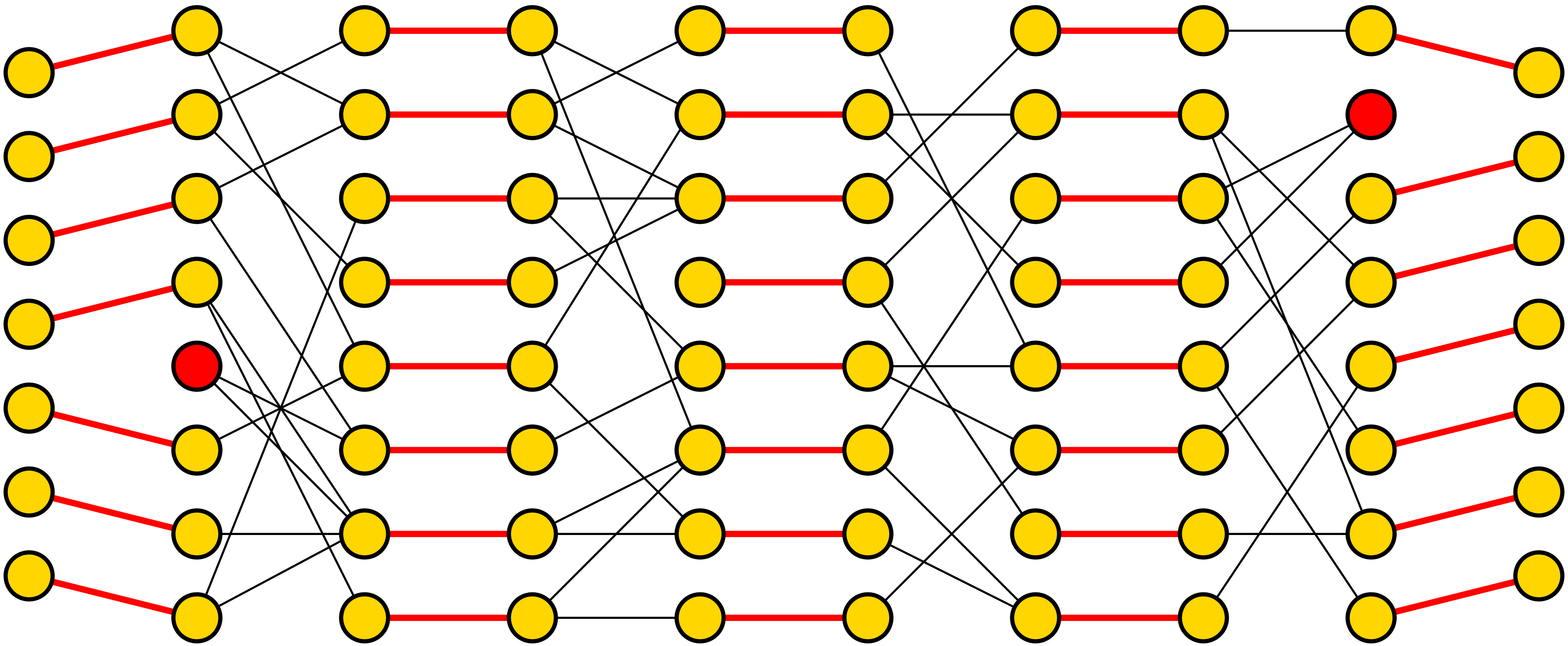


$\Theta(1)$ columns

Each column $\Theta(n)$ rows

Construction for Perfect Matching

Is there a perfect matching?

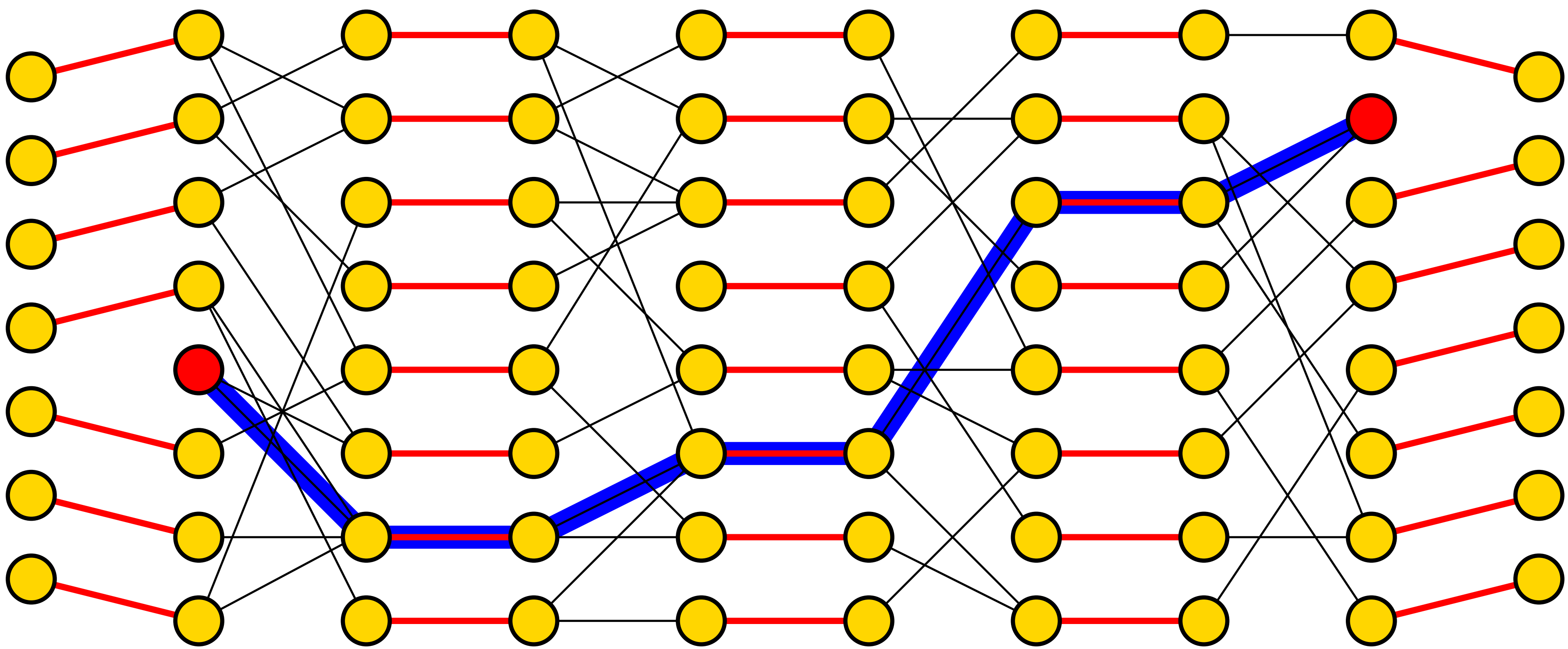


$\Theta(1)$ columns

Each column $\Theta(n)$ rows

Construction for Perfect Matching

Is there a perfect matching?

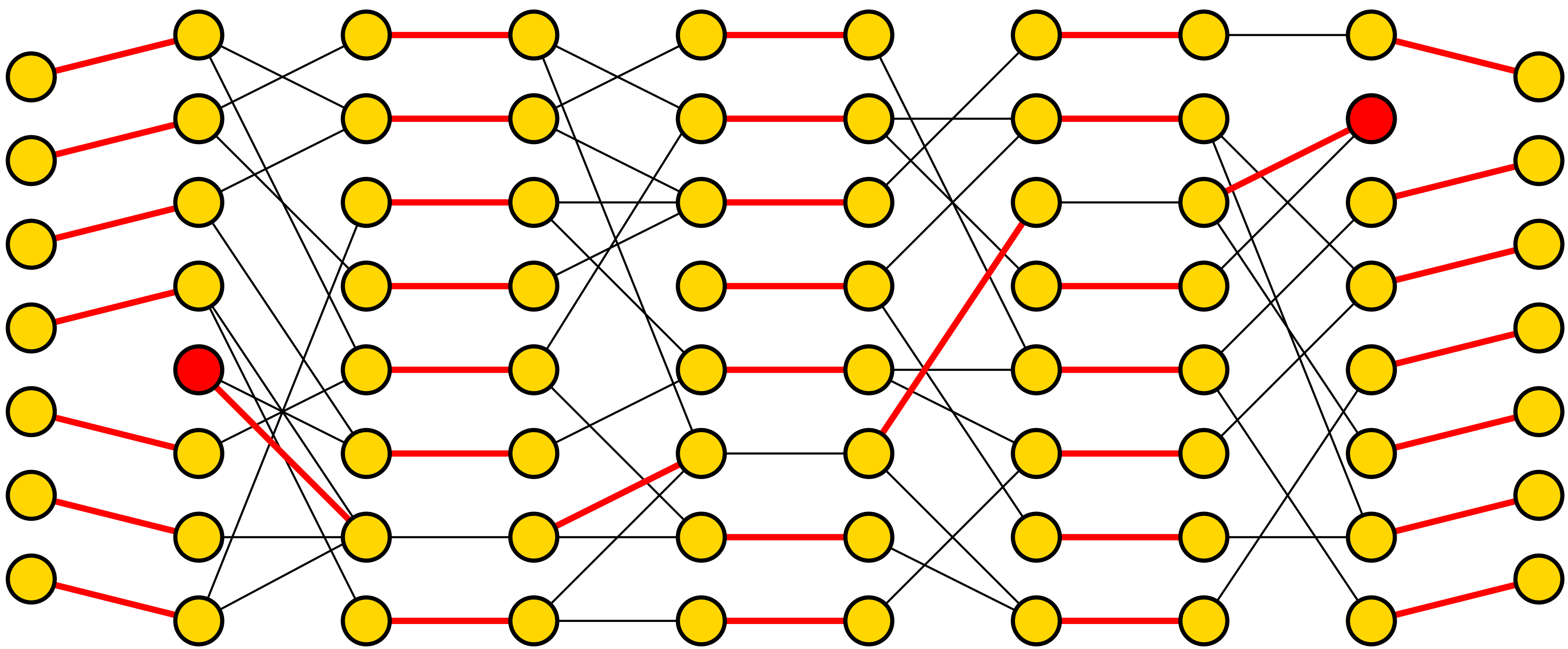


$\Theta(1)$ columns

Each column $\Theta(n)$ rows

Construction for Perfect Matching

Is there a perfect matching?

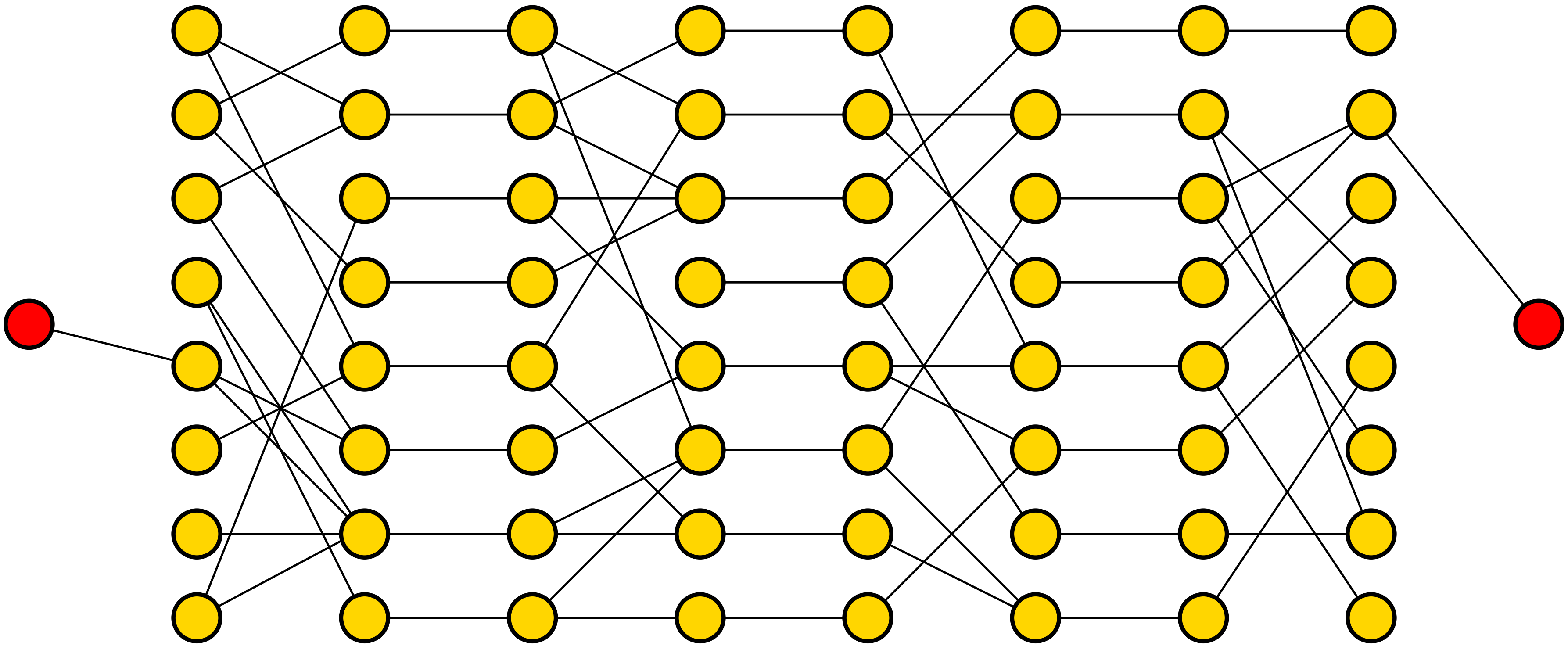


$\Theta(1)$ columns

Each column $\Theta(n)$ rows

Construction for Perfect Matching

Is there a path of length 9 between red nodes?

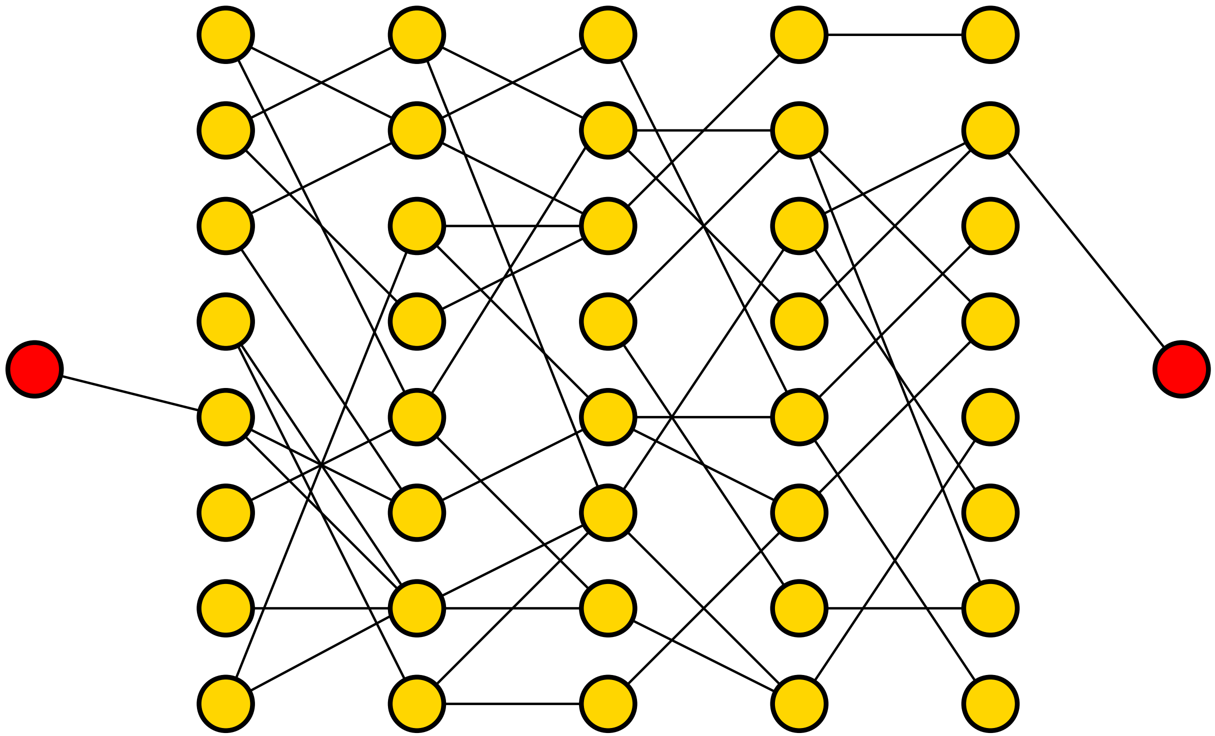


$\Theta(1)$ columns

Each column $\Theta(n)$ rows

Construction for Perfect Matching

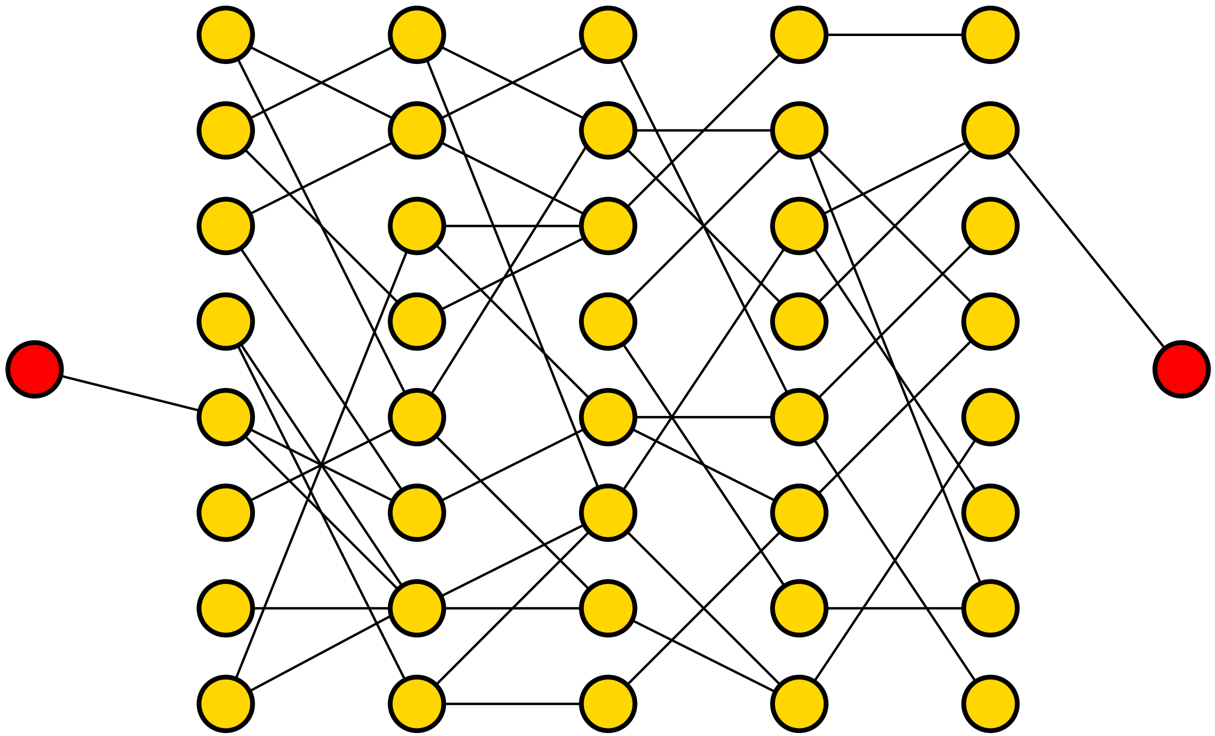
Is there a path of length 6 between red nodes?



$\Theta(1)$ columns Each column $\Theta(n)$ rows

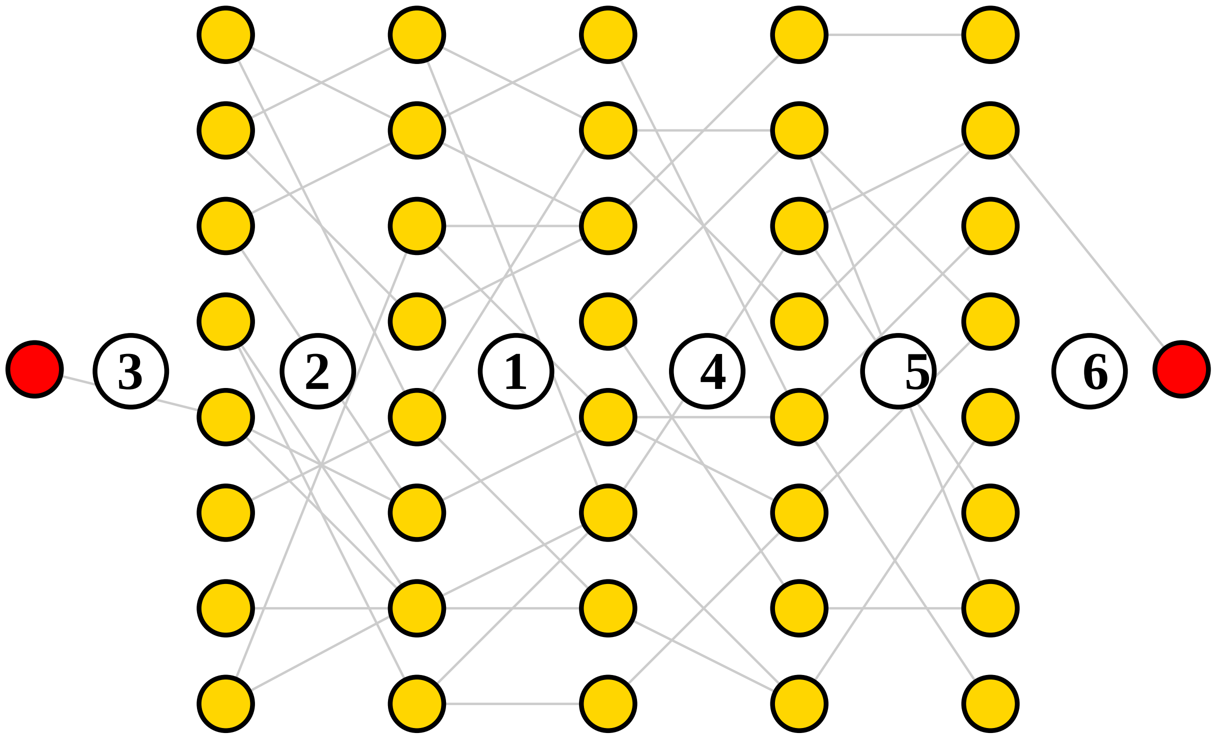
Our Stream Ordering

Is there a path of length 6 between red nodes?



Our Stream Ordering

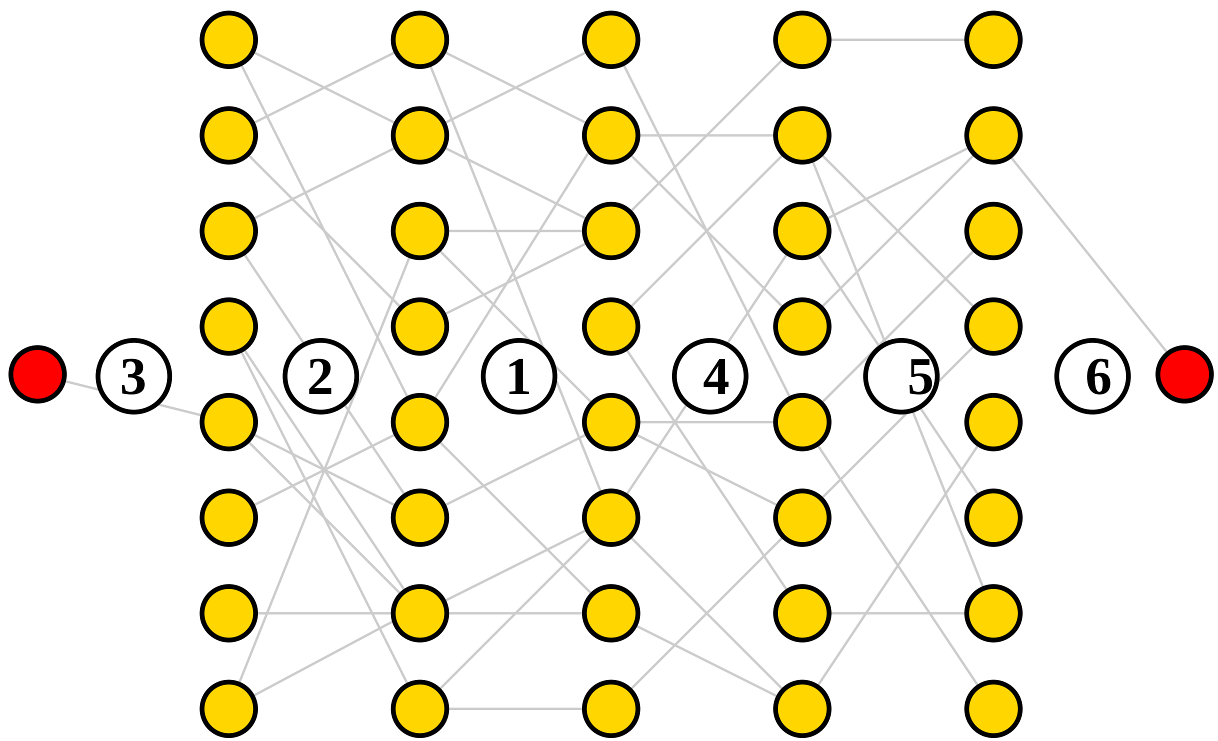
Is there a path of length 6 between red nodes?



Stream = ① ② ③ ④ ⑤ ⑥

Our Stream Ordering

Is there a path of length 6 between red nodes?

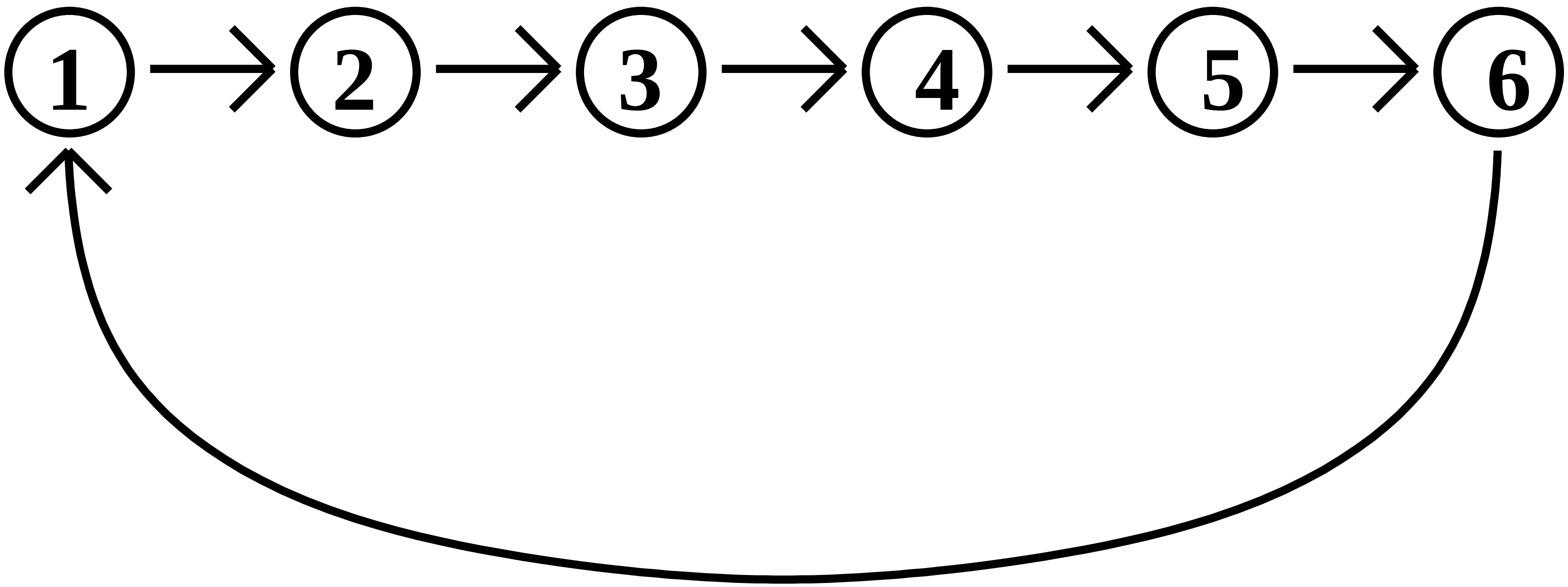


Stream = ① ② ③ ④ ⑤ ⑥

③ ② ① ⑥ ⑤ ④ is easy in $O(n)$ space

Streaming and Communication Protocols

- Assign each layer to one player



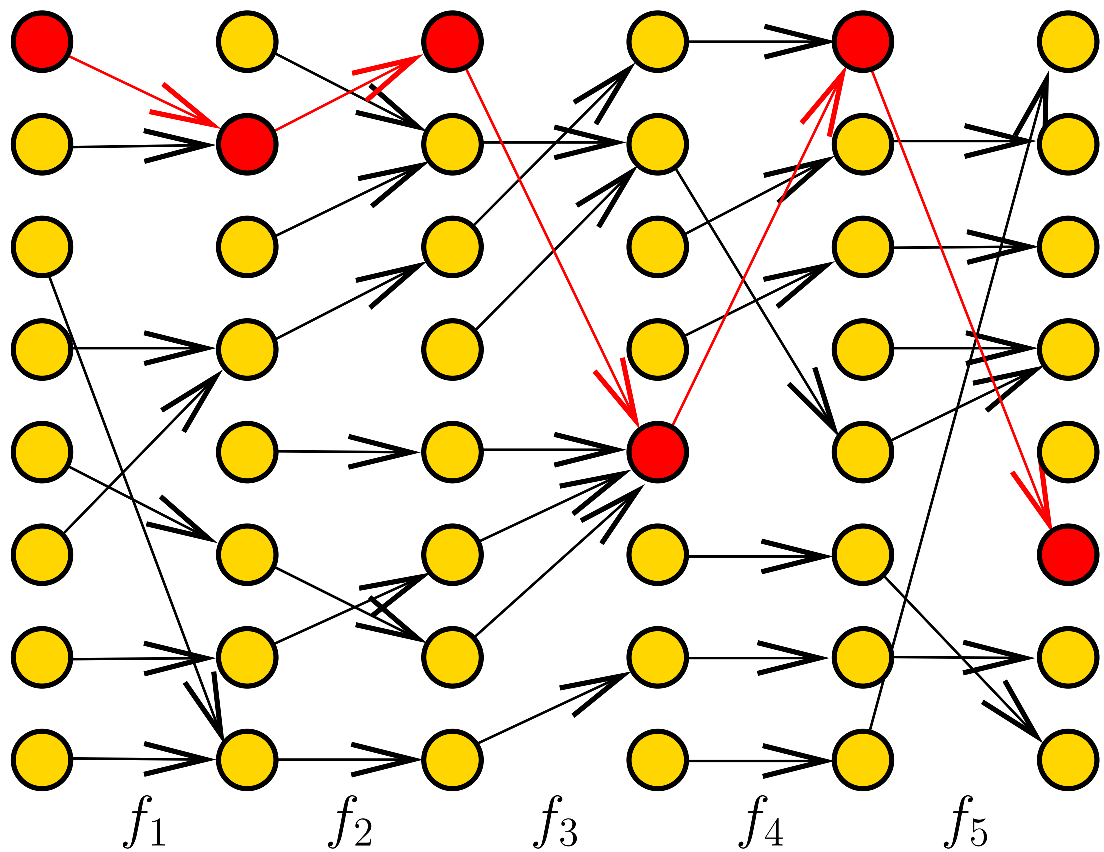
- Small-space streaming algorithm
  $\Rightarrow$ efficient communication protocol
- Goal: prove communication lower bound

The Proof

Important Problem: Pointer Chasing

Definition:

- Input: p functions $f_i : [n] \to [n]$
- Goal: Compute $f_p(f_{p-1}(\ldots f_2(f_1(1))\ldots))$



Important Problem: Pointer Chasing

Definition:

- Input: p functions $f_i : [n] \to [n]$
- Goal: Compute $f_p(f_{p-1}(\ldots f_2(f_1(1))\ldots))$

Two-player version:

- What players have:

Alice	Bob
$f_2, f_4, f_6, \ldots$	$f_1, f_3, f_5, \ldots$

- Alice speaks first

Important Problem: Pointer Chasing

Definition:

- Input: p functions $f_i : [n] \to [n]$
- Goal: Compute $f_p(f_{p-1}(\ldots f_2(f_1(1))\ldots))$

Two-player version:

- What players have:

Alice	Bob
$f_2, f_4, f_6, \ldots$	$f_1, f_3, f_5, \ldots$

- Alice speaks first
- Nisan, Wigderson (1993):

Computing in less then $p = \Theta(1)$ messages
of communication requires $\Omega(n)$ communication

Important Problem: Pointer Chasing

Definition:

- Input: p functions $f_i : [n] \to [n]$
- Goal: Compute $f_p(f_{p-1}(\ldots f_2(f_1(1))\ldots))$

p-player version:

- What players have:

Player 1	Player 2	...	Player $p-1$	Player p
f_p	f_{p-1}	...	f_2	f_1

- Each round: players speak in order Player 1 through Player p

Important Problem: Pointer Chasing

Definition:

- Input: p functions $f_i : [n] \to [n]$
- Goal: Compute $f_p(f_{p-1}(\ldots f_2(f_1(1))\ldots))$

p-player version:

- What players have:

Player 1	Player 2	...	Player $p-1$	Player p
f_p	f_{p-1}	...	f_2	f_1

- Each round: players speak in order Player 1 through Player p
- Guha, McGregor (2007):

Computing in less then $p = \Theta(1)$ rounds
requires $\Omega(n)$ communication

FKMSZ Lower Bound for BFS

Their Problem: Compute p levels of BFS tree from v

FKMSZ Lower Bound for BFS

Their Problem: Compute p levels of BFS tree from v

Sketch of their proof:

- Take communication lower bound for pointer chasing

FKMSZ Lower Bound for BFS

Their Problem: Compute p levels of BFS tree from v

Sketch of their proof:

- Take communication lower bound for pointer chasing
- Apply direct sum theorem of [Jain, Radhakrishnan, Sen (2003)]:
  - Solving k instances requires k times more communication

FKMSZ Lower Bound for BFS

Their Problem: Compute p levels of BFS tree from v

Sketch of their proof:

- Take communication lower bound for pointer chasing
- Apply direct sum theorem of [Jain, Radhakrishnan, Sen (2003)]:
  - Solving k instances requires k times more communication
- Show: Computing a level p BFS tree in graph of degree $k = n^{\Theta(1/p)}$ enables solving k instances of pointer chasing.

FKMSZ Lower Bound for BFS

Their Problem: Compute p levels of BFS tree from v

Sketch of their proof:

- Take communication lower bound for pointer chasing
- Apply direct sum theorem of [Jain, Radhakrishnan, Sen (2003)]:
  - Solving k instances requires k times more communication
- Show: Computing a level p BFS tree in graph of degree $k = n^{\Theta(1/p)}$ enables solving k instances of pointer chasing.

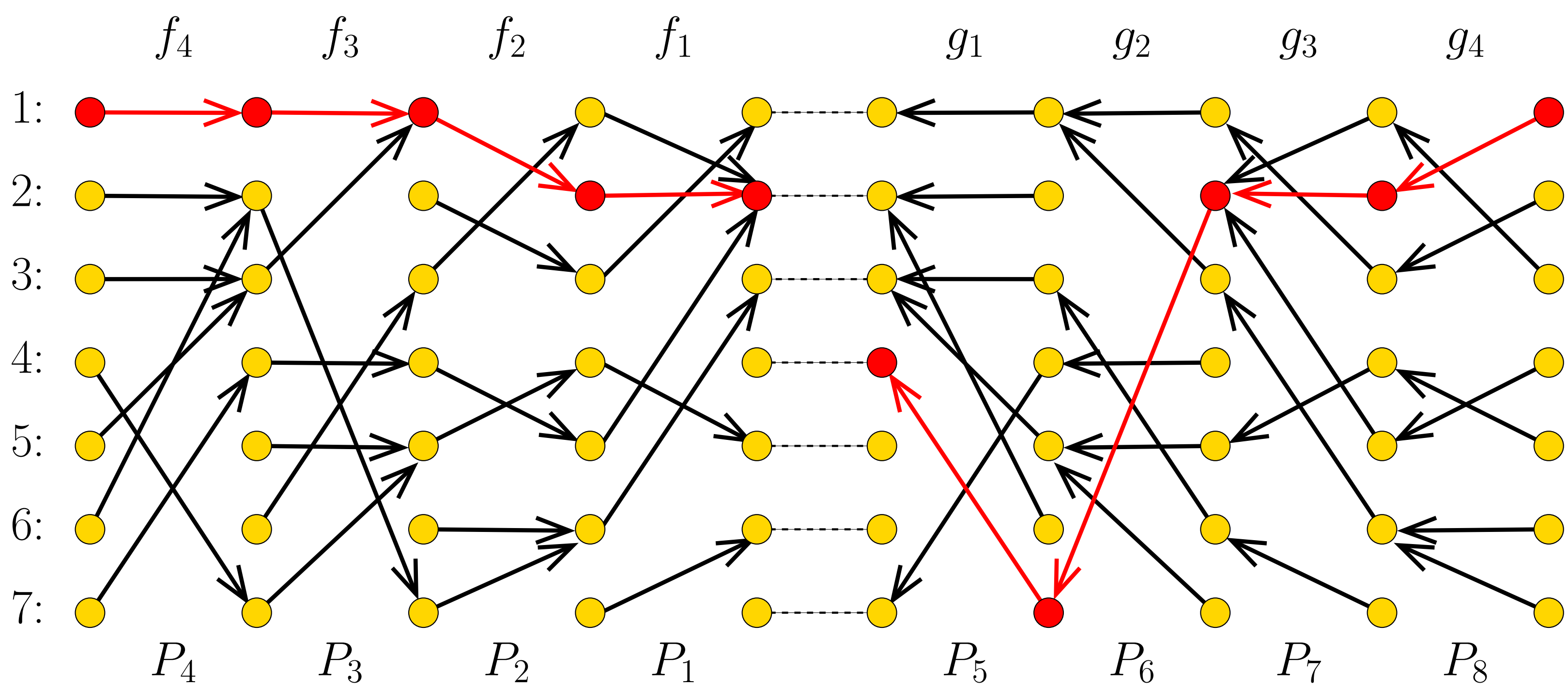
Our problem:

- Only need to check if BFS trees intersect
- Seems hard to infer full tree from this

Proof Overview

Problem BBB (Basic Building Block):

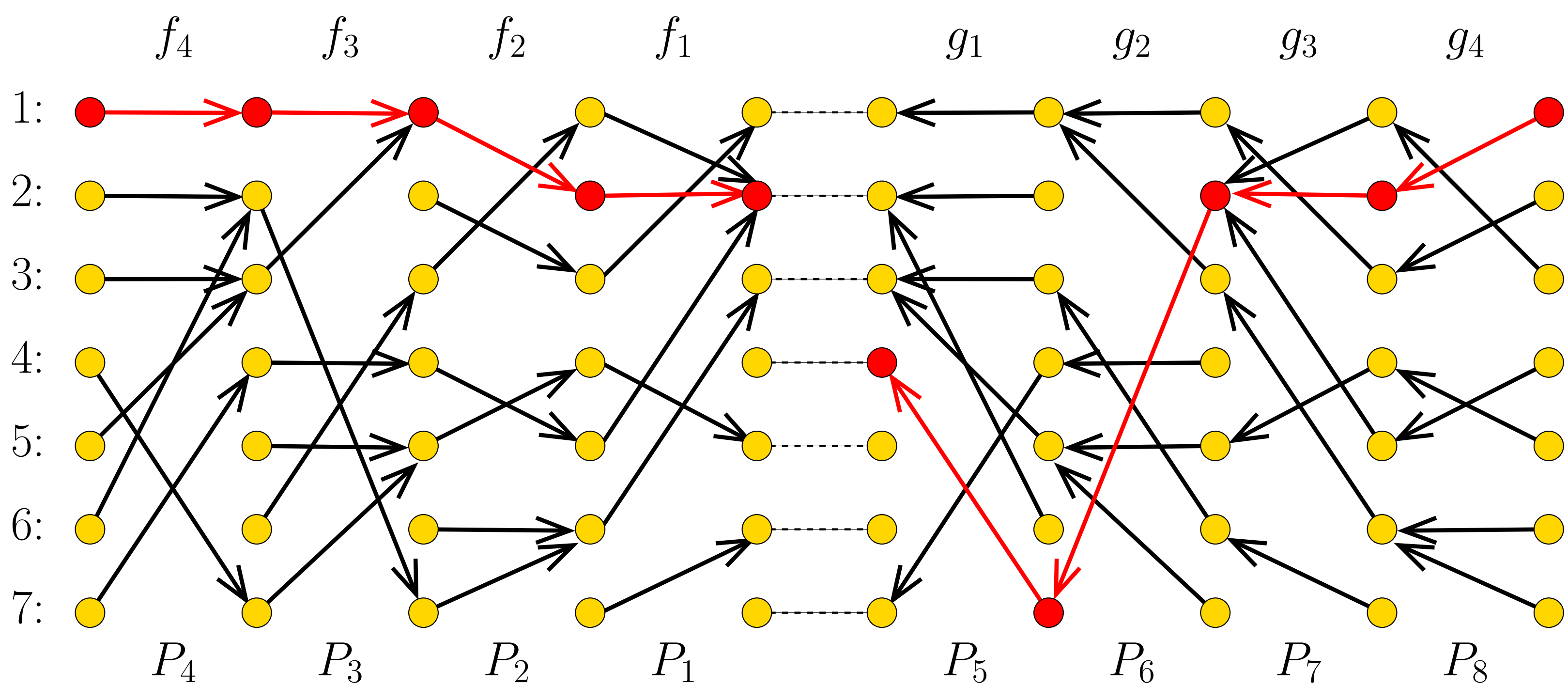
- $2p$ players with two instances of pointer chasing



Proof Overview

Problem BBB (Basic Building Block):

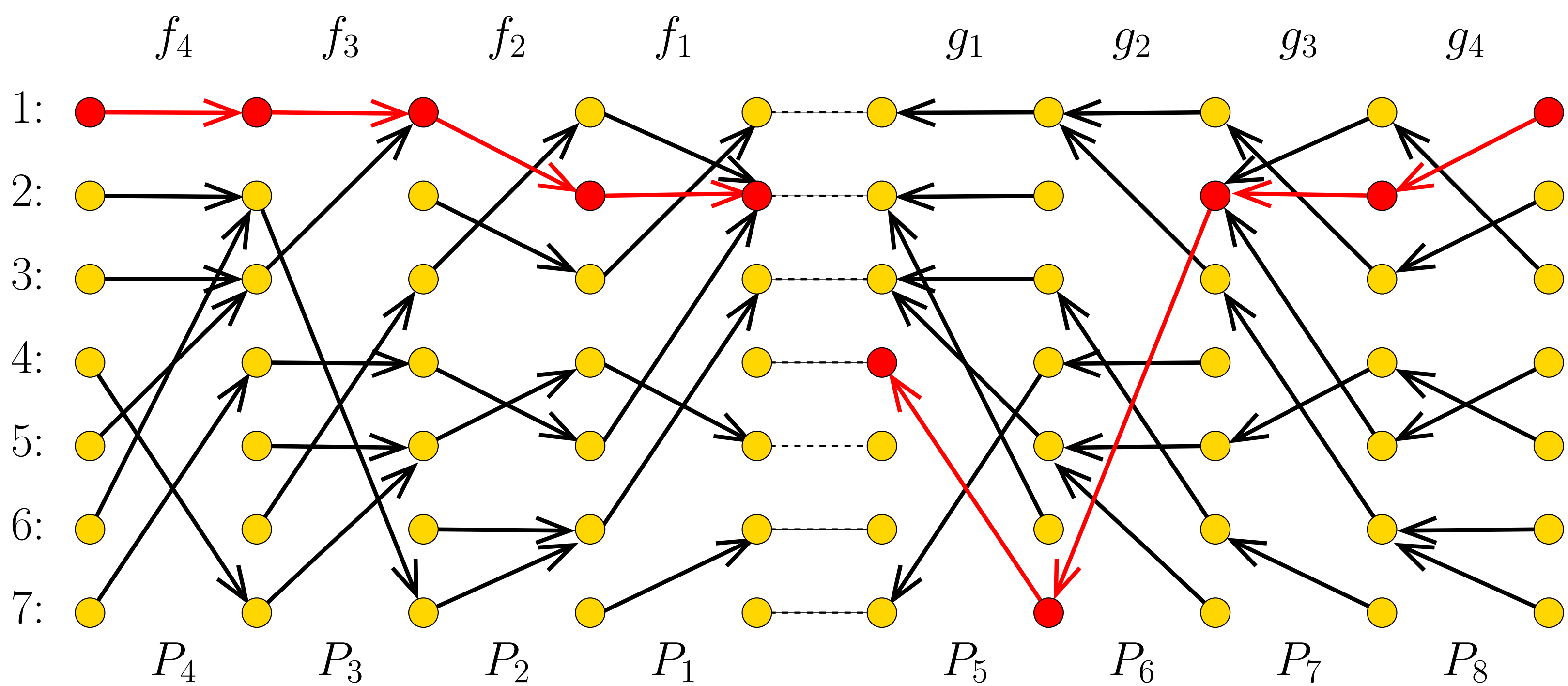
- $2p$ players with two instances of pointer chasing
- Problem to solve: Is the result the same?



Proof Overview

Problem BBB (Basic Building Block):

- $2p$ players with two instances of pointer chasing
- Problem to solve: Is the result the same?
- (If some function maps $\Omega(\log n)$ elements to one element, also say YES)



Proof Overview

Problem BBB (Basic Building Block):

- $2p$ players with two instances of pointer chasing
- Problem to solve: Is the result the same?
- (If some function maps $\Omega(\log n)$ elements to one element, also say YES)

Three Steps: ($\mu =$ uniform distribution)

1. $\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$

Proof Overview

Problem BBB (Basic Building Block):

- $2p$ players with two instances of pointer chasing
- Problem to solve: Is the result the same?
- (If some function maps $\Omega(\log n)$ elements to one element, also say YES)

Three Steps: ($\mu =$ uniform distribution)

1. $\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$
2. $\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn)$ for $k \ll n$

Proof Overview

Problem BBB (Basic Building Block):

- $2p$ players with two instances of pointer chasing
- Problem to solve: Is the result the same?
- (If some function maps $\Omega(\log n)$ elements to one element, also say YES)

Three Steps: ($\mu =$ uniform distribution)

1. $\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$
2. $\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn)$ for $k \ll n$
   - Implies: $\mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim \Omega(kn)$

Proof Overview

Problem BBB (Basic Building Block):

- $2p$ players with two instances of pointer chasing
- Problem to solve: Is the result the same?
- (If some function maps $\Omega(\log n)$ elements to one element, also say YES)

Three Steps: ($\mu =$ uniform distribution)

1. $\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$
2. $\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn)$ for $k \ll n$
   - Implies: $\mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim \Omega(kn)$
3. $\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$ for $k = n^{O(1/p)}$

Step 3

$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$

for $k = n^{\Theta(1/p)}$

Step 3

$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$

for $k = n^{\Theta(1/p)}$

Want: Protocol for $\bigvee_{i=1}^{k}$ BBB using protocol for BFS intersection

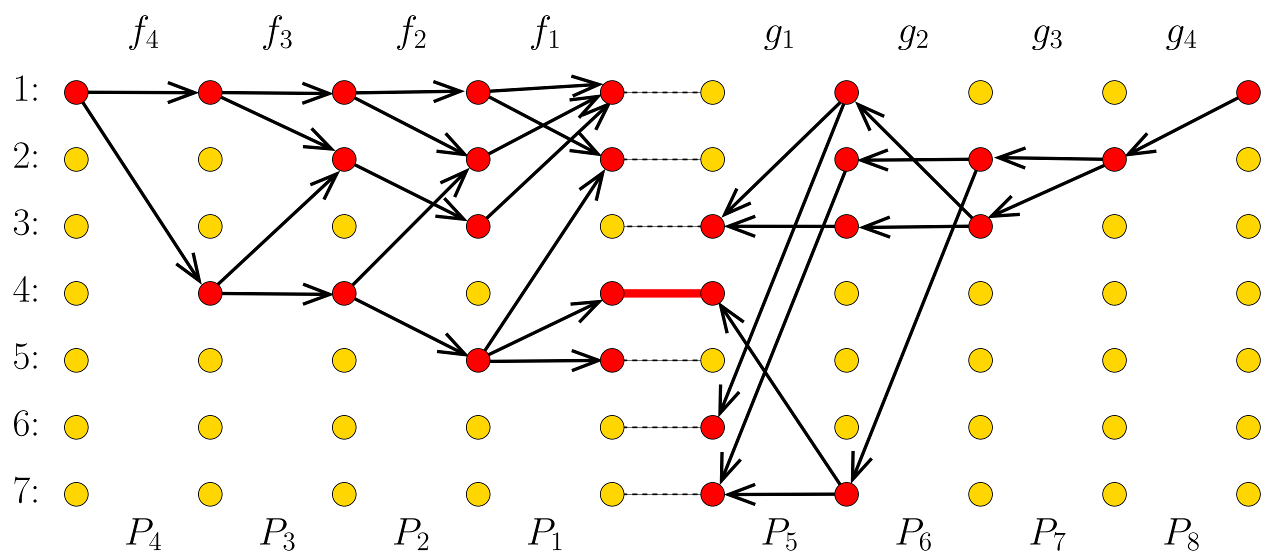
Step 3

$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$

for $k = n^{\Theta(1/p)}$

Want: Protocol for $\bigvee_{i=1}^{k}$ BBB using protocol for BFS intersection

- "Stack" k instances of BBB on top of each other

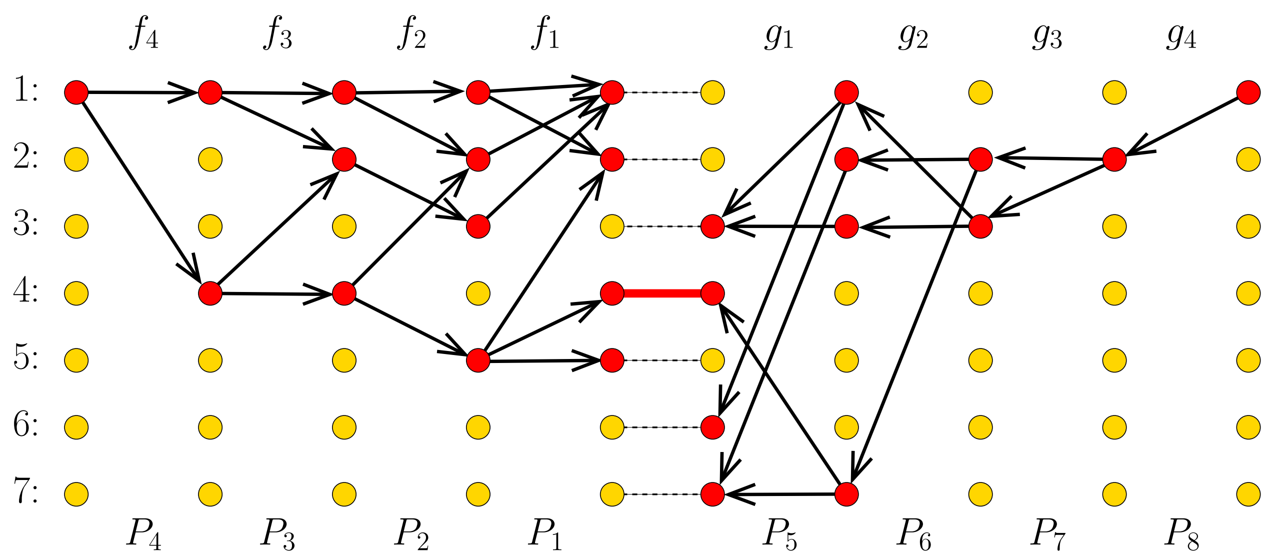


Step 3

$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$
for $k = n^{\Theta(1/p)}$

Want: Protocol for $\bigvee_{i=1}^{k}$ BBB using protocol for BFS intersection

- "Stack" k instances of BBB on top of each other



Gives instance of BFS tree intersection, but pointers from two different instances may intersect

Step 3

$$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$$

for $k = n^{\Theta(1/p)}$

Want: Protocol for $\bigvee_{i=1}^{k}$ BBB using protocol for BFS intersection

- Randomly relabel intermediate results of functions and stack them on top of each other

Step 3

$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$
for $k = n^{\Theta(1/p)}$

Want: Protocol for $\bigvee_{i=1}^{k}$ BBB using protocol for BFS intersection

- Randomly relabel intermediate results of functions and stack them on top of each other
  - If pair of pointer chasing instances gives the same element, BFS trees intersect

Step 3

$\mathrm{CC}_{1/20}(\text{BFS tree intersection}) \gtrsim \mathrm{CC}_{1/10}(\bigvee_{i=1}^{k} \text{BBB})$

for $k = n^{\Theta(1/p)}$

Want: Protocol for $\bigvee_{i=1}^{k}$ BBB using protocol for BFS intersection

- Randomly relabel intermediate results of functions and stack them on top of each other
  - If pair of pointer chasing instances gives the same element, BFS trees intersect
  - $k^p \ll n$ and random scrambling $\Longrightarrow$ If no pair gives the same element (and no $\Theta(\log n)$-to-1 mapping), BFS trees unlikely to intersect

Step 2

Statement:

$\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn)$ for $k \ll n$

Step 2

Statement:

$\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \textsf{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(kn)$ for $k \ll n$

How:

- Product distribution:
  information cost $= \sum_{i=1}^{k}$ information cost on instance i

Step 2

Statement:

$\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^k \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn)$ for $k \ll n$

How:

- Product distribution:
  information cost $= \sum_{i=1}^k$ information cost on instance i
- Trivial, if all instances must be solved.
  The problem asks only for $\bigvee$ (the instances)

Step 2

Statement:

$\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \textsf{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(kn)$ for $k \ll n$

How:

- Product distribution:
  information cost $= \sum_{i=1}^{k}$ information cost on instance i
- Trivial, if all instances must be solved.
  The problem asks only for $\bigvee$ (the instances)
- For specific instance,
  $\bigvee$(other instances) $=$ false most of the time

Step 2

Statement:

$$\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn) \text{ for } k \ll n$$

How:

- Product distribution:
  information cost $= \sum_{i=1}^{k}$ information cost on instance i
- Trivial, if all instances must be solved.
  The problem asks only for $\bigvee$ (the instances)
- For specific instance,
  $\bigvee$(other instances) $=$ false most of the time
- Fix at random other instances s.t. $\bigvee$(other instances) $=$ false
  $\Rightarrow$ protocol must solve the instance

Step 2

Statement:

$$\mathrm{IC}_{\mu^k,1/(2n^2)}(\bigvee_{i=1}^{k} \text{BBB}) \gtrsim k \cdot \mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(kn) \text{ for } k \ll n$$

How:

- Product distribution:
  information cost $= \sum_{i=1}^{k}$ information cost on instance i
- Trivial, if all instances must be solved.
  The problem asks only for $\bigvee$ (the instances)
- For specific instance,
  $\bigvee$(other instances) $=$ false most of the time
- Fix at random other instances s.t. $\bigvee$(other instances) $=$ false
  $\Rightarrow$ protocol must solve the instance
- Information cost won't decrease significantly on
  $\bigvee$(other instances) $=$ true

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

What is known:

- communication complexity for pointer chasing is $\Omega(n)$ for uniform distribution [Nisan, Wigderson 1993], [Guha, McGregor 2007]

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

What is known:

- communication complexity for pointer chasing is $\Omega(n)$ for uniform distribution [Nisan, Wigderson 1993], [Guha, McGregor 2007]

Obstacles:

1. Need a proof for information complexity
2. Equality of pointer chasing instances
   - Need to account for impact of $\Theta(\log n)$-to-1 maps

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\mathsf{BBB}) \approx \Omega(n)$$

Obstacle 1: Need a proof for information complexity

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Obstacle 1: Need a proof for information complexity

- Use [Jain, Radhakrishnan, Sen 2003]?
- $\Pi =$ constant-round protocol revealing information IC with error ϵ:

  There is a protocol Π' with total communication $\sim \mathrm{IC}/\delta^2$ that errs with probability $\epsilon + \delta$

  i.e., "small information $\Rightarrow$ small communication"

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Obstacle 1: Need a proof for information complexity

- Use [Jain, Radhakrishnan, Sen 2003]?
- $\Pi =$ constant-round protocol revealing information IC with error ϵ:

  There is a protocol Π' with total communication $\sim \mathrm{IC}/\delta^2$ that errs with probability $\epsilon + \delta$

  i.e., "small information $\Rightarrow$ small communication"

- Won't suffice for us: $\delta = o(1/n)$

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Use techniques of [JRS] to produce a protocol Π'
  - Π' is deterministic
  - errs with twice the probability
  - sends messages of length $\leq \mathrm{IC} \cdot p^{O(1)}$ with probability $1 - p^{-\Omega(1)}$

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Use techniques of [JRS] to produce a protocol Π'
  - Π' is deterministic
  - errs with twice the probability
  - sends messages of length $\leq \mathrm{IC} \cdot p^{O(1)}$ with probability $1 - p^{-\Omega(1)}$

“Typically concise” protocol

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Use techniques of [JRS] to produce a protocol Π'
  - Π' is deterministic
  - errs with twice the probability
  - sends messages of length $\leq \mathrm{IC} \cdot p^{O(1)}$ with probability $1 - p^{-\Omega(1)}$

“Typically concise” protocol

- Note: prob. of long message $\gg$ prob. of answer YES

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Modify [NW] for “typically concise” protocols and equality

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Modify [NW] for “typically concise” protocols and equality
- Original argument for protocol tree:

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\mathsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Modify [NW] for “typically concise” protocols and equality
- Original argument for protocol tree:
  - simulate in parallel trivial algorithm:
    make step forward when possible

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Modify [NW] for “typically concise” protocols and equality
- Original argument for protocol tree:
  - simulate in parallel trivial algorithm:
    make step forward when possible
  - $x_{\mathrm{current}} =$ current value in this algorithm

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Modify [NW] for "typically concise" protocols and equality
- Original argument for protocol tree:
  - simulate in parallel trivial algorithm:
    make step forward when possible
  - x_{current} = current value in this algorithm
  - by induction, $H(f_{\mathrm{next}}(x_{\mathrm{current}})) = \log n - o(1)$
    - for $1 - o(1)$ fraction of internal nodes
    - for $1 - o(1)$ fraction of leaves

    with $o(n)$ communication

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Modify [NW] for "typically concise" protocols and equality
- Original argument for protocol tree:
  - simulate in parallel trivial algorithm:
    make step forward when possible
  - $x_{\text{current}} =$ current value in this algorithm
  - by induction, $H(f_{\text{next}}(x_{\text{current}})) = \log n - o(1)$
    - for $1 - o(1)$ fraction of internal nodes
    - for $1 - o(1)$ fraction of leaves

    with $o(n)$ communication
  - with this entropy, prob. of correct solution is $o(1)$

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\mathsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- simulate in parallel trivial algorithm for both instances:
  make step forward when possible

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- simulate in parallel trivial algorithm for both instances:
  make step forward when possible
- $(x_{\text{current}}, y_{\text{current}}) =$ current values in this algorithm

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\mathsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- simulate in parallel trivial algorithm for both instances:
  make step forward when possible
- $(x_{\mathrm{current}}, y_{\mathrm{current}})$ = current values in this algorithm
- by induction, $H(f_{\mathrm{next}}(x_{\mathrm{current}})) = \log n - o(1)$
  and $H(g_{\mathrm{next}}(y_{\mathrm{current}})) = \log n - o(1)$
  - for $1 - o(1)$ fraction of internal nodes
  - for $1 - o(1)$ fraction of leaves

  with $o(n)$ communication

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- simulate in parallel trivial algorithm for both instances:
  make step forward when possible
- $(x_{\text{current}}, y_{\text{current}})$ = current values in this algorithm
- by induction, $H(f_{\text{next}}(x_{\text{current}})) = \log n - o(1)$
  and $H(g_{\text{next}}(y_{\text{current}})) = \log n - o(1)$
  - for $1 - o(1)$ fraction of internal nodes
  - for $1 - o(1)$ fraction of leaves

  with $o(n)$ communication
  (impact of rare long messages small)

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Entropy of solution to each pointer chasing $\log n - o(1)$
- Probability $\Omega(1/n)$ for $\frac{3}{4}n$ elements

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Entropy of solution to each pointer chasing $\log n - o(1)$
- Probability $\Omega(1/n)$ for $\frac{3}{4}n$ elements
- Distributions independent:
  - deterministic protocol
  - pointer chasing instances held by different players
  - product input distribution

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Entropy of solution to each pointer chasing $\log n - o(1)$
- Probability $\Omega(1/n)$ for $\frac{3}{4}n$ elements
- Distributions independent:
  - deterministic protocol
  - pointer chasing instances held by different players
  - product input distribution
- must collide with probability

  $$n/4 \cdot \Omega(1/n)^2 = \Omega(1/n)$$

Step 1

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\text{BBB}) \approx \Omega(n)$$

Our solution (part 2):

- Entropy of solution to each pointer chasing $\log n - o(1)$
- Probability $\Omega(1/n)$ for $\frac{3}{4}n$ elements
- Distributions independent:
  - deterministic protocol
  - pointer chasing instances held by different players
  - product input distribution
- must collide with probability
  $$n/4 \cdot \Omega(1/n)^2 = \Omega(1/n)$$
- protocol errs with probability $\Omega(1/n)$

Main result:

Shortest Path, Perfect Matching, and Directed Connectivity require $\sim n^{1+\Omega(1/p)}$ space in p passes

Main result:

Shortest Path, Perfect Matching, and Directed Connectivity require $\sim n^{1+\Omega(1/p)}$ space in p passes

Open Questions:

- Simpler proof?
- Improve lower bounds from $\sim\Omega(n^{1+1/(2p)})$ to $\sim\Omega(n^{1+1/p})$?

Main result:

Shortest Path, Perfect Matching, and Directed Connectivity require $\sim n^{1+\Omega(1/p)}$ space in p passes

Open Questions:

- Simpler proof?
- Improve lower bounds from $\sim\Omega(n^{1+1/(2p)})$ to $\sim\Omega(n^{1+1/p})$?
- Better bounds for maximum matching?
  - Is looking for a few augmenting paths harder?
  - Can the techniques be used for approximate matchings?

Questions?

Step 1 (more details)

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Conversion based on “message compression” of [JRS]

Step 1 (more details)

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Conversion based on “message compression” of [JRS]
- Replace messages by induction from last to first

Step 1 (more details)

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Conversion based on “message compression” of [JRS]
- Replace messages by induction from last to first
- More details (for message sent by P_i):
  - Consider two distributions:
    - $\mathcal{D}$ = message distribution if P_i's input unknown
    - $\mathcal{D}'$ = message distribution if P_i's input known

Step 1 (more details)

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\mathsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Conversion based on "message compression" of [JRS]
- Replace messages by induction from last to first
- More details (for message sent by P_i):
  - Consider two distributions:
    - $\mathcal{D}$ = message distribution if P_i's input unknown
    - $\mathcal{D}'$ = message distribution if P_i's input known
  - Public randomness $\Rightarrow$ sequence of samples from $\mathcal{D}$

Step 1 (more details)

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Conversion based on "message compression" of [JRS]
- Replace messages by induction from last to first
- More details (for message sent by P_i):
  - Consider two distributions:
    - $\mathcal{D}$ = message distribution if P_i's input unknown
    - $\mathcal{D}'$ = message distribution if P_i's input known
  - Public randomness $\Rightarrow$ sequence of samples from $\mathcal{D}$
  - In expectation KL-divergence between $\mathcal{D}$ and $\mathcal{D}'$ bounded by information revealed about P_i's input

Step 1 (more details)

Statement:

$$\mathrm{IC}_{\mu,1/n^2}(\textsf{BBB}) \approx \Omega(n)$$

Our solution (part 1):

- Conversion based on "message compression" of [JRS]
- Replace messages by induction from last to first
- More details (for message sent by P_i):
  - Consider two distributions:
    - $\mathcal{D} \;=$ message distribution if P_i's input unknown
    - $\mathcal{D}' =$ message distribution if P_i's input known
  - Public randomness $\Rightarrow$ sequence of samples from $\mathcal{D}$
  - In expectation KL-divergence between $\mathcal{D}$ and $\mathcal{D}'$ bounded by information revealed about P_i's input
  - For most nodes and messages, index of the actual message of P_i is small