Chapter 4.

Deeper Discussions

Danupon Nanongkai

KTH, Sweden

ADFOCS 2018

Last edited: Aug. 16, 2018

Plan

- Query-update time tradeoffs
- Other conjectures
- Unconditional lower bounds
- Partially-dynamic algorithms

Part 1

Query-Update Time Tradeoffs

Motivation

- So far, we focuses on outputting something small (yes/no, numbers) after each update.
- More realistic: output when users want.
- Also: Users may just want **part** of the (large) output.

Example

Single-Source Reachability

with **queries**

Thanks Thatchaphol Saranurak for slides

How should we define single-source reachability?

Option 1: Output list of reachable nodes

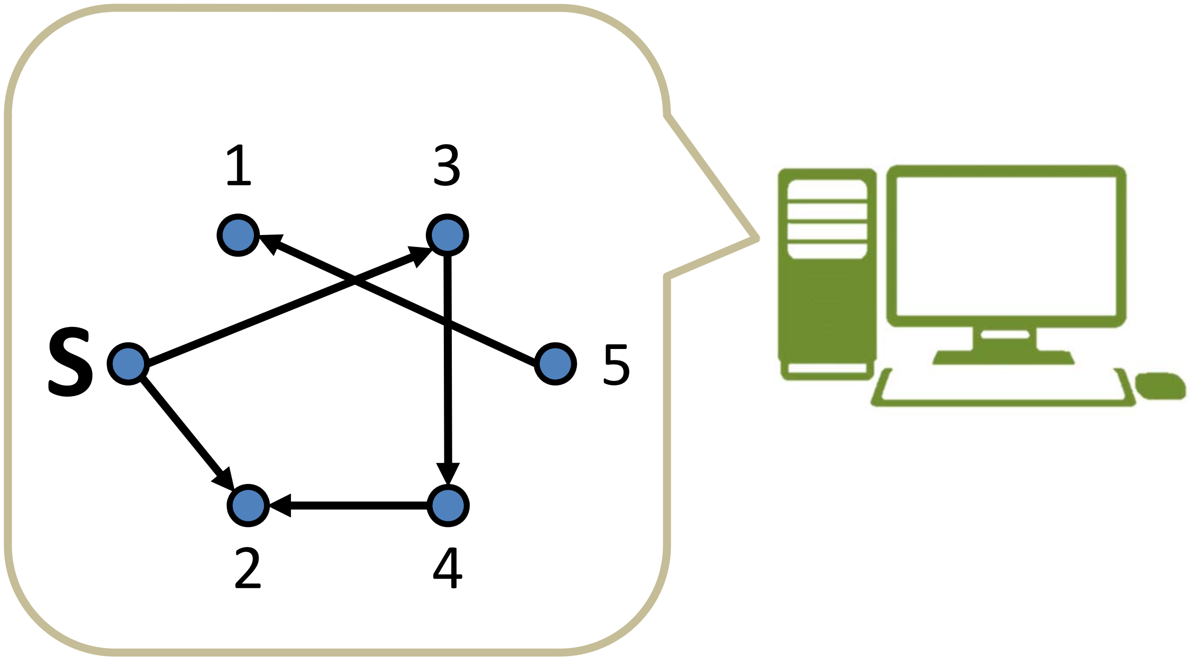
- $\Omega(n)$ is an obvious update time lower bound
- ... not so interesting

Option 2: Answer query "Can s reach u?"

- Possible to get polylog update time in this case?
- Let's look into this

Single Source Reachability (ss-Reach)

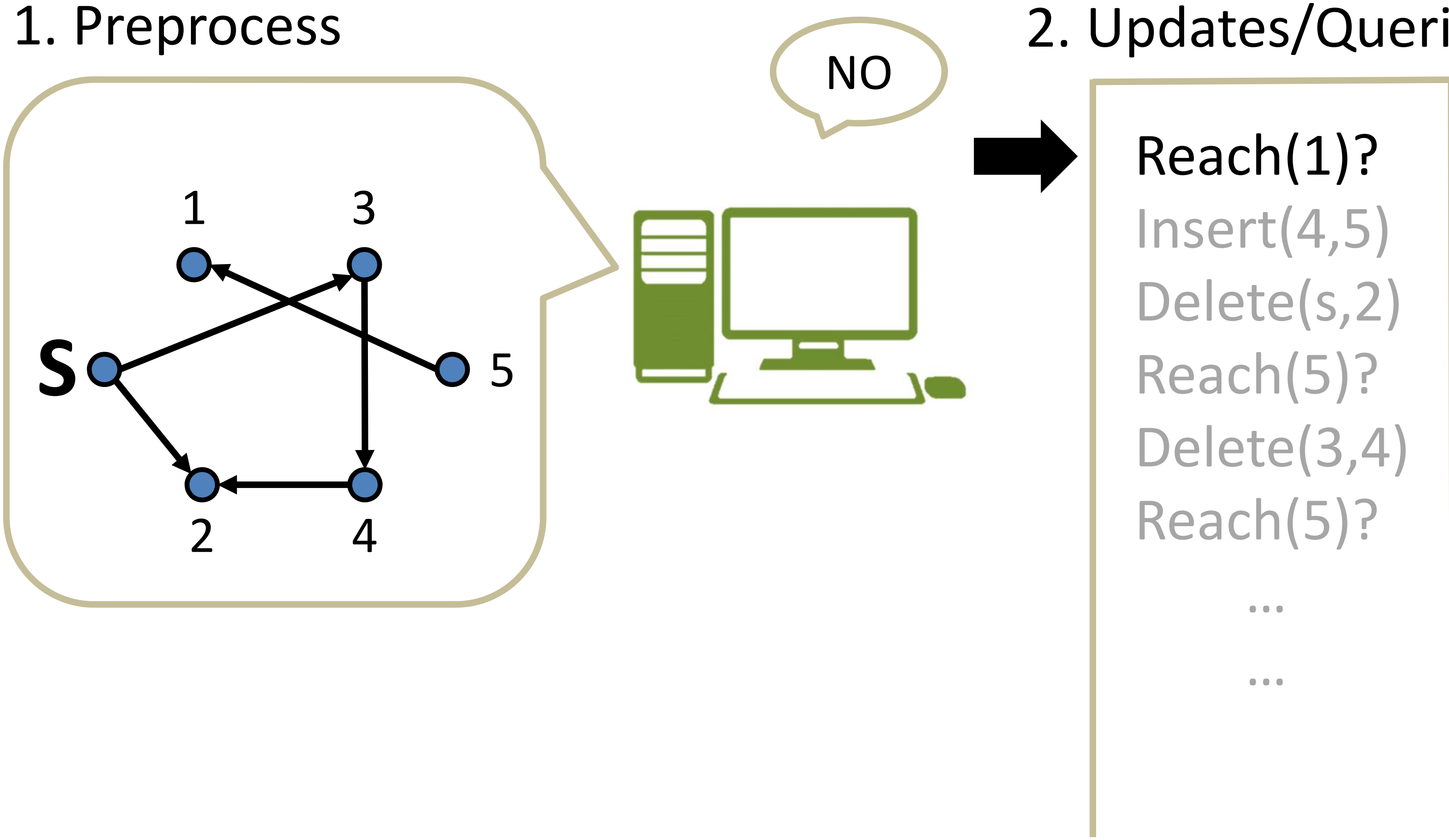
1. Preprocess



Single Source Reachability (ss-Reach)

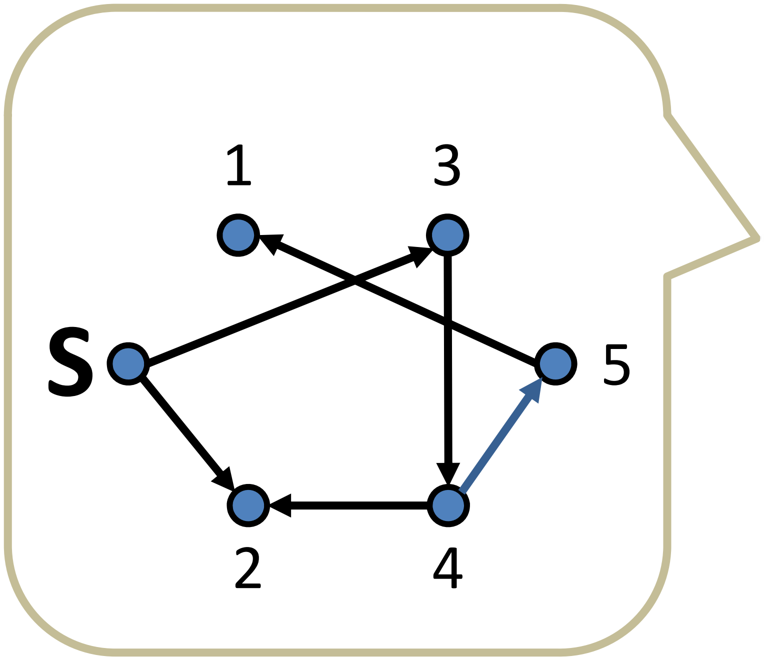
1. Preprocess

2. Updates/Queries



Single Source Reachability (ss-Reach)

1. Preprocess

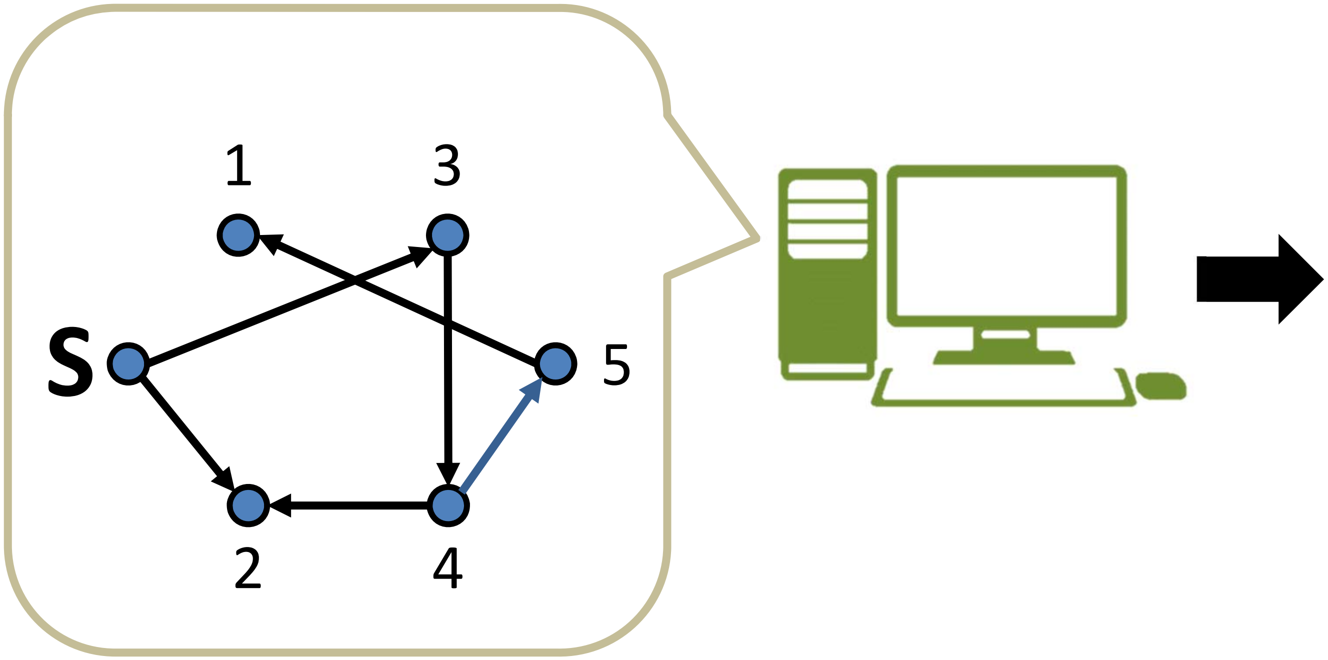


2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...

Single Source Reachability (ss-Reach)

1. Preprocess

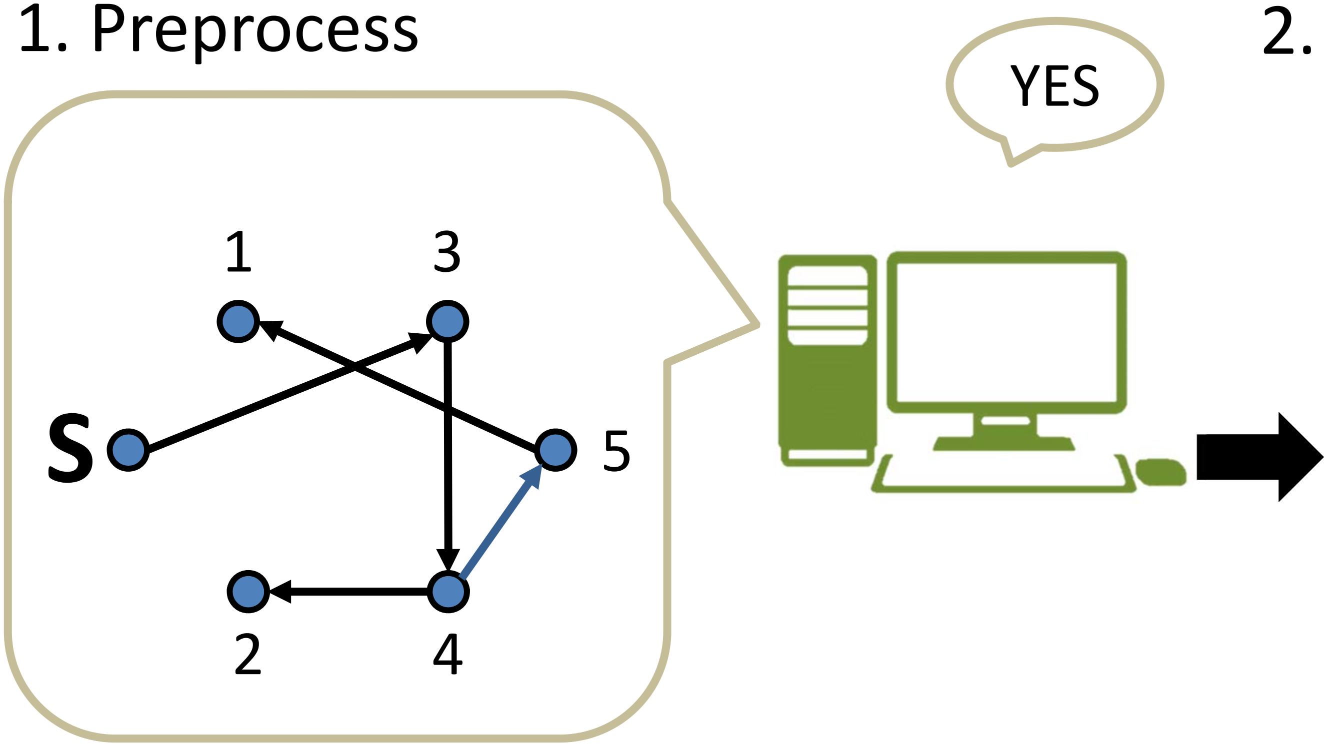


2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...

Single Source Reachability (ss-Reach)

1. Preprocess



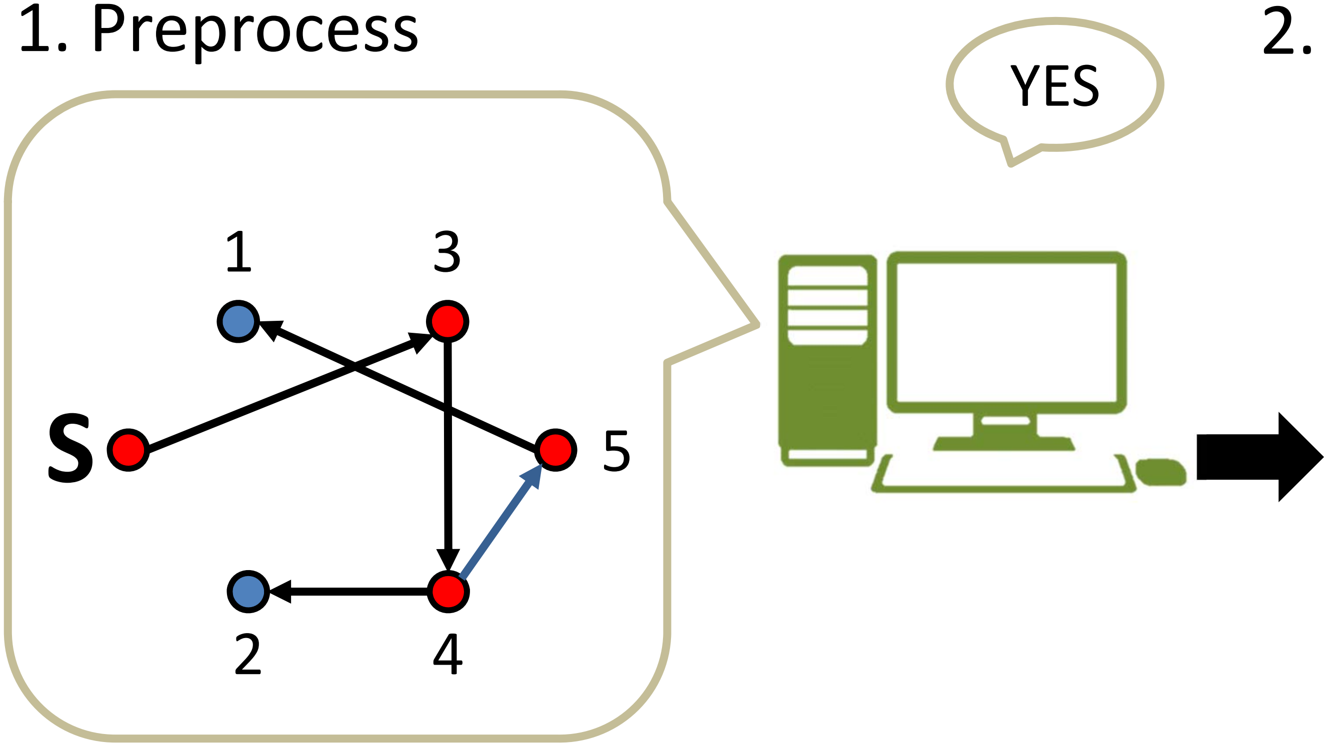
2. Updates/Queries

Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...

Single Source Reachability (ss-Reach)

1. Preprocess

2. Updates/Queries



Reach(1)?
Insert(4,5)
Delete(s,2)
Reach(5)?
Delete(3,4)
Reach(5)?
...
...

A **Naïve** Algorithm for (fully dynamic) ss-Reach:

	Update time	Query time
BFS from s when update	m	1

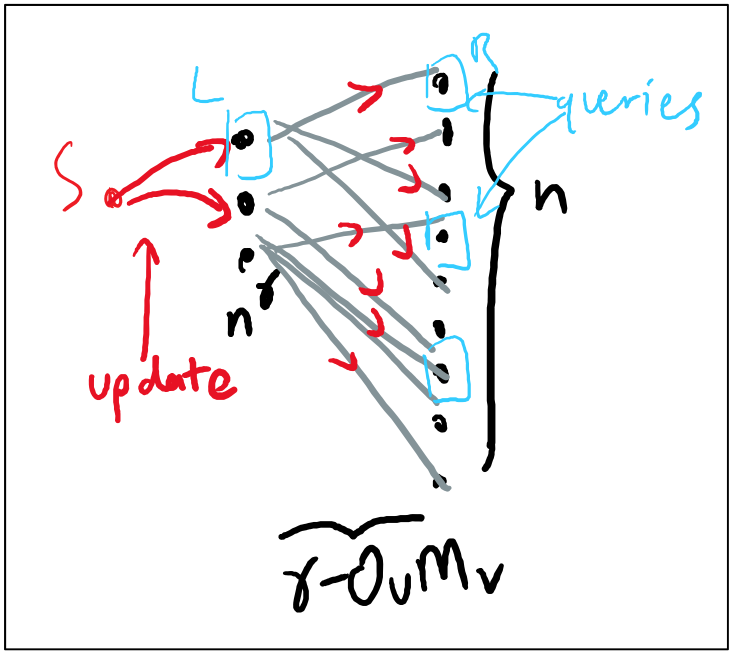
Can we improve update time

$m \Rightarrow m^{1-\epsilon}$? (maybe amortized)

m = max #edges
n = #nodes

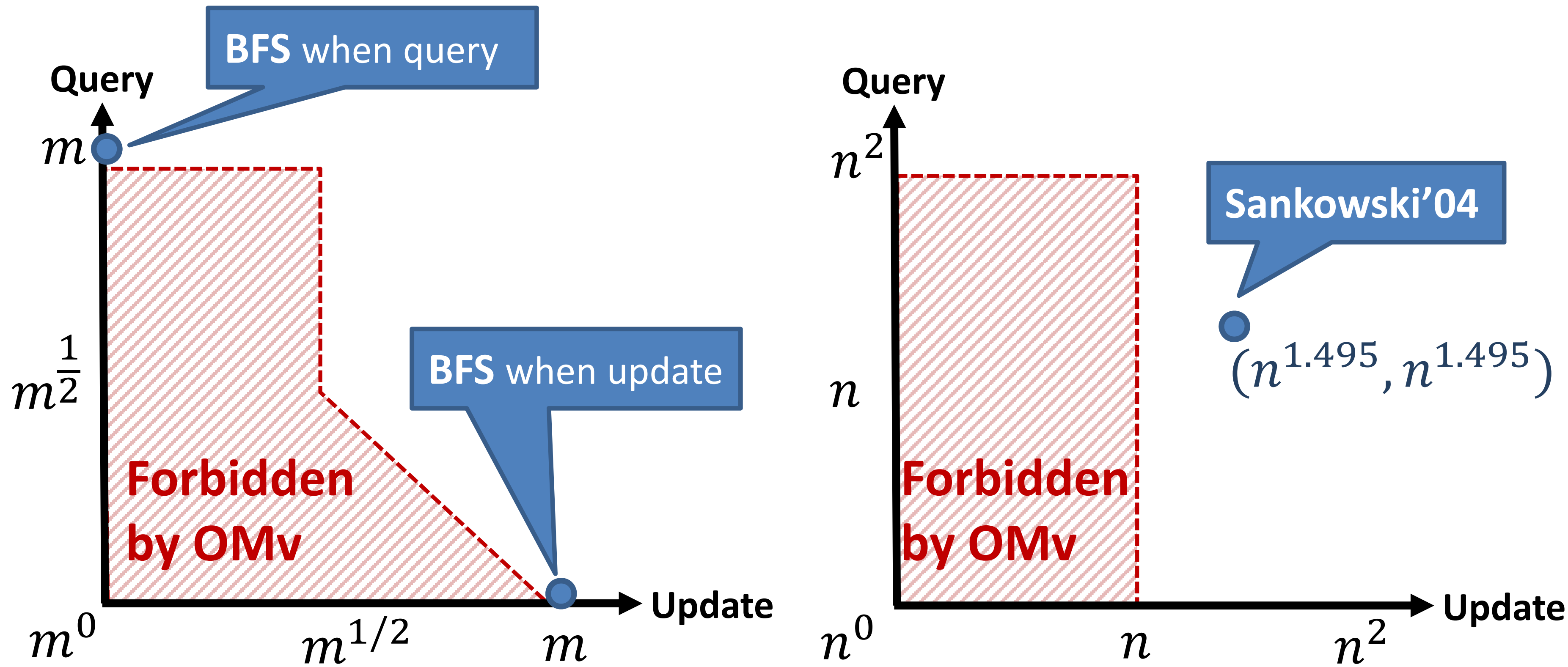
Reduction from γ-OuMv to ss-Reach

(sketched)



Sketch:

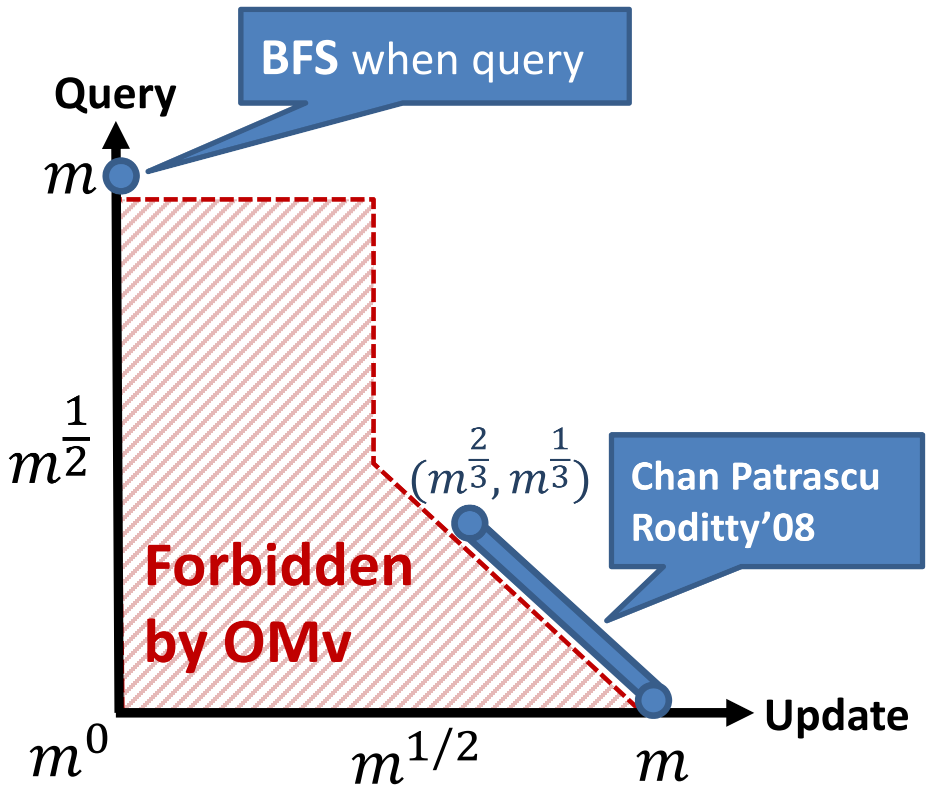
1. For each (u_i, v_i): $\boldsymbol{n^\gamma}$ updates and $\boldsymbol{n}$ queries
2. **γ-OuMv** implies that amortized time over $\boldsymbol{n^\gamma}$ updates and $\boldsymbol{n}$ queries cannot be $\boldsymbol{O(n^{1+\gamma-\epsilon})}$ for any $\epsilon > 0$.
3. If query time is $n^{o(1)}$, then update time cannot be $\boldsymbol{O(n^{1-\epsilon})}$ for any $\epsilon > 0$.
4. For any $\boldsymbol{\epsilon'} > 0$, update time of $\mathbf{O}(\boldsymbol{m^{1-\epsilon'}})$ implies amortized time of $\boldsymbol{O(n^{(1+\gamma)(1-\epsilon')})}$.
5. … which is $O(n^{1-\epsilon})$ for some $\epsilon > 0$ for small enough γ.

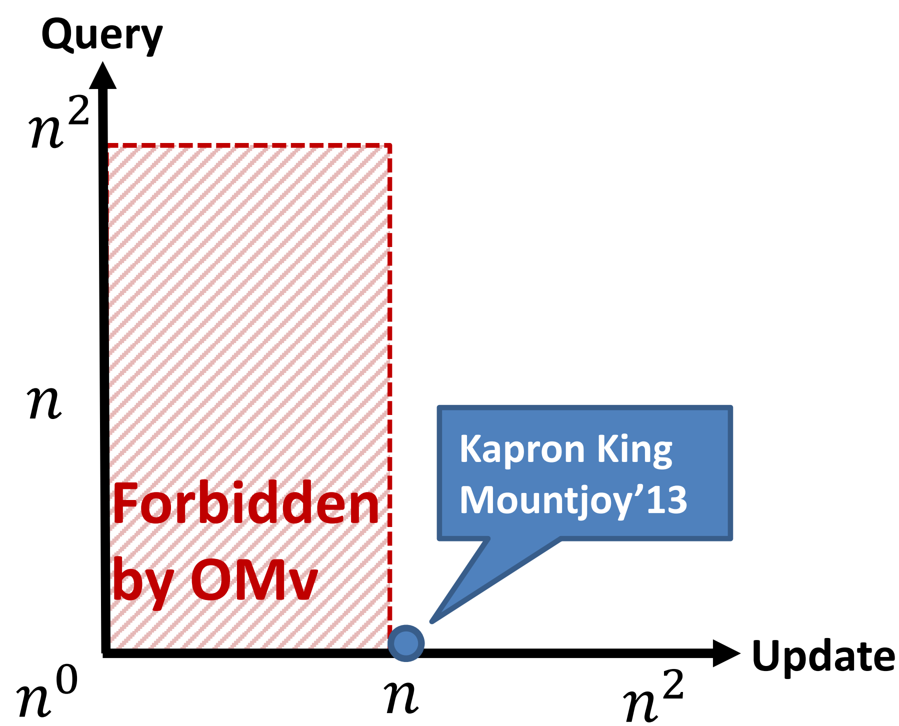


Bounds for ss-Reach via OMv

Further notes

- **OMv** (in fact $\boldsymbol{\gamma}$**-OuMv**) gives tight lower bounds of query time and update-query tradeoffs for many problems





Open: Close bounds for

Subgraph Connectivity via OMv

Part 2

Other Conjectures?

As an algorithm designer, I’m not sure I should give up when I see lower bounds from other conjectures.

But they sometimes guide to good directions.

Example: st-Reach

- Bounds hold only for small preprocessing time
- Time smaller than OMv
- Bounds from BMM is only for **"combinatorial"** algorithms
  - They were broken by algorithms based on fast matrix multiplication

Prepro	update	query	Conj
$m^{4/3}$	$m^{\delta-\epsilon}$	$m^{2/3-\delta-\epsilon}$	3SUM
$\mathbf{m^{1+\delta-\epsilon}}$	$\mathbf{m^{2\delta-\epsilon}}$	$\mathbf{m^{2\delta-\epsilon}}$ (*)	Triangle
$\mathbf{n^{3-\epsilon}}$	$\mathbf{n^{2-\epsilon}}$	$\mathbf{n^{2-\epsilon}}$ (*)	BMM
$\text{poly}(\mathbf{n})$	$\mathbf{m^{1/2-\epsilon}}$	$\mathbf{m^{1-\epsilon}}$	OMv

Should I make new conjectures?

Our own study case: **st-reach**

- After failing to further improve our algebraic algorithms for st-reach and related problems. We made three conjectures. One of them:

> **<u>v-hinted OMv (informal)</u>**
> <u>Input</u>: **Phase 1**: Boolean matrix **M**, **Phase 2:** a Boolean matrix **V, Phase 3:** index i.
> <u>Output</u> the matrix-vector product $\boldsymbol{MV_i}$, where V_i is the i-th column of **V** .
> **Naïve algorithm:** Compute MV in phase 2 or MV_i in phase 3.
> **Conjecture:** Nothing better than the naive algorithm.

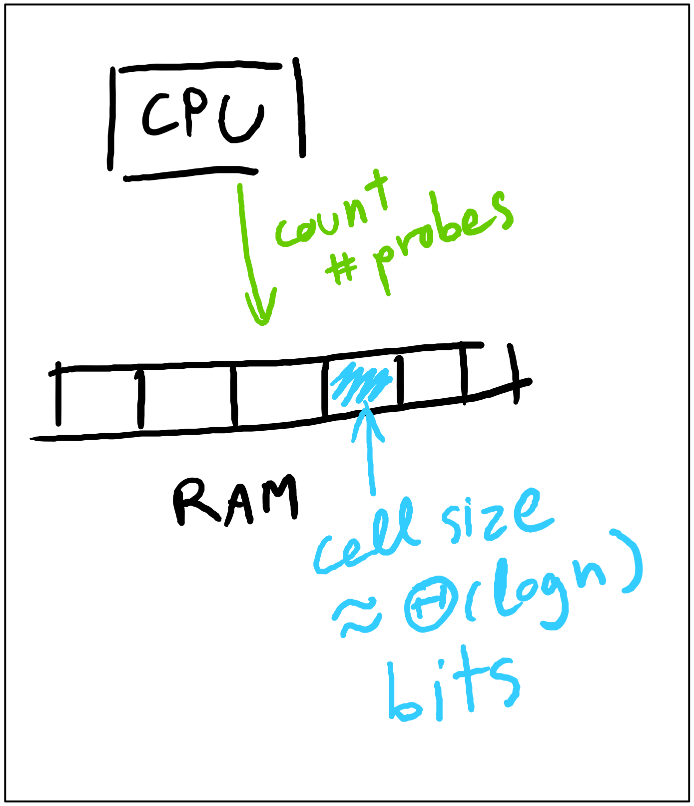
- The three together give tight lower bounds for $\approx$ 20 problems, including st-reach.

Part 3

Unconditional Lower Bounds?

Typical Model: Cell-Probe

Disclaimer: I'm not an expert



<u>Conjectures are sometimes attempted in the cell-probe model.</u>

<u>Examples:</u>

- [Cl-Gr-L'15]: Cell probe lower bounds for **OMv** problem over **very large finite fields F**, space usage **$S=\min(n \log|F|, n^2)$** when $|F|=n^{\Omega(1)}$, $S=O(n)$.
  - This does not imply the OMv Conjecture (need the Boolean case).
- [Larsen-Williams'17]: The OMv conjecture cannot be true on the cell-probe model.

Patrascu's multiphase problem and communication model

Multiphase Problem: Three phases of inputs

- Phase 1: $\boldsymbol{n} \times \boldsymbol{n}$ Boolean matrix $\boldsymbol{M}$
- Phase 2: Vector **v**
- Phase 3: Integer $\boldsymbol{i}$

Output: $(Mv)_i$

Naïve algorithm: Compute Mv in phase 2 ($O(n^2)$ time) or $M_i v$ in Phase 3 ($O(n)$ time)

Observe: OMv implies that the native algorithm is best.

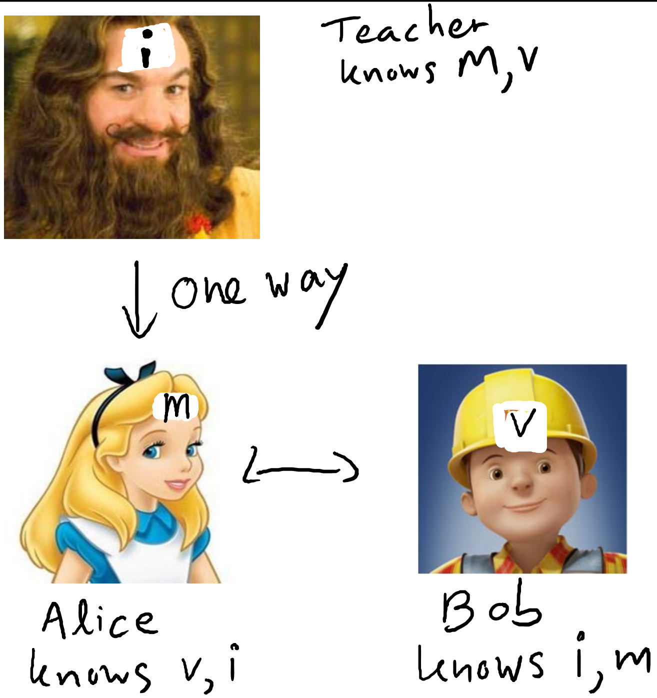
Weaker lower bounds can be derived from, e.g., **3SUM**

Patrascu's multiphase problem and communication model

- Phase 1: $\boldsymbol{n} \times \boldsymbol{n}$ Boolean matrix $\boldsymbol{M}$
- Phase 2: Vector **v**
- Phase 3: Integer $\boldsymbol{i}$

Output: $(Mv)_i$

Enough to show lower bounds for communication with Advice



Want: $(Mv)_i$

Naïve: Teacher sends M ($O(n^2)$) or Alice sends v ($O(n)$)

Claim:

- If exists algorithm A with $O(n^{1.9})$ & $O(n^{0.9})$ time in Phases 2 & 3,
- Then exists protocol where teacher sends $O(n^{1.9})$ bits and Alice and Bob exchanges $O(n^{0.9})$ bits.

Proof:

- Teacher sends what CPU wrote on memory in Phase 2 to Alice. [$O(n^{1.9})$ bits]
- Alice simulate Phase 3, and ask Bob for some missing bits (written in Phase 1). [$O(n^{0.9})$ bits]

Part 4

Partially-Dynamic Algorithms

Notes

- Partially dynamic means insertions-only or deletions-only
- Instead of **amortized update time**, we can analyze **total update time** instead.
- We have see:
  - **Incremental connectivity** with **O(log n)** worst-case update time.
  - **Incremental single-source reachability** with **O(m)** total update time (O(1) amortized).

Motivation

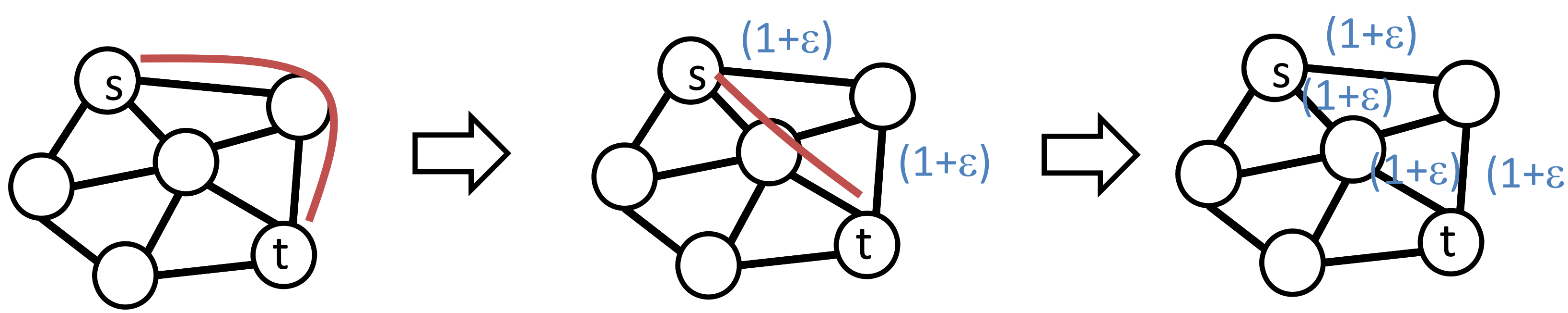
- **Enough for some data**: social networks rarely have deletions ("unfriend")
- Sometimes **equivalent** to fully-dynamic case
  - E.g. fully-dynamic connectivity is equivalent to the deletion-only one
- Enough as a subroutine for some problems

Decremental All-Pairs Shortest Paths [Roditty-Zwick FOCS'04]	Approx. multi-commodity flow [Madry STOC'10]
Decremental SSSP [HKN FOCS'14, ?]	Approx. s-t flow
Decremental min-cut (restricted)	Interval Packing, Traveling salesperson [Chekuri-Quanrud SODA'17, FOCS'17]

Example: Dyn. Shortest Paths → Max Flow

Garg-Konemann [FOCS'98], Madry [STOC'10]:
Deterministic $m^{1+o(1)}$ total update time for weighted (1+ε)-approx decremental st-shortest path → $m^{1+o(1)}$-time (1+ε)-approx max flow

Randomized algorithm against adaptive adversary is also enough.



Known: Randomized $m^{1+o(1)}$ total update time
[HenzingerKN. FOCS'14]

Example 1:

st-Distance under insertions

(It is possible to prove tight total update time!)

Thanks Thatchaphol Saranurak for slides

Theorem [Even-Shiloach JACM'81, Dinitz'71]

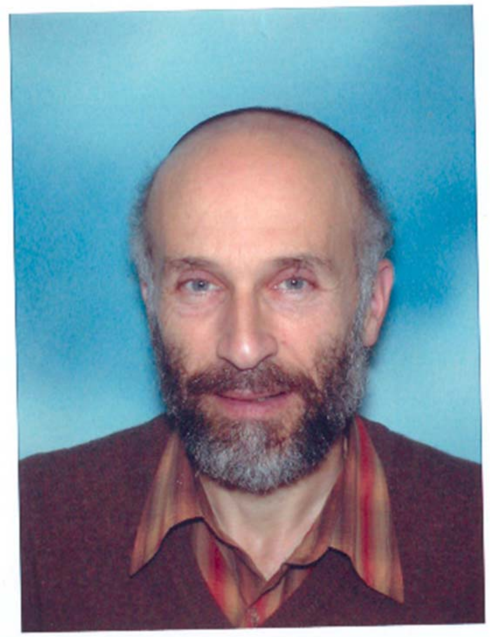
A BFS tree can be maintained with O(mn) total time for m edge insertions.

(Thus O(n) amortized over m insertions)

Even-Shiloach [JACM'81]

Well-known as Even-Shiloach Tree
(ES-tree)

Dinitz [Voprosy Kibernetiki'71]



Original version of Dinitz's maxflow algorithm

For detail, see "Dinitz' Algorithm: The Original Version and Even's Version"

Description of Even-Shiloach tree

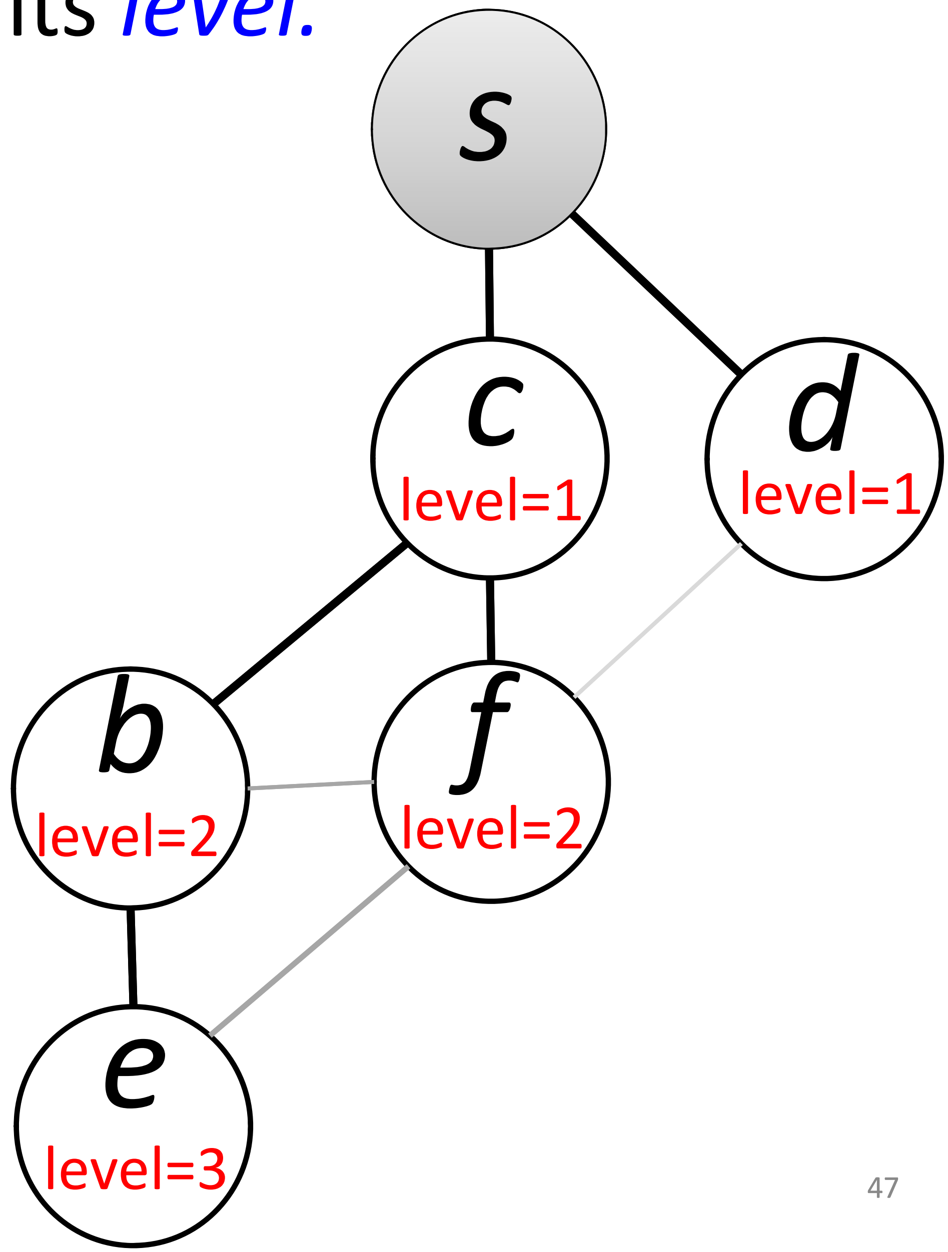
as nodes talking to each other

Optional

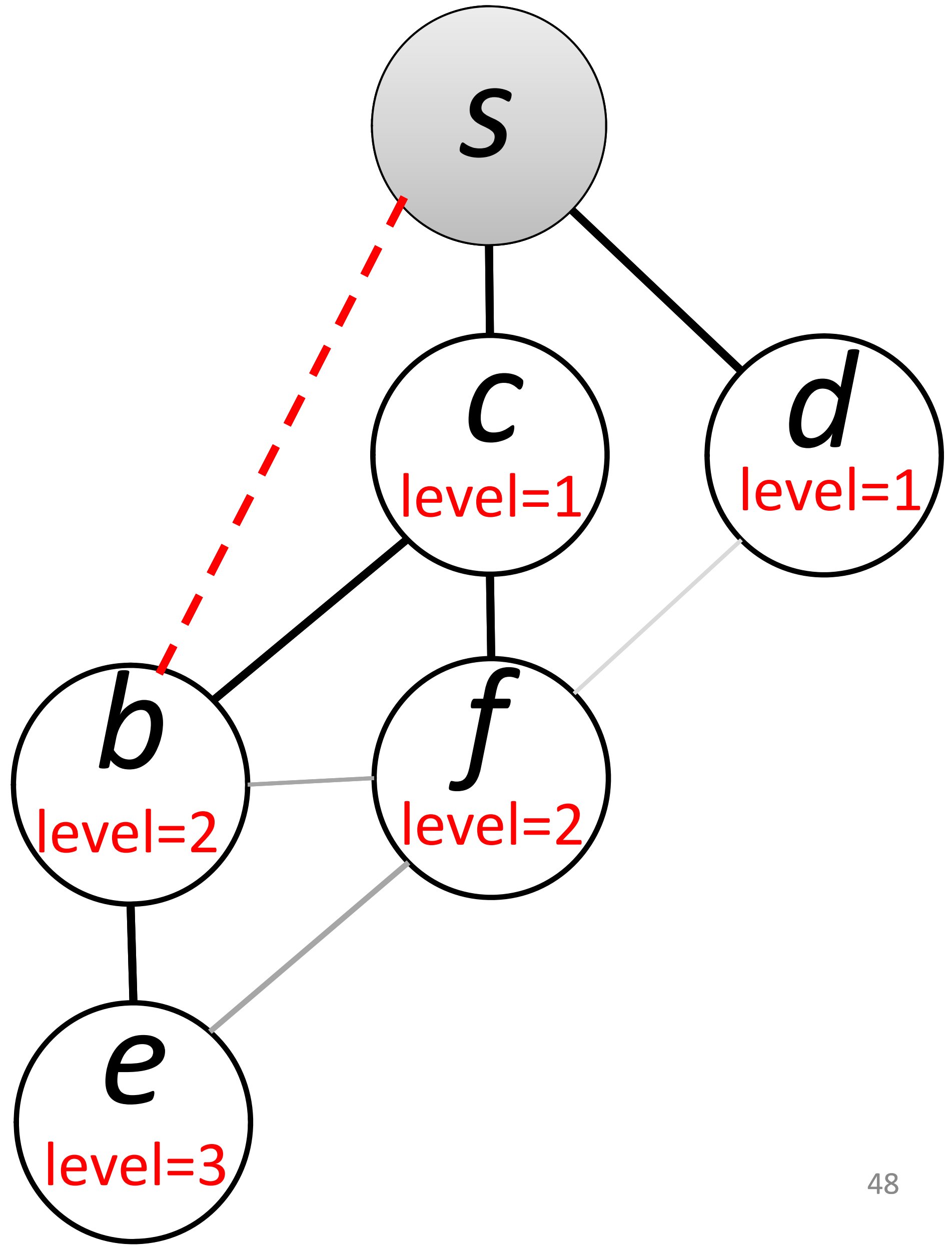
\begin{technical}

Compute BFS tree from *s*.

Every node maintains its *level.*

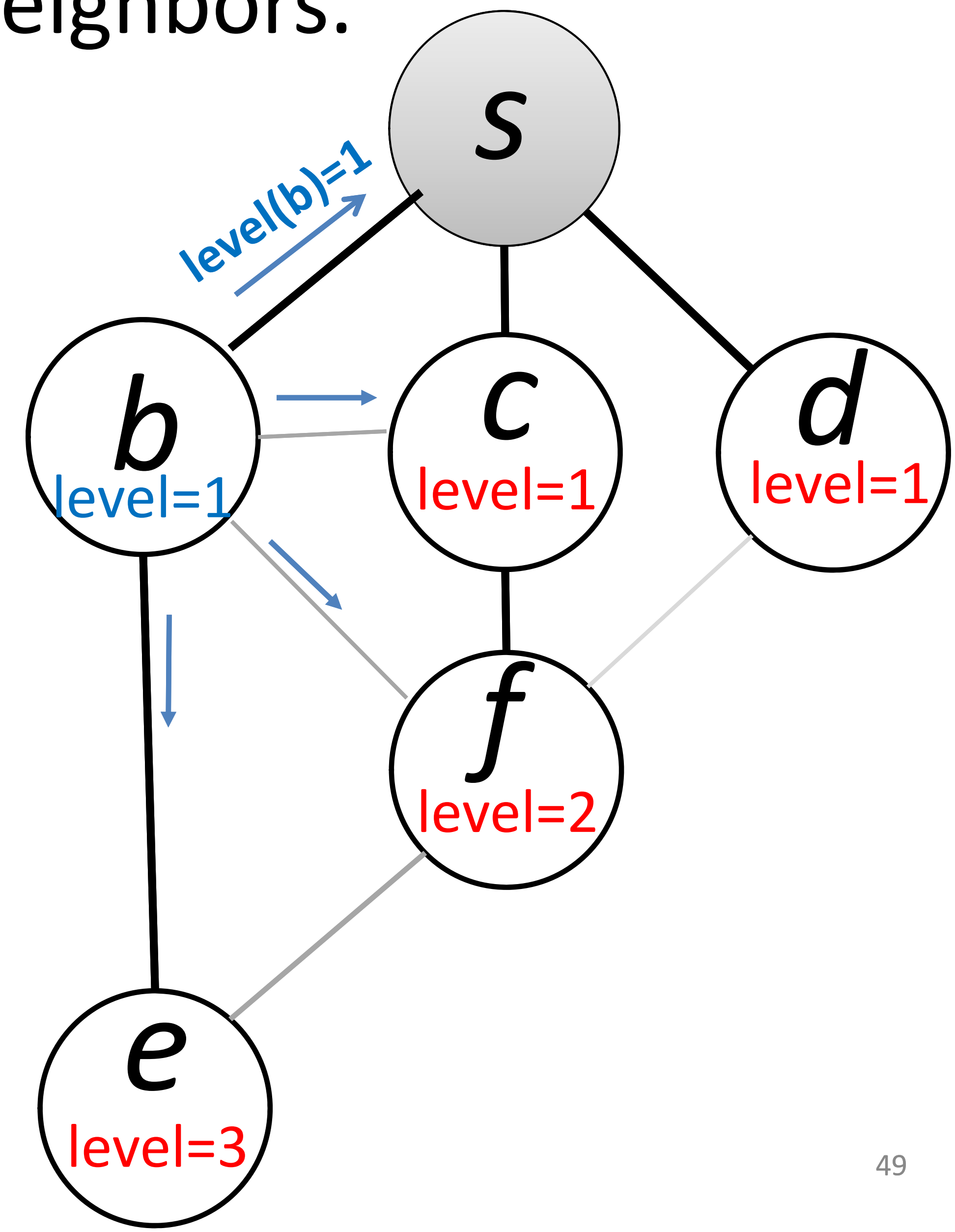


Add edge (s, b) → s and b check if their levels should change

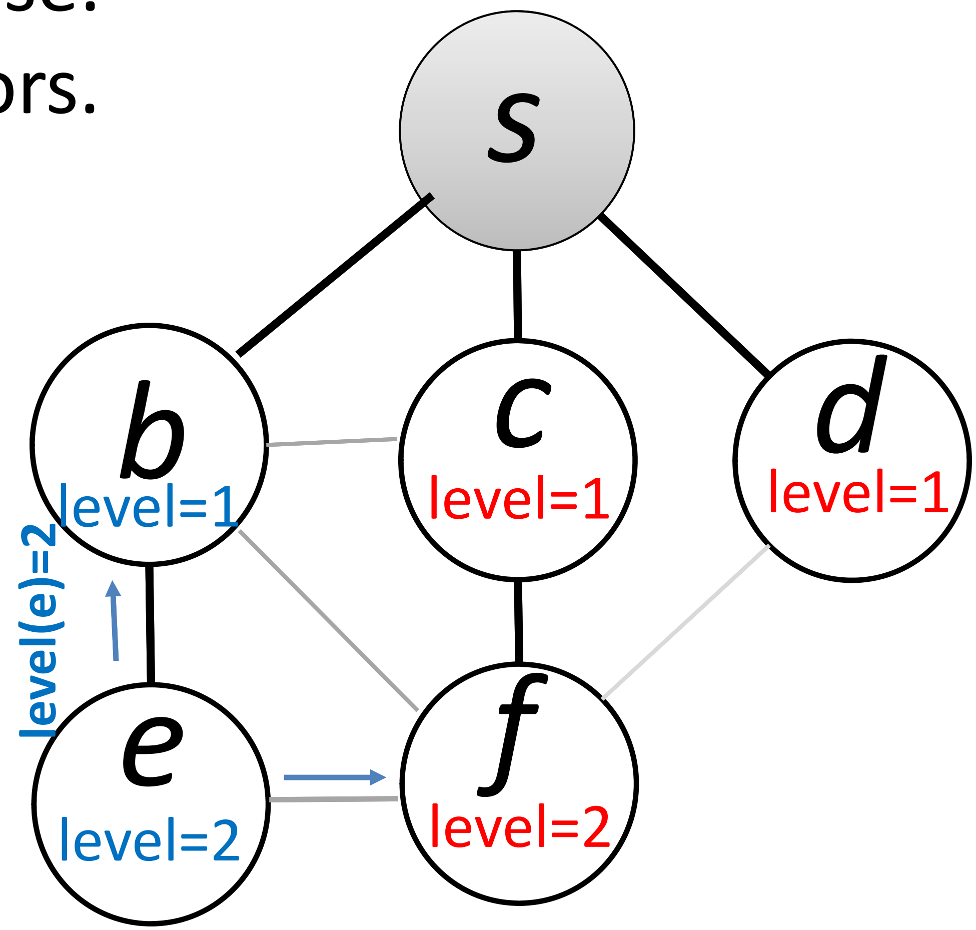


b changes its level.

It informs this to all neighbors.



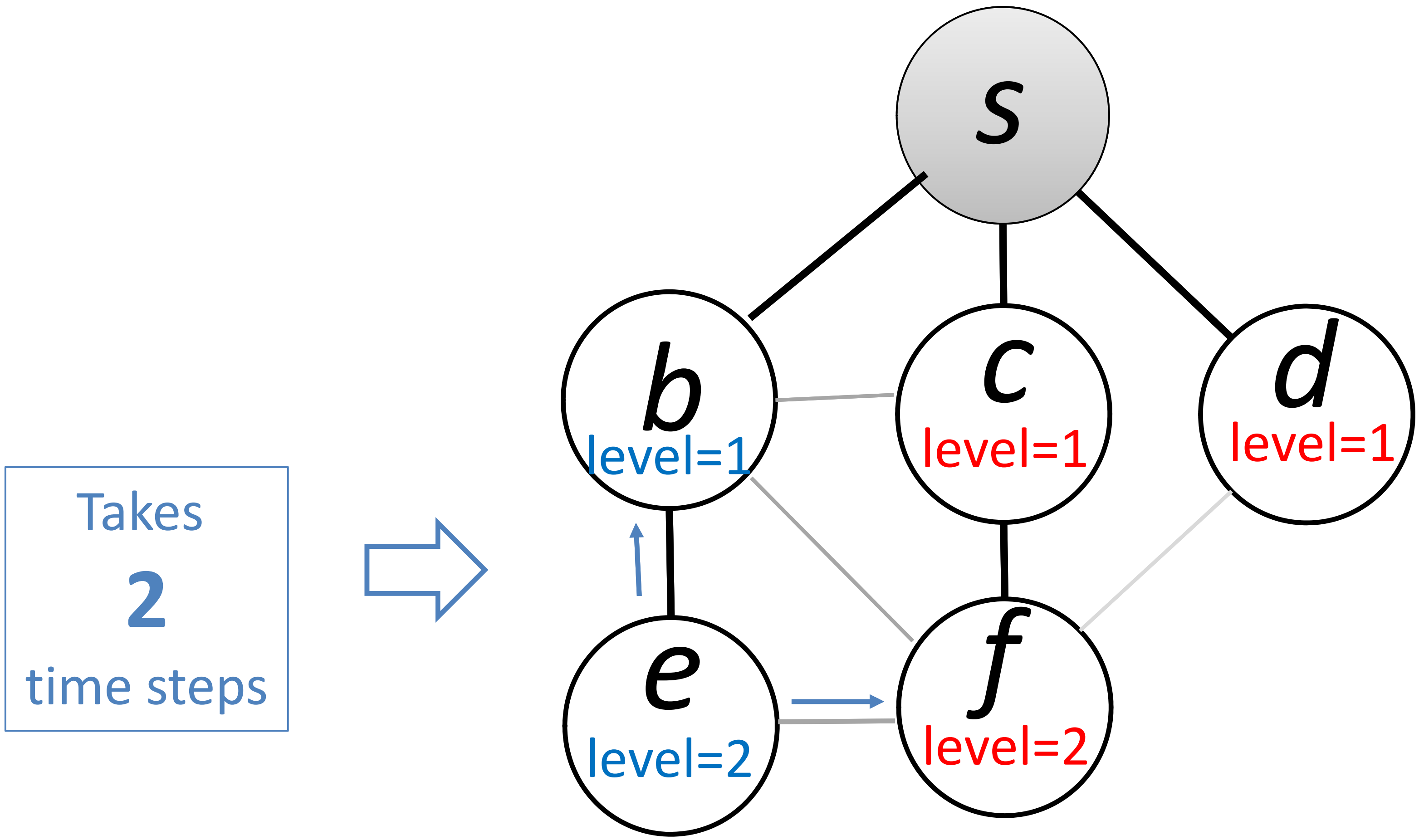
Neighbors check if they should change levels.
Node e should in this case.
Again e informs neighbors.



This is what we obtain after adding (s,b)

Even-Shiloach tree can be implemented in such a way that

total update time = number of messages



Exercise

Number of messages
(thus time complexity)
after m insertions is

O(mn)

Hint

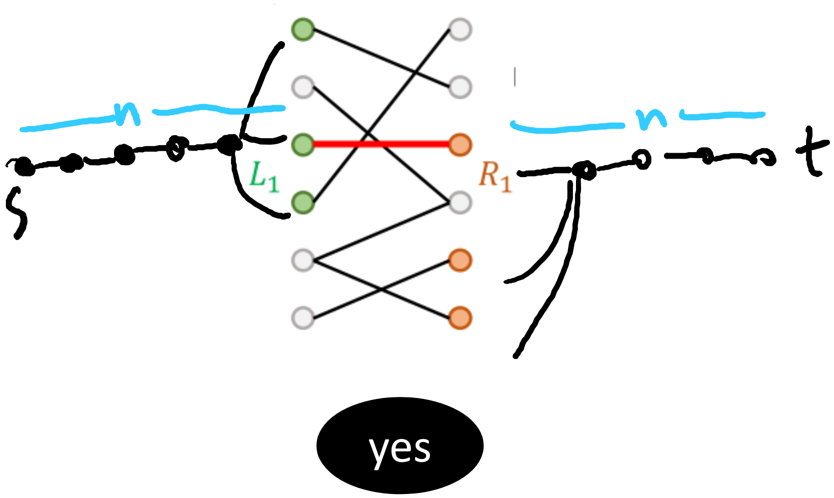
Node v sends degree(v) messages every time level(v) *decreases*.

\end{technical}

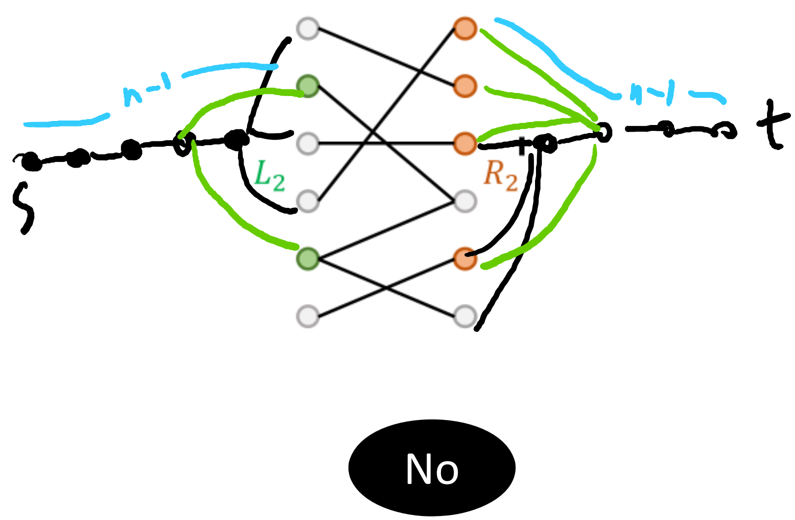
Tight Lower Bound

Lemma: st-distance cannot have total update time $O((mn)^{1-\epsilon})$, assuming the OMv conjecture.

Proof sketch:



dist(s,t)=2n+1 iff "yes"



dist(s,t)=2(n-1)+1 iff "yes"

Example 2

st-Reach under insertions

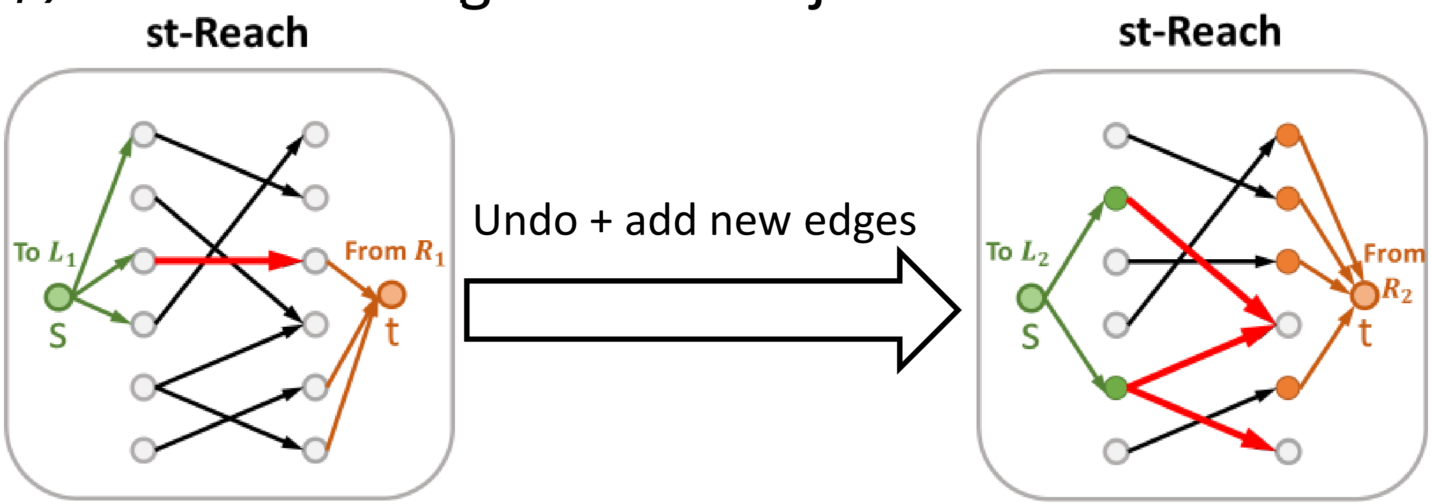
Thanks Thatchaphol Saranurak for slides

This example shows …

- Converting **amortized** fully-dynamic lower to **worst-case (only!)** for partially-dynamic lower bounds.
- It works for most problems.

Claim: Incremental st-Reach has $\mathbf{\Omega(n)}$ **worst-case** lower bound

- **Trick:** *Undo (roll-back)* insertions before new insertions
- **Worst-case** update time $\boldsymbol{O(n^{0.9})}$ → $O(\boldsymbol{n^{1.9}})$ time per (L_i, R_i). Contradicting OuMv conj.



Doesn't work for total update time: If assume, say, $\boldsymbol{O(n^2)}$ total update time, we may spend $\boldsymbol{O(mn^{1-\epsilon})}$ time per (L_i, R_i). Nothing to contradict.

Questions?

Acknowledgements:
Sayan Bhattacharya, Jan van den Brand, Deeparnab Chakraborty, Sebastian Forster, Monika Henzinger, Christian Wulff-Nilsen, Thatchaphol Saranurak

This project has received funding from the *European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme* under grant agreement No 715672