

Chapter 1.

Introduction to Dynamic Graph Algorithms

Danupon Nanongkai

KTH, Sweden

About This Lecture

What (some) dynamic graph algorithm **designers** want.

Plan

1. Dynamic algorithms & update time
2. Example: Dynamic connectivity
3. Intermediate questions, amortization, randomization
4. The story of connectivity

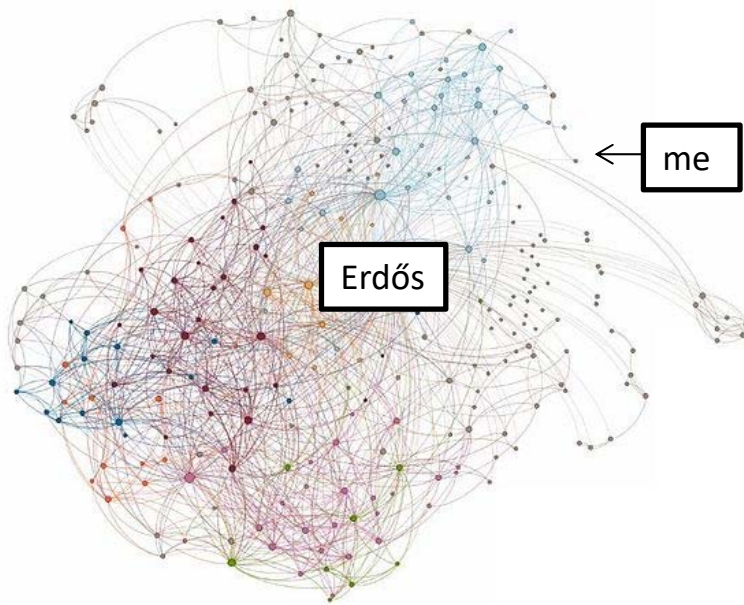
Disclaimer: This session is mostly about upper bounds

Please feel free to ask questions at any time!

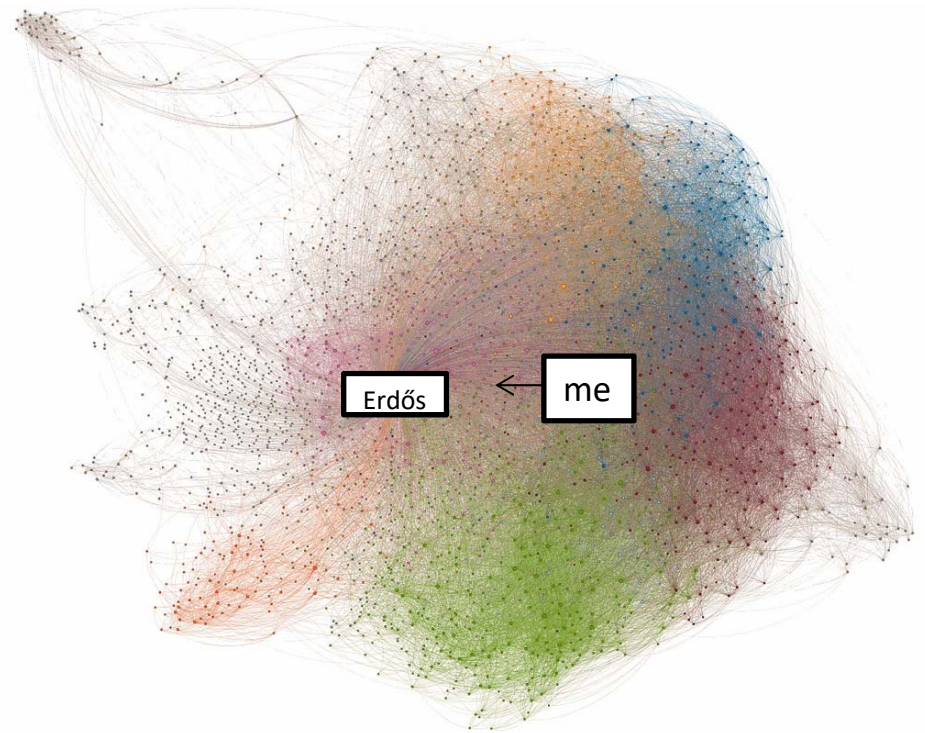
Part 1

DYNAMIC ALGORITHMS & UPDATE TIME

Example 1: What's my Erdős number? (How far am I from Erdős?)

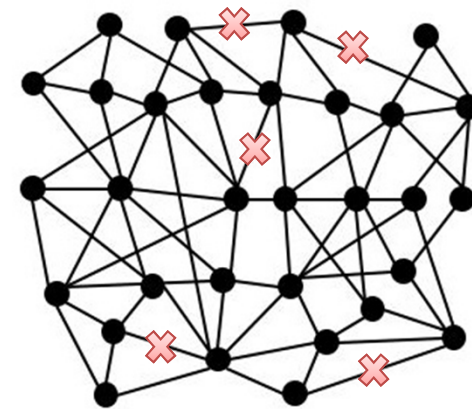
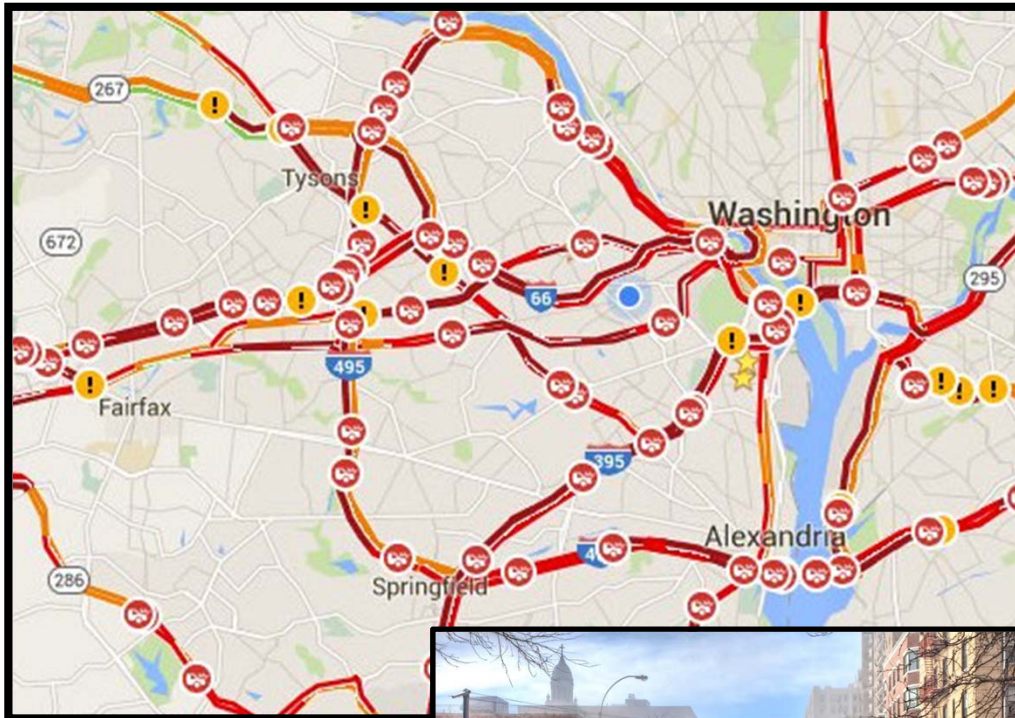


collaboration network 10 years ago



collaboration network today

Example 2: Fastest driving route



Dynamic graph problems

Want: **maintain** some graph properties

Distance, connectivity, MST, Maximum matching, etc.

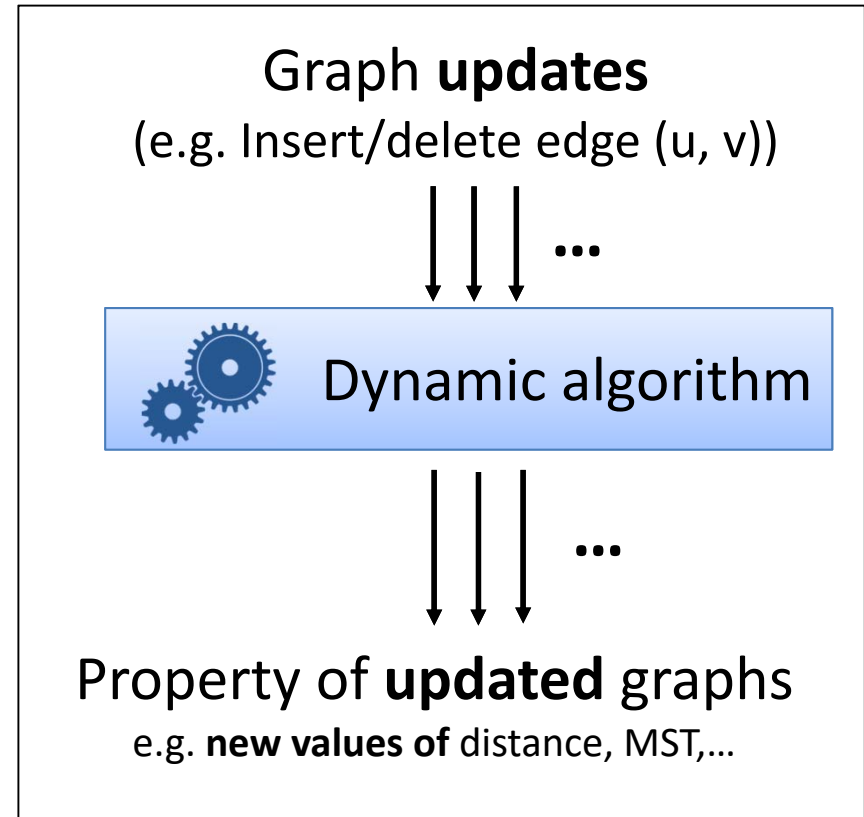
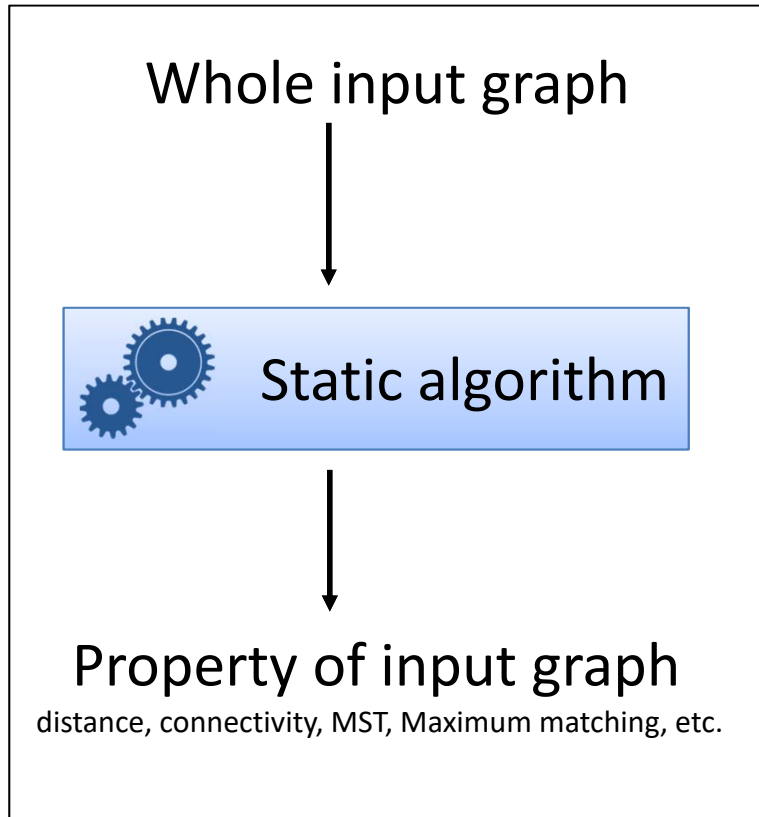
Challenge: Graph **changes** over time

Edge insertions, deletions, weight changes, etc.

Goal: Fast algorithms

Minimize **update time** = time to process each update

Static vs. Dynamic Graph Algorithms



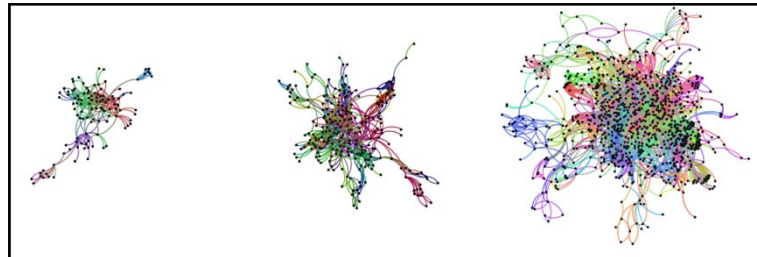
Running time: Time to return output

Update time: Time to return output **after each update**

Remark: Query operation/time is omitted for simplicity

Why Dynamic Algorithms?

Evolving Networks Analysis:



Subroutine for Static Algorithms:

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
Inverse [SL FOCS'15]	Interior point method [SL FOCS'15]
Connectivity on trees (link-cut tree) in $\tilde{O}(1)$	Directed max flow [Goldberg Tarjan STOC'88]
Minimum spanning tree in $\tilde{O}(1)$	$(1 + \epsilon)$ -approx Global min cut, tree packing [Thorup STOC'01, Karger-Thorup SWAT'00]
2-edge connectivity in $\tilde{O}(1)$	Unique perfect matching [Gabow Kaplan Tarjan STOC'99]
Decremental min-cut (restricted)	Interval Packing, Traveling salesperson [Chekuri-Quanrud SODA'17, FOCS'17]

Part 2

EXAMPLE: DYNAMIC CONNECTIVITY

Notations



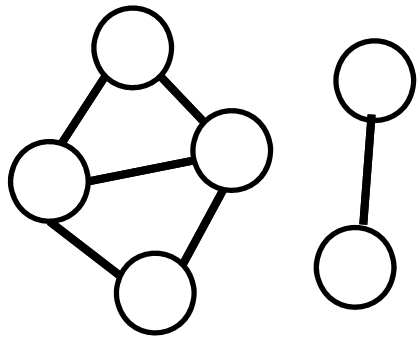
- n = number of nodes
- m = number of edges
- $\text{polylog}(n)$ is mostly hidden

Example

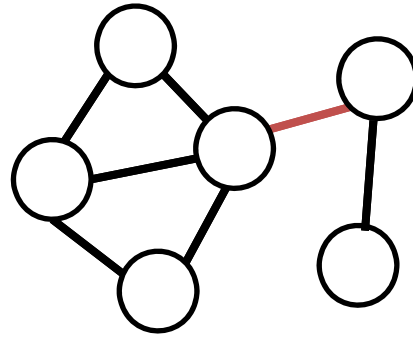
Dynamic Connectivity

Each Update: An edge insertion/deletion.

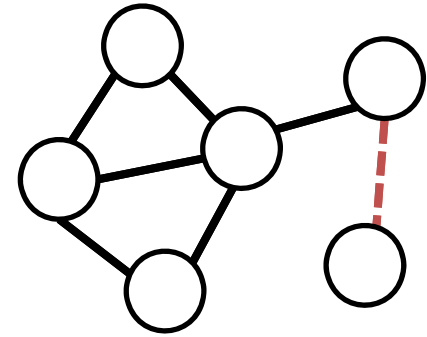
Maintain: Is the graph connected?



Output = "not connected"



Output = "connected"



Output = "not connected"

Naïve algorithms

Solve **from scratch** every time

(e.g. Breath-First Search)

Update time = $O(m)$

?

Happy with linear update time?

?

Do we need to read the whole data every time?

NO

Ultimate goal

What's the **best update time** for basic graph problems under edge updates?

(st-reach, connectivity, distances, min-cut, max-matching, max-flow etc.)

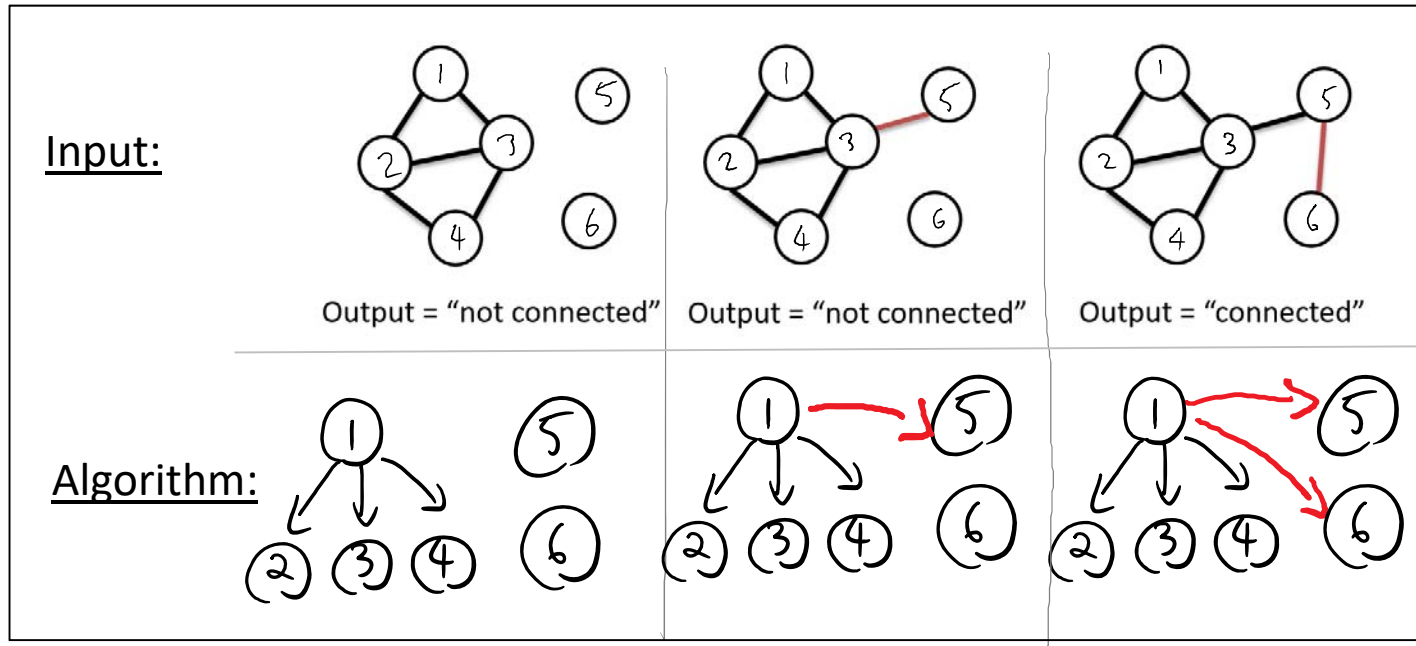
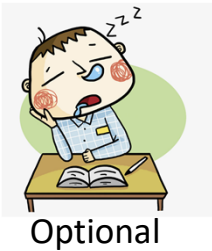
Remarks Other settings (not focused today):

- Queries, e.g. “distance between two nodes = ?”
- Other types of updates, e.g. node updates

Example

Under **only edge insertions**, we can maintain connectivity in **$O(\log n)$** update time

Example: Connectivity, only insertions



Algorithm:

- Maintain a **tree** for each connected component
- ⚠ Merge components: make **small** tree a subtree

Analysis: $O(\log n)$ update time

- Tree depth = $O(\log n)$
- Node's distance to root **increases** \rightarrow component size **doubles**

Theorem

Under **only edge insertions**, we can maintain connectivity in **$O(\log n)$** update time

?

How about edge deletions only & deletions+insertions?

?

How about maintaining **distance** between a pair (Erdős number)?

?

How about higher edge connectivity, spanning tree, minimum spanning tree?

Can you guess η ? η ? η ?

What's the *right* update time for following problems under edge insertions/deletions?

i) $O(m)$ ii) $O(\text{polylog } n)$ iii) something else

Answer: It's complicated ...

1. st-connectivity

Exists undirected st-path?

Randomized or amortized (but not both): $\text{Polylog}(n)$

2. st-reachability

Exists directed st-path?

Amortized: $\Omega(m^{1/2-o(1)})$

Worst-case, randomized: May be $\Theta(n^{1.407})$

3. st-distance

Output length of shortest (directed, weighted) st-path.

Amortized: $\Omega(m^{1/2})$

Worst-case, randomized: $O(n^{1.724})$

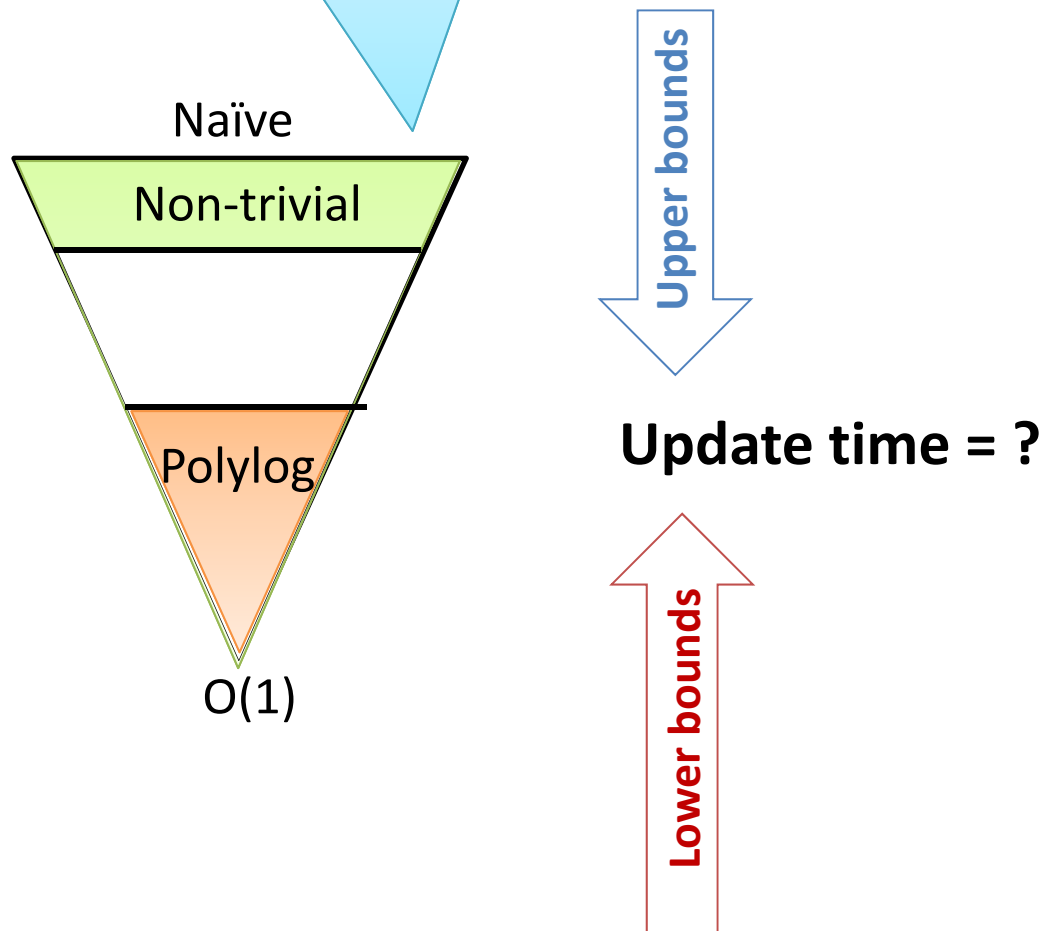
Part 3

INTERMEDIATE QUESTIONS (FOR ALGORITHM DESIGNERS)

Questions

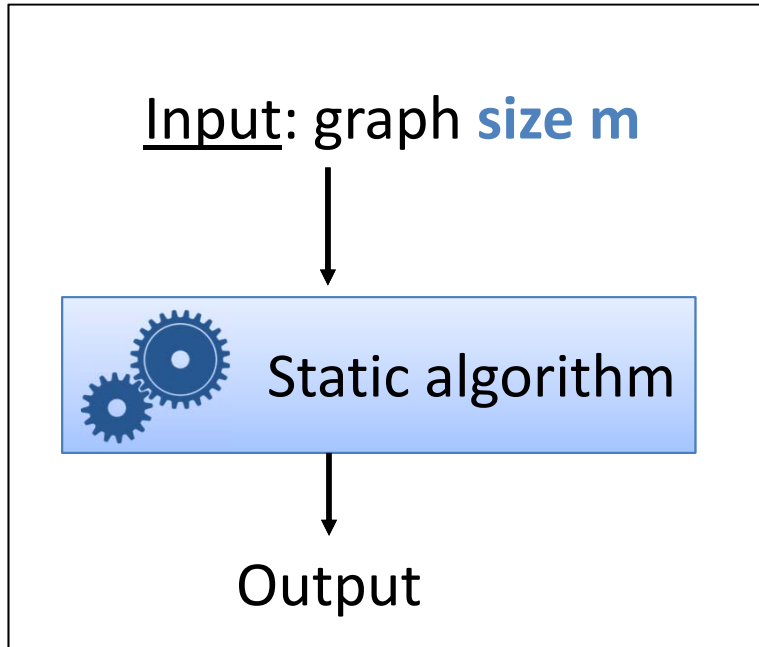
1. Tight update time = ?

2. Non-trivial time (beating static)? Polylog time?



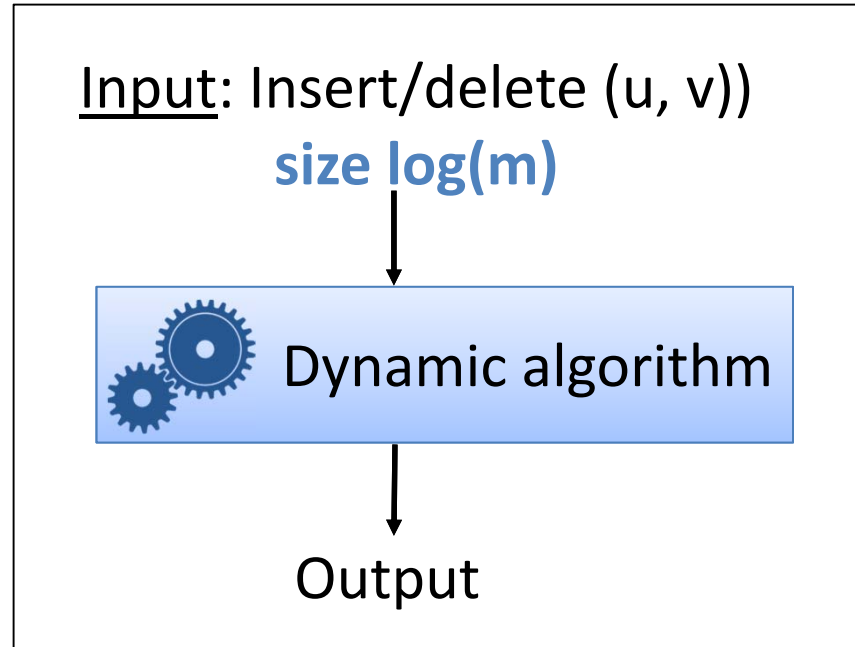
Why polylog update time?

The Class **Dynamic P**



In P?

Exist **poly**(m)-time algorithm?



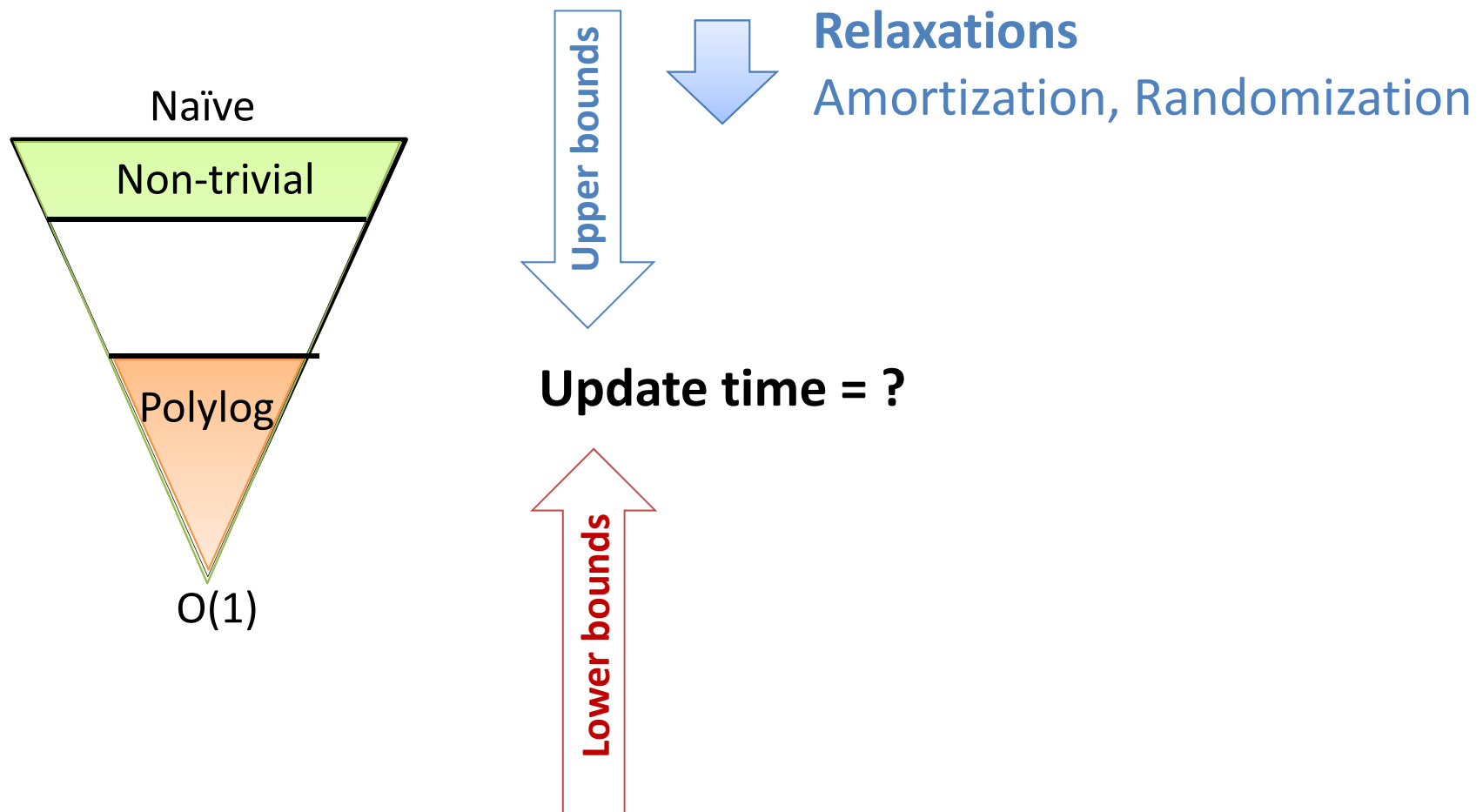
In dynamic P?

Exist **polylog**(m)-time algorithm?

Questions

1. Tight update time = ?
2. Non-trivial time (beating static)? Polylog time?

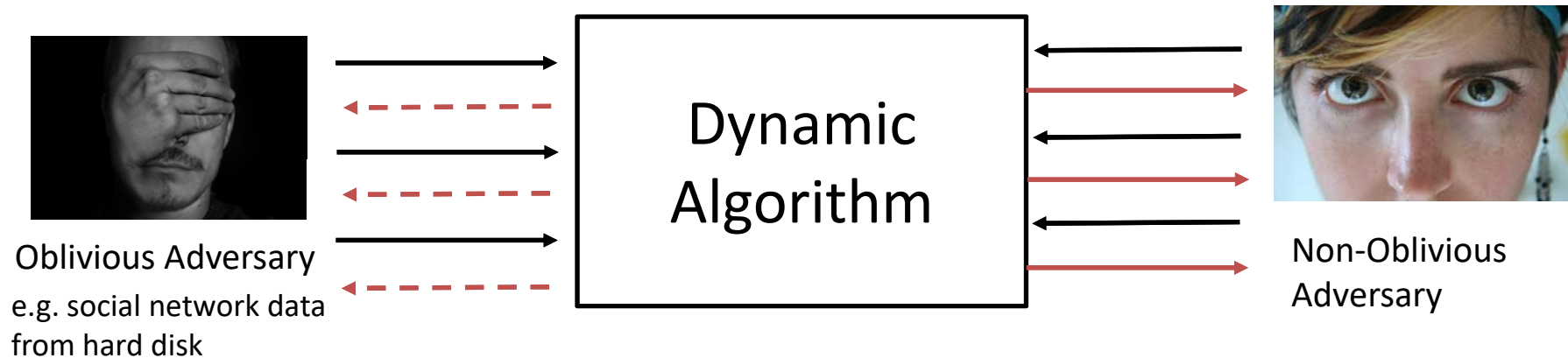
3. Answer with relaxations?
4. Remove relaxations



Randomized Dynamic Algorithms

- Las Vegas: **Expected** update time
- Monte Carlo: **Wrong** output with small probability

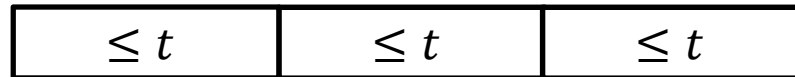
Assumption: *Oblivious* adversary. (more on this later)



Amortized Update Time

Worst case t

for **each** update
time $\leq t$



...



Amortized t

("average case")
after u updates
time $\leq ut$



...

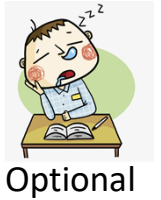


Empty-start Assumption (for graphs):

Start from empty graphs

Typical dynamic graph algorithms with amortized time

- **Initialize(n):** Create an empty n -node graph
- **Insert(u,v):** Insert edge (u,v)
- **Delete(u,v):** Delete edge (u,v)
- (optional) **Query(u,v):** Ask about the current graph.

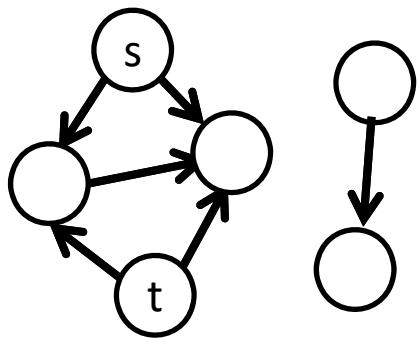


Example

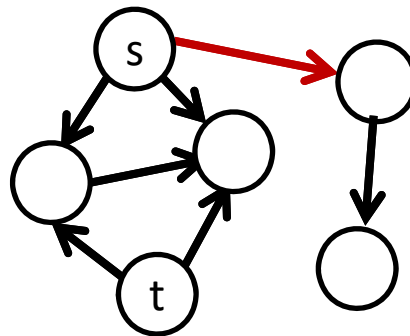
st-Reachability under insertions

Each Update: An edge insertion.

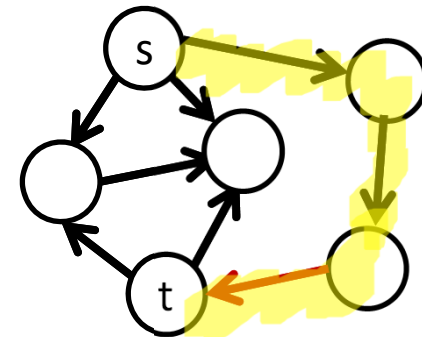
Maintain: Exists directed st-path?



Output = "no"



Output = "no"



Output = "yes"



Example

st-Reachability under insertions

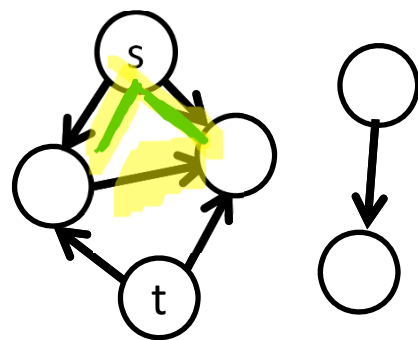
Claim: Exists algorithm with $O(m)$ time after m edge insertions

Algorithm (sketched):

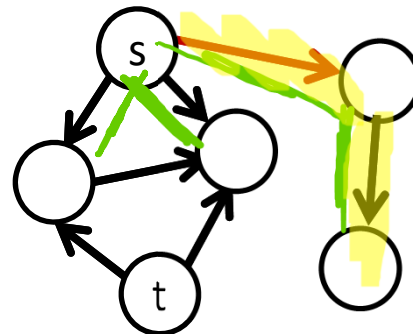
Keep track of nodes reachable from s using directed edges.

- When see new node, explore its out-going edges.

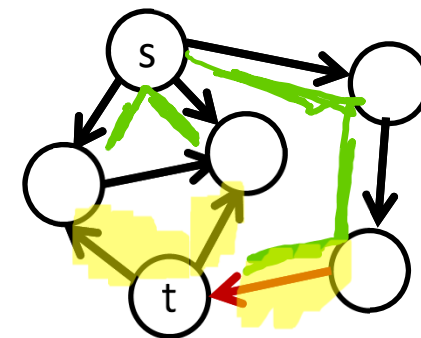
Analysis: Read each edge only once.



Output = "no"



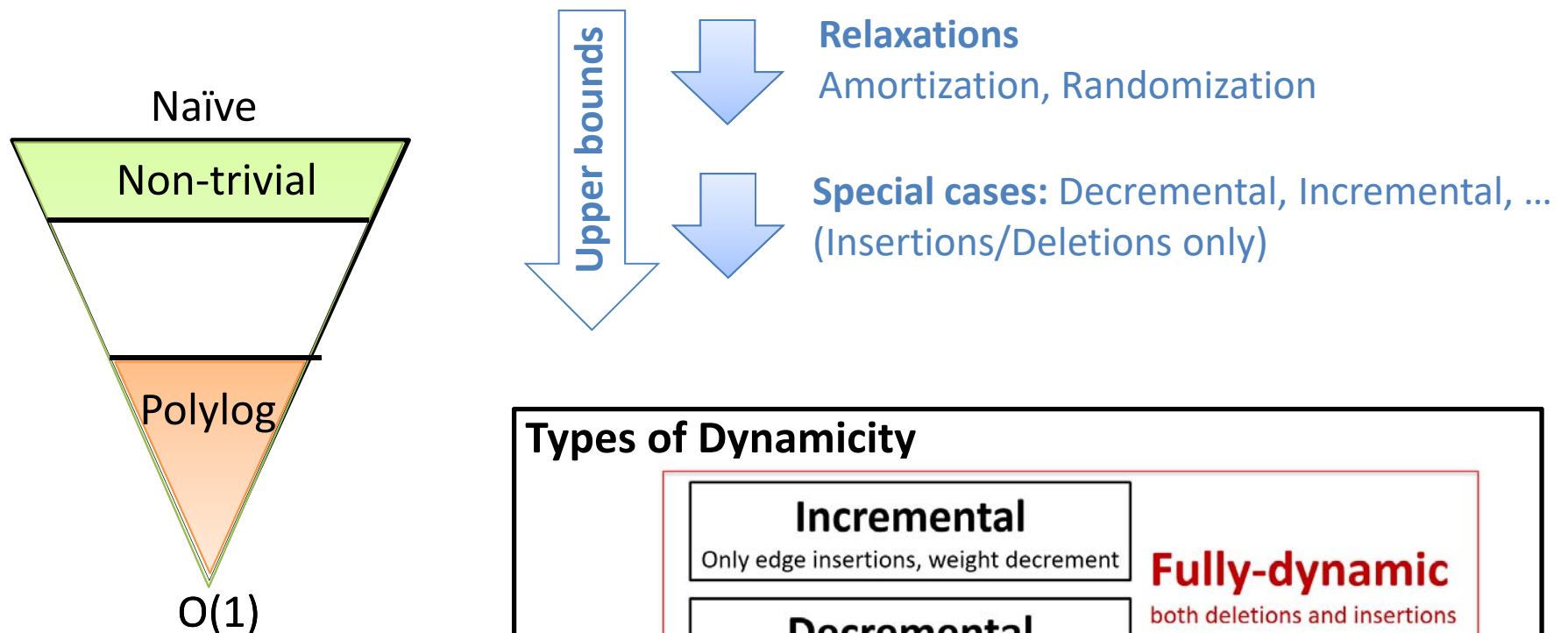
Output = "no"



Output = "yes"

Questions

1. Tight update time = ?
2. Non-trivial time (beating static)? Polylog time?
3. Answer with relaxations?
4. Remove relaxations



Types of Dynamicity

Incremental

Only edge insertions, weight decrement

Decremental

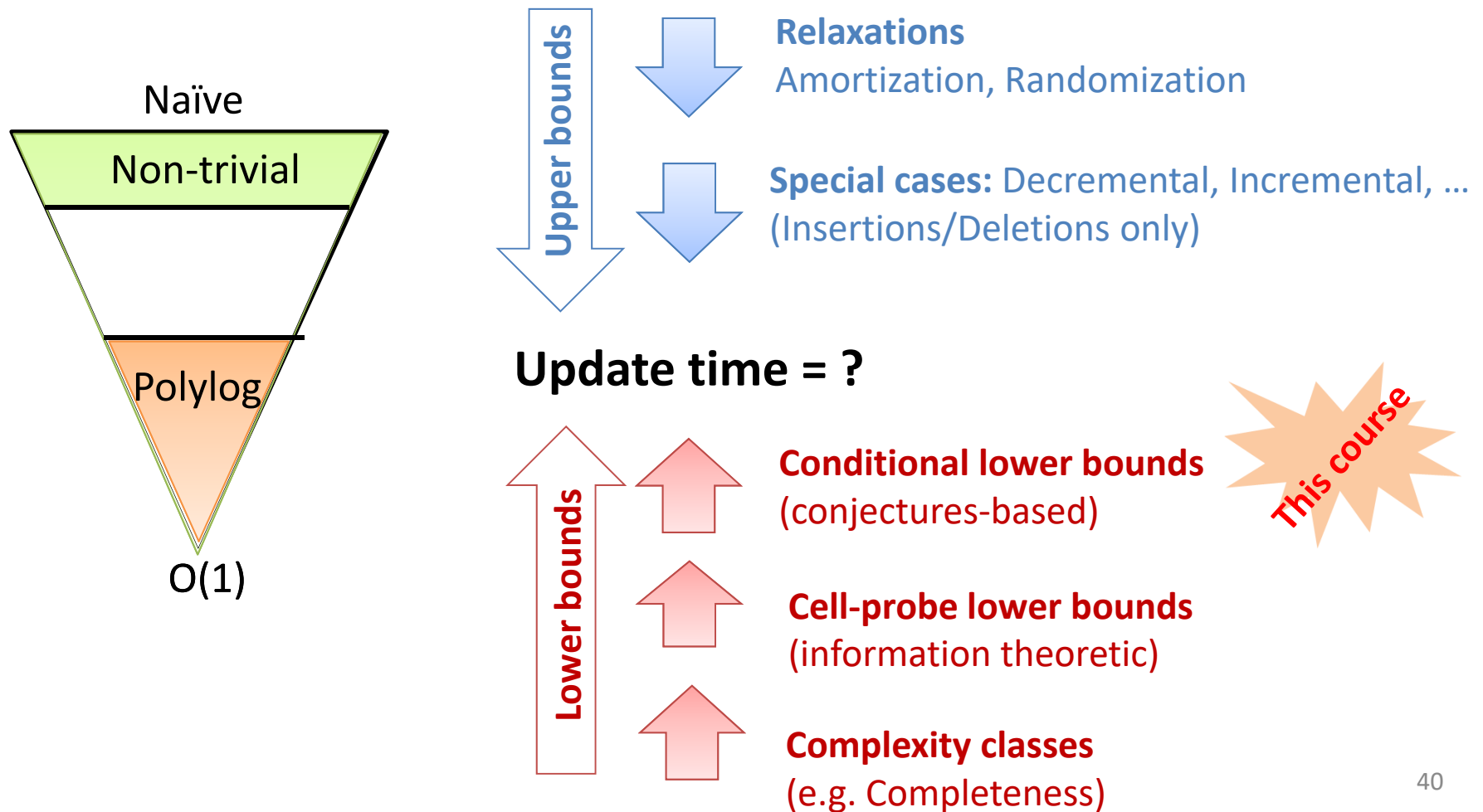
Only edge deletions, weight increment

Fully-dynamic
both deletions and insertions

Others: Emergency planning, f-sensitivity, nodes on/off, offline, etc.

Questions

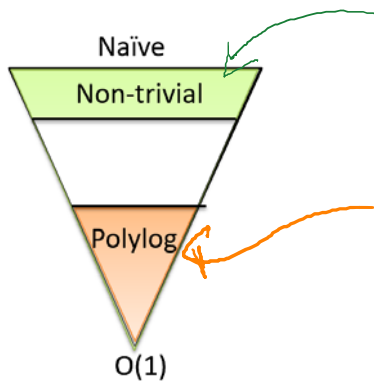
1. Tight update time = ?
2. Non-trivial time (beating static)? Polylog time?
3. Answer with relaxations?
4. Remove relaxations




Part 4

THE STORY OF DYNAMIC CONNECTIVITY

The Story of Dynamic Connectivity



Reference	Update Time	Amortized?	Random?
Naïve	m	x	x
Frederickson [STOC'83]	\sqrt{m}	x	x
EGIN [FOCS'92]	\sqrt{n}	x	x
Henzinger, King [STOC'95]	$polylog n$	✓	✓
T [STOC'00], PD [STOC'04], HHKP [SODA'17]	$\tilde{O}(\log n)$	✓	✓
HLT [STOC'98], W [SODA'13]	$polylog n$	✓	x
Kapron King Mountjoy [SODA'13] Also [GibbKKT'15]	$polylog n$	x	✓
KKPT [ESA'16]	$\sqrt{n} \cdot \frac{\log \log n}{(\log n)^{1/2}}$	x	x
 Major open problem	$polylog n$ or just $n^{\frac{1}{2}-\epsilon}$	x	x

$n = \#$ of nodes, $m = \#$ of edges

Open: Lower bounds for dynamic connectivity?

Challenges: Need technique that can distinguish between

- **Decremental vs. Incremental Algorithms**
 - Incremental connectivity is easy
 - Decremental connectivity is as hard as the fully-dynamic one [Wulff-Nilsen STOC'17]
- **Randomized vs. Deterministic Algorithms**
 - Already exists fast randomized algorithms
 - even without oblivious adversary assumption



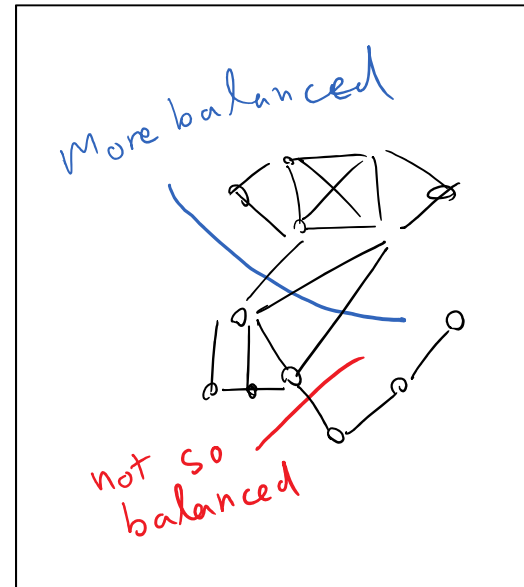
Suffice: Balanced Sparsest Cut

[N, Saranurak, Wulff-Nilsen FOCS'17]

The problem (Informally)

Want: **Sparse** cut with many nodes on both sides (“Balanced”)

Known [NS STOC'17]: **Randomized** almost-linear-time polylog-approx*



If derandomized →

Update Time	Amortized?	Random?
$n^{o(1)}$	x	x

Result holds even for dynamic MST

First barrier: Deterministically expander testing

*For further consequences of this result, see [ChuGPSSW FOCS'18].

CONCLUSION

Some Jagons

- Update Time
- Incremental/Decremental Algorithms
- Amortization
 - Also: Empty-start assumption*
- Randomization
 - Also: Oblivious-adversary assumption

* This is not a common name.

Algorithms designers' goals:

- Small update time
- Deterministic, worst-case, or at least without assumptions:
 - oblivious-adversary, and
 - empty-start.

Questions to keep in mind when prove lower bounds



Does your lower bound hold for **randomized** and **amortized** algorithms?

- If yes for amortized: Hold when start from **empty graphs**?

- If yes for randomized: Hold for **oblivious adversary**?

If all answers are yes → Algorithms designer can give up

Heads-up

Good news: “Yes” for most lower bounds we will see.

Bad news: We lack lower bound techniques to separate

- randomized vs. deterministic algorithms,
- amortized vs. worst-case bounds, and
- incremental vs. decremental algorithms.

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

