Automatically Speeding Up LOCAL Algorithms

Seth Pettie

University of Michigan

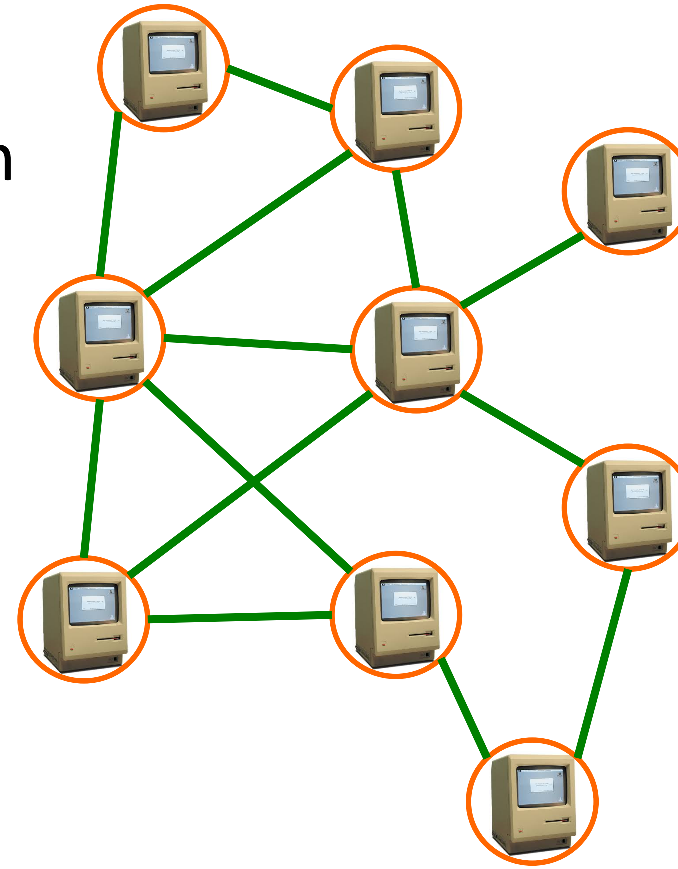
Joint work with Yi-Jun Chang

Naor, Stockmeyer, SICOMP 1995
Chang, Kopelowitz, Pettie, FOCS 2016
Chang, Pettie, FOCS 2017, SICOMP 2018

The LOCAL Model

[Linial'92]

- A graph G=(V,E)
  - Vertex = processor
  - Edge = bidirected communication
  - Time: ***synchronized*** rounds. In each round, each vertex sends a message to each neighbor.
  - ***Computation is free.***
  - ***Message size is unbounded.***
  - "Time" = number of rounds



- ***Randomized*** LOCAL
  - Can generate an ***unbounded number of random bits***

The LOCAL Model

[Linial'92]

- What a vertex $\boldsymbol{v}$ knows:
  - Global graph parameters: $\boldsymbol{n} = |V|$, $\Delta = \max_u \deg(u)$
  - A unique O(log n)-bit ***ID***(***v***).
  - A ***port-numbering*** of its deg(***v***) incident edges.

The LOCAL Time Hierarchy

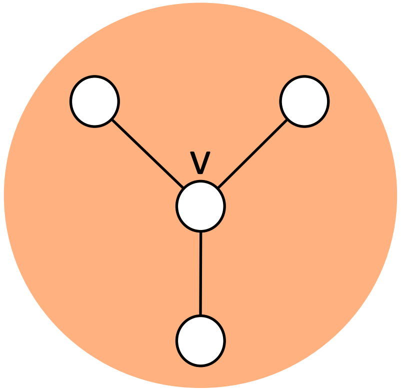
- **<u>Q:</u>** Which ***time complexities*** are obtainable by ***"natural" problems***.
  - To reduce the number of problem parameters, assume $\boldsymbol{\Delta = O(1)}$.

What is a "*natural problem*" ?

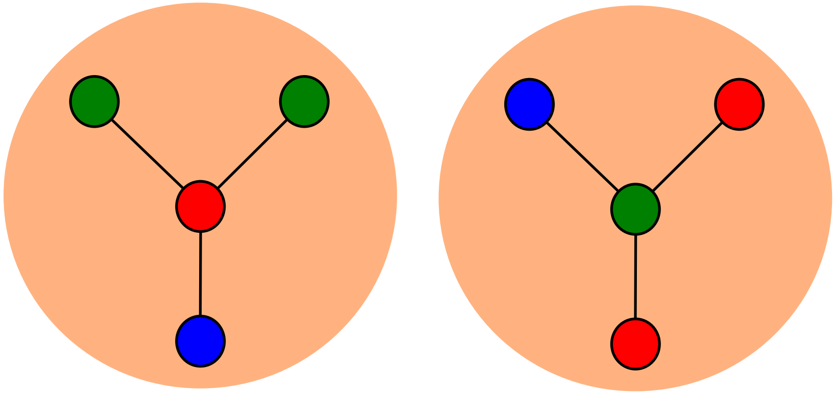
[Naor-Stockmeyer'95]

- Locally Checkable Labeling (LCL) problem: NTIME(O(1))
  - Input and Output alphabets Σ_{in}, Σ_{out}, integer radius **r**.

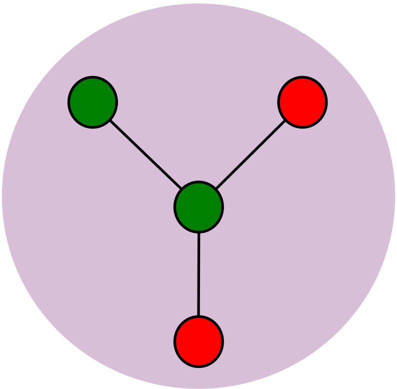
    $|\Sigma_{in}|$, $|\Sigma_{out}|$ may depend on Δ, but are independent of **n**.
  - Set **C** of acceptable radius-**r** centered subgraphs.
- Problem: given $V \rightarrow \Sigma_{in}$, compute $V \rightarrow \Sigma_{out}$ such that every vertex's radius-**r** view is in **C**.



radius-1 view from v



some acceptable configurations for 3-coloring.



an unacceptable configuration

Greedy vs. Nongreedy LCL Problems

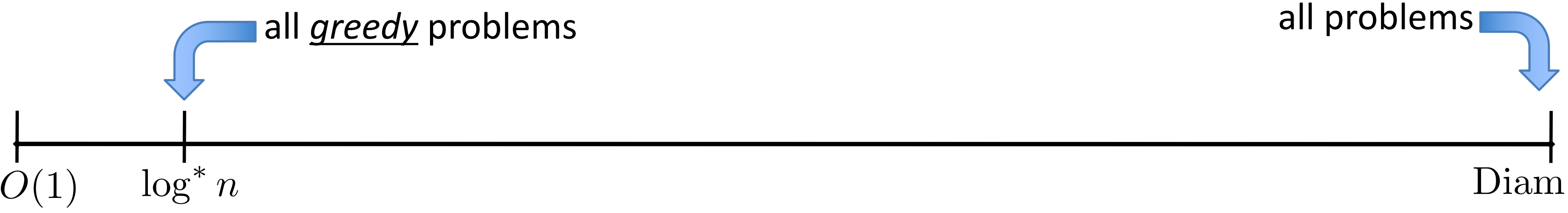
- The canonical ***greedy*** problems:
  - Maximal independent set
  - Maximal matching
  - $(\Delta+1)$-vertex coloring
  - $(2\Delta-1)$-edge coloring

every partial solution extends
to a total solution

- Some ***non-greedy*** problems
  - (approximate) maximum matching
  - Sinkless orientation
  - Δ-vertex coloring
  - $(2\Delta\text{-}2)$-edge coloring
  - Frugal coloring
  - Defective coloring
  - Versions of List coloring, etc.

Time Hierarchies: Δ=O(1)

1. O(Δ^2)-color the graph in $\log^* n$ time. [Linial's algorithm '92]
2. Apply greedy algorithm to each color class, one at a time.



Time Hierarchies: Δ=O(1)

n-path/cycle, $(\sqrt{n} \times \sqrt{n})$-grid/torus

Naor, Stockmeyer'95

via hypergraph Ramsey argument…
any O(1) time algorithm can be made
order-invariant w.r.t. vertex IDs.

In O(log* n) time, we can make all vertices
think they are in an O(1)-size path/cycle/grid/torus.

Chang, Pettie'17

Brandt, Hirvonen, Korhonen, Lempiäinen, Östergård, Purcell, Rybicki, Suomela'17

Chang, Kopelowitz, Pettie'16

Brandt, Hirvonen, Korhonen, Lempiäinen, Östergård, Purcell, Rybicki, Suomela'17

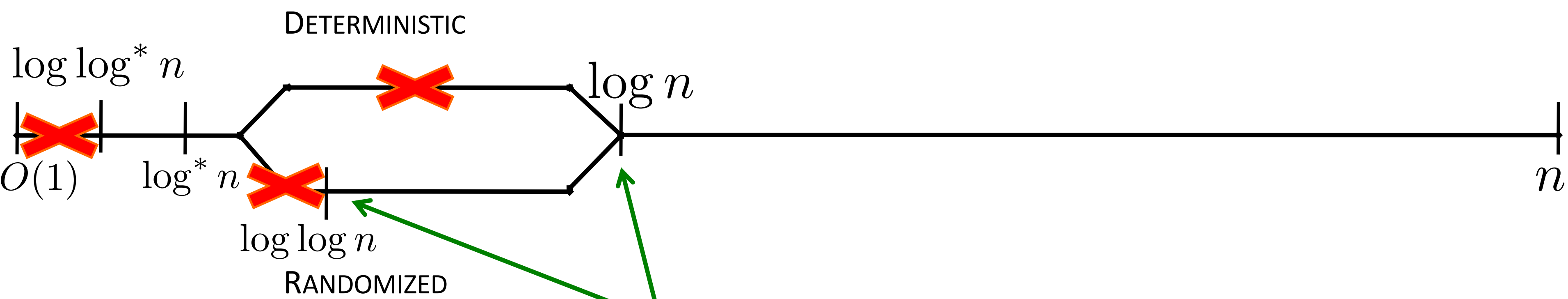
Time Hierarchies: Δ=O(1)

General graphs, Trees

Time Hierarchies: Δ=O(1)
General graphs, Trees

Chang, Kopelowitz, Pettie'16

$\mathrm{Det}_{\mathcal{P}}(n,\Delta) \leq \mathrm{Rand}_{\mathcal{P}}(2^{n^2},\Delta)$



Exponential Separations:

– Δ-coloring degree-Δ trees
– Sinkless Orientation
– (2Δ-2)-edge coloring trees

Brandt, Fischer, Hirvonen, Keller, Lempiäinen, Rybicki, Suomela, Uitto'16

Chang, Kopelowitz, Pettie'16

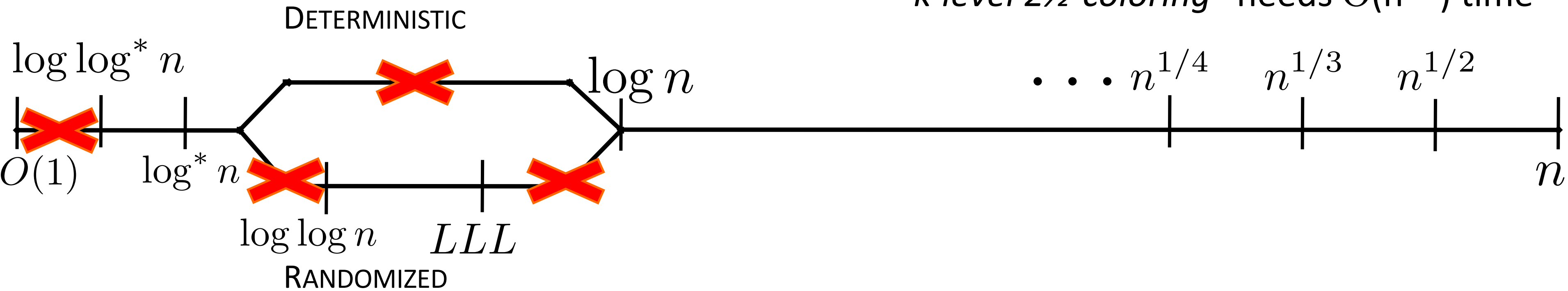
Pettie, Su'15

Ghaffari, Su'17

Chang, He, Li, Pettie, Uitto'18

Time Hierarchies: Δ=O(1)
General graphs, Trees

An infinite number of complexities.
"*k-level 2½-coloring*" needs $\Theta(n^{1/k})$ time

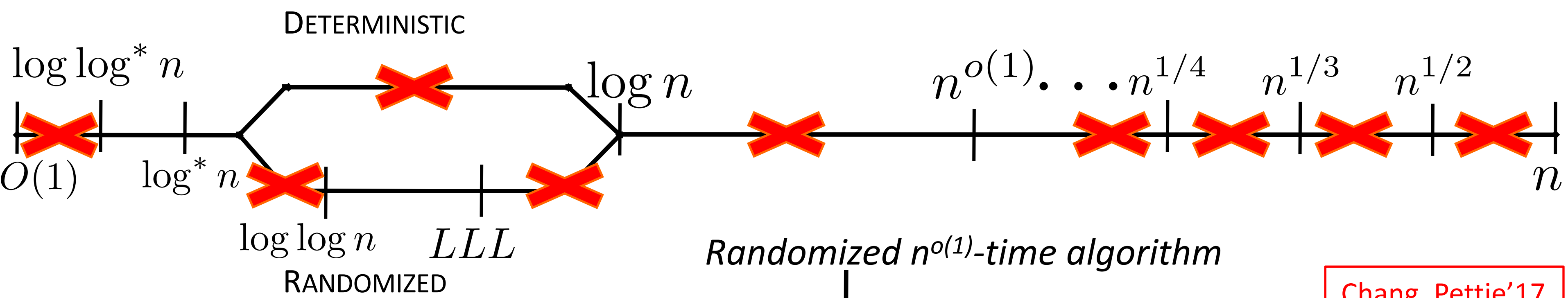


Lovász Local Lemma is "complete" for sublogarithmic randomized time!

Every o(log n)-time randomized algorithm can be automatically sped up to run in O(LLL) time.

Time Hierarchies: Δ=O(1)

Trees



Randomized $n^{o(1)}$-time algorithm

↓

Deterministic O(log n)-time algorithm

Chang, Pettie'17

Randomized $o(n)$-time algorithm

↓

Deterministic $O(\sqrt{n})$-time algorithm

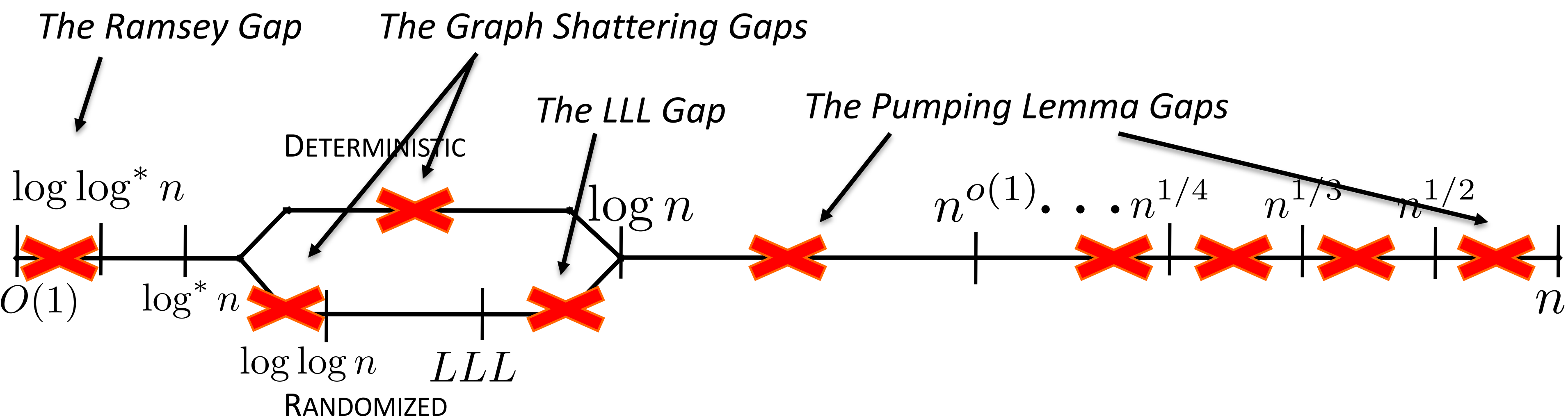
Balliu, Brandt, Olivetti, Suomela'18

Remaining gaps…

Chang (unpublished)

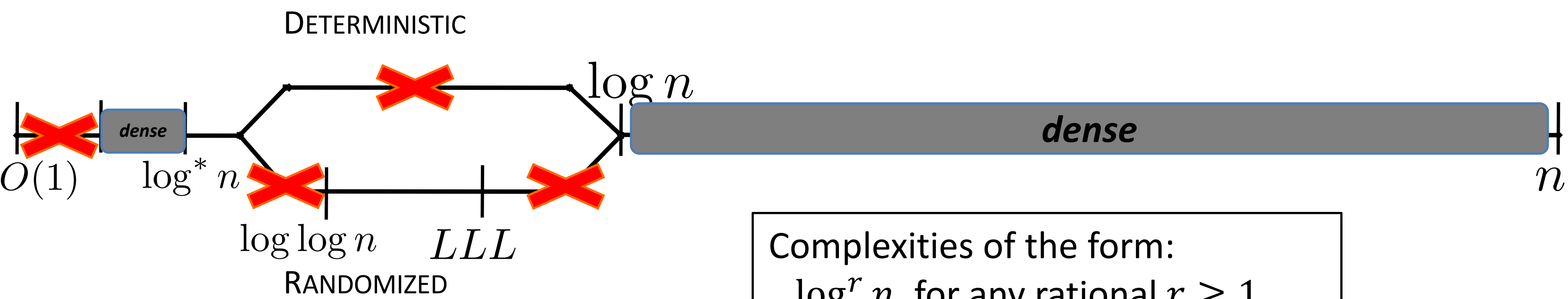
Time Hierarchies: Δ=O(1)

Trees



Time Hierarchies: Δ=O(1)

General Graphs



Complexities of the form:
- $\log^r(\log^* n)$, for any rational $r \geq 1$.
- $2^{\log^r(\log^* n)}$, for any rational $r \in (0,1)$
- $(\log^* n)^r$, for any rational $r \in (0,1)$

Complexities of the form:
- $\log^r n$, for any rational $r \geq 1$.
- $2^{\log^r n}$, for any rational $r \in (0,1)$
- n^r, for any rational $r \in (0,1)$
- $\frac{n}{\log^r n}$, for any integer $r \geq 1$

Balliu, Hirvonen, Korhonen, Keller, Lempiäinen, Olivetti, Suomela 2018

Balliu, Brandt, Olivetti, Suomela 2018

Little white lies

- What does "n" refer to in the LOCAL model?

  (1) $n = |V|$ = size of the graph.

  (2) $O(\log n)$ = bits in vertex IDs.

  (3) $1/\text{poly}(n)$ = standard error bound for ***randomized*** algs.

The Ramsey Gap

[Naor-Stockmeyer'95], [Chang-Pettie'17]

- **Step 1:** Show that any sufficiently fast algorithm can be made "order invariant."
- **Step 2:** Show that any order invariant algorithm can be tricked into thinking n=O(1).

General algorithm

```
...
Z = MSB(ID(u) XOR ID(v))
...
```

Order-invariant algorithm

```
...
If (ID(u) < ID(v)) then
        ...
Else
        ....
...
```

- $R = R(p, m, c)$: any c-coloring of the edges of a p-uniform hypergraph with R vertices has a monochromatic m-clique.

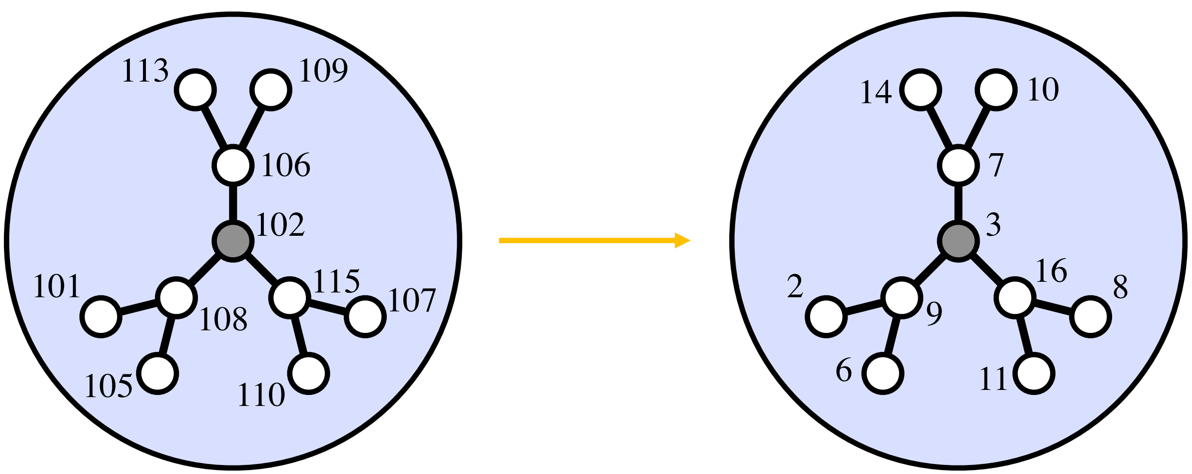
$$p = 1: \; R(p, m, c) = c(m-1) + 1$$

$$p > 1: \; R(p, m, c) \leq 2c^{R(p-1, m, c)^p}$$

- $\log^*(R(p, m, c)) = p + \log^*(m) + \log^*(c) + O(1)$

- Some algorithm runs in $t = \mathrm{o}\big(\log(\log^* \mathrm{n})\big)$ time and solves an LCL problem with radius r.
  - $p = \Delta^t$ upper bound on number of IDs we can see.
  - $m = \Delta^{t+r}$
- Consider p IDs $x_1 < x_2 < x_3 < \cdots < x_p$
- The color of hyper-edge $(x_1, x_2, \ldots, x_p)$ encodes
  - For every distinct radius-t subgraph H centered @ v,
    - For every one of p! assignments of IDs to nodes in H,
      - The output of v when the algorithm is run with this ID assignment and neighborhood H.

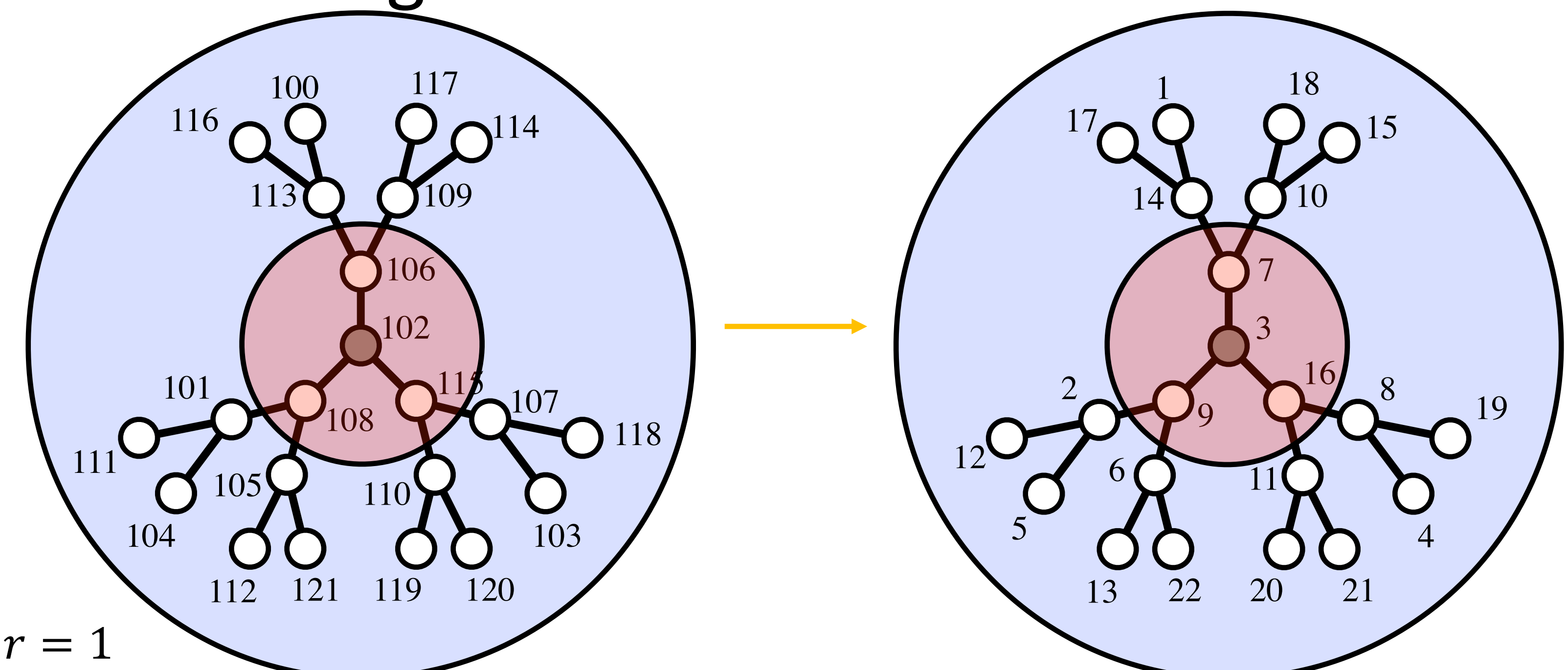
- There exists a monochromatic m-clique. W.l.o.g., suppose $S = \{1,2,3,\ldots,m\}$ is monochromatic.
- New algorithm: H = t-neighborhood of v. Reassign IDs in H to be from S in an *order-preserving* way. Run the old algorithm.



$t = 2, r = 1$

Map IDs to $\{2,3,6,7,8,9,10,11,14\}$

- There exists a monochromatic m-clique. W.l.o.g., suppose $S = \{1,2,3,\ldots,m\}$ is monochromatic.
- New algorithm: H = t-neighborhood of v. Reassign IDs in H to be from S in an *order-preserving* way. Run the old algorithm.



$t = 2, r = 1$

The "Graph Shattering" Gaps

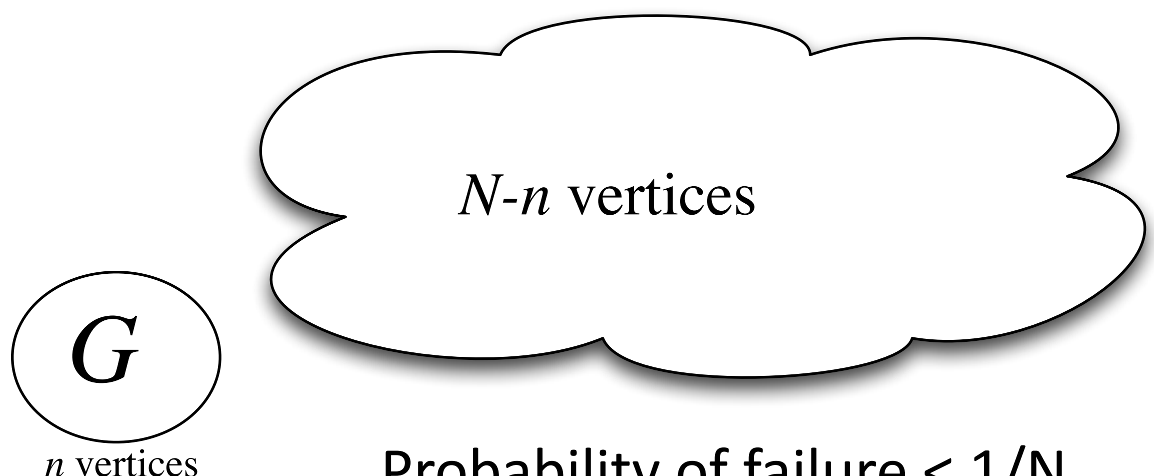
[Chang, Kopelowitz, Pettie'16]

- No **det**. complexities in $\omega(\log^* n)$—$o(\log n)$
  - **Theorem.** Any $o(\log n)$-time *deterministic* algorithm can be sped up to run in $O(\log^* n)$ time.
- No **rand.** complexities in $\omega(\log^* n)$—$o(\log \log n)$

  Alt. proof: [Fischer, Ghaffari'17]
- **Theorem.** If A_{rand} solves an LCL in $T(n, \Delta)$ time with failure probability $1/n$, then there exists an A_{det} that solves it in $T(2^{n^2}, \Delta)$ time.

Derandomization

- A_{rand} generates a string of $r(n)$ random bits.
- A_{det} will generate "random" bits using a ***magic*** function $\phi : \{0,1\}^{O(\log n)} \rightarrow \{0,1\}^{r(n)}$.
  - v's string of local random bits will be $\phi(ID(v))$.
- Imagine running A_{det} on G with a ***random*** ϕ, but telling vertices they're in a graph with $N = 2^{n^2}$ vertices.



Probability of failure < 1/N.

- The probability that a ***random*** ϕ fails to work for ***every*** graph topology and ***every*** ID-assignment is

$$2^{\binom{n}{2}} \cdot 2^{O(n \log n)} \cdot \left(\frac{1}{N}\right) < 1$$

Number of graphs | Number of ID assignments | Failure prob.

- Hence there is exists a magic ϕ that always works!

$$Det(n, \Delta) \leq Rand(2^{n^2}, \Delta)$$

A Randomized Complexity Gap

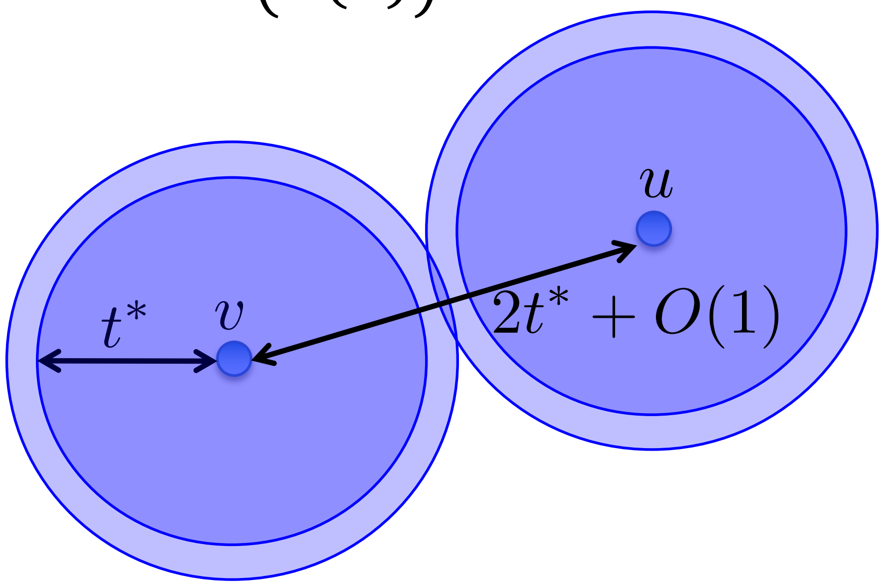
- **Theorem.** No randomized LCL complexities in $\omega(\log^* n)$— $o(\log\log n)$.

  Proof. Suppose A_{rand} solves some LCL in $o(\log\log n)$ time.
  - This implies an A_{det} that solves it in $o(\log n)$ time.
  - Any deterministic $o(\log n)$-time algorithm can be sped up to run in $O(\log^* n)$ time. [Chang, Kopelowitz, Pettie'16]

The Lovasz Local Lemma Gap

- The ***distributed*** (symmetric) LLL problem:
  - Network and dependency graph G=(V,E) are identical
  - V : “bad events”; u∈V depends on set of discr. r.v.s vbl(u)
  - $E = \{(u,v) : \mathrm{vbl}(u) \cap \mathrm{vbl}(v) \neq \emptyset\}$
  - ***d*** = maximum degree in G, ***p*** = maximum Pr(v).
    - Satisfies some ***LLL Criterion***, e.g., $ep(d+1) < 1$, $\boxed{p(ed)^c < 1.}$
- Compute a variable assignment such that no bad event occurs.

- Suppose A solves some LCL problem in *sub*logarithmic time with failure probability $1/n$.
  - For any ε>0, can write time as $T(n, \Delta) \leq C(\Delta) + \epsilon \log_\Delta n$
- n^* = min. value such that: $T(n, \Delta) = t^* \leq \frac{1}{2c} \log_\Delta n^* - O(1)$
  - Follows that $t^* = O\big(C(\Delta)\big)$.



every vertex sees a subgraph that is
consistent with an n-vertex graph.*

- Build the dependency graph:
  - X_v = the random bits generated locally at v.
  - vbl(v) = $\{X_u \mid u \in N^{t^*+O(1)}(v)\}$
  - E_v = the event that v's neighborhood is incorrectly labeled, when running alg. A with "n" = n^*.
  - $H = (\{E_v\},\ \{(E_u,E_v) \mid \text{dist}(u,v) \le 2t^*+O(1)\})$
  - LLL parameters: $p = 1/n^*$, $d = \Delta^{2t^*+O(1)}$

$$pd^c = p \cdot \Delta^{c(2t^*+O(1))} < (1/n^*)n^* = 1$$

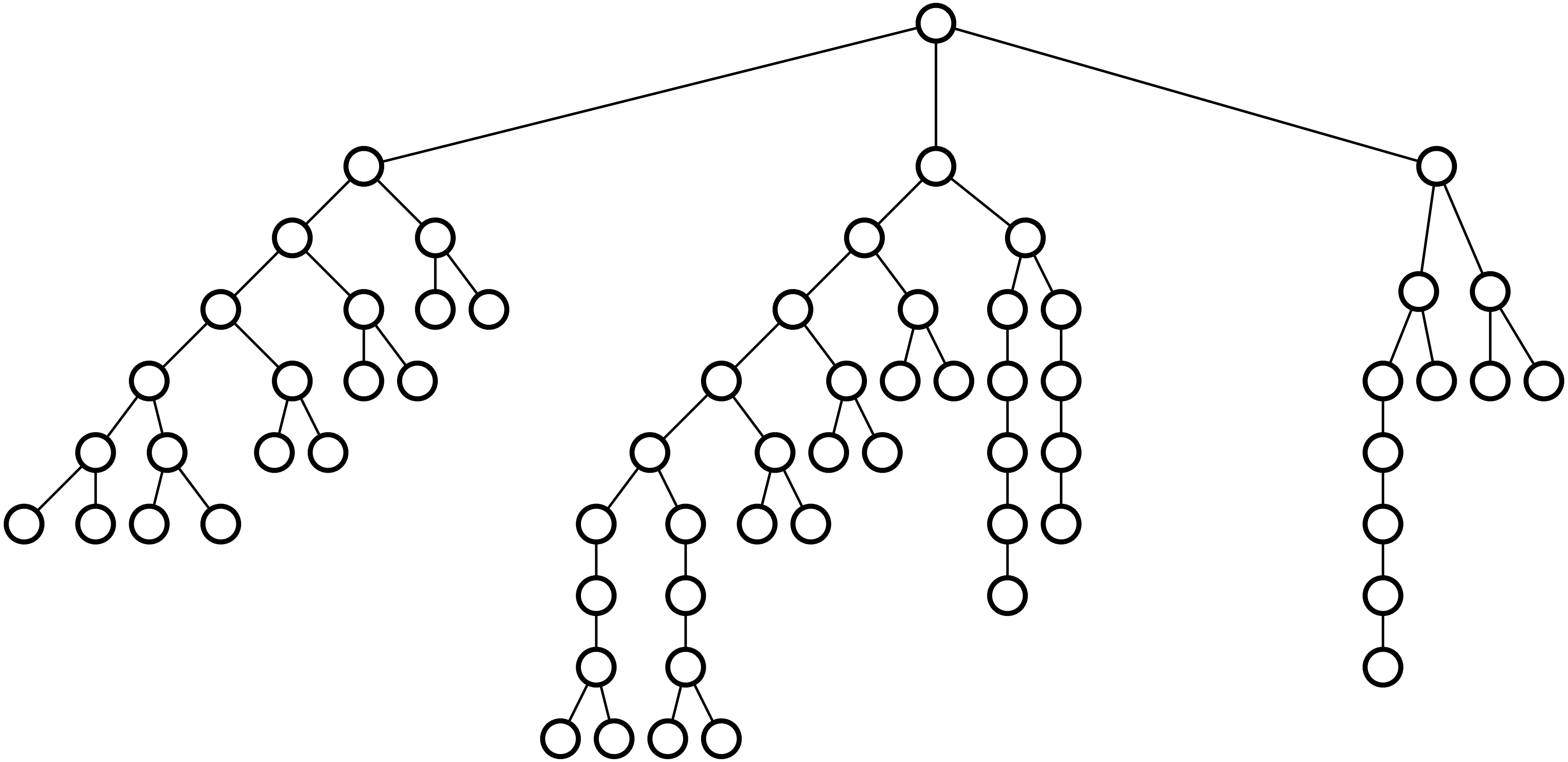
- Run a distributed LLL algorithm on "H."
  - 1 step in H simulated with $O(C(\Delta))$ steps in G.
  - Alg. A can be automatically sped up to $O(C(\Delta)\cdot T_{LLL})$ time.

Pumping Lemma Speedups

- **Theorem:** any ***randomized*** $n^{o(1)}$-time LCL algorithm on ***trees*** can be converted into a ***deterministic*** $O(\log n)$-time algorithm.

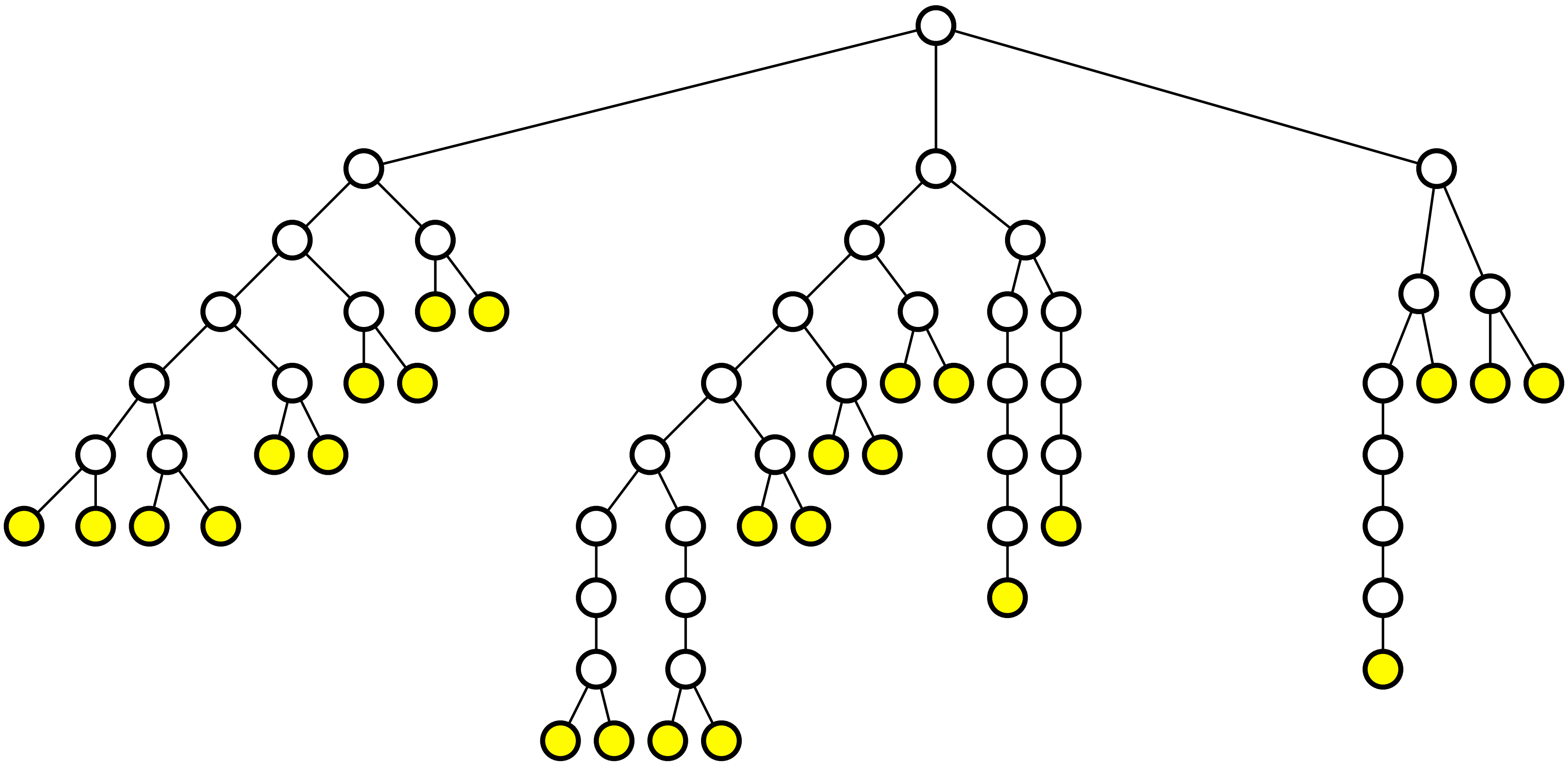
Rake and Compress

[Miller and Reif 1989]



Rake and Compress

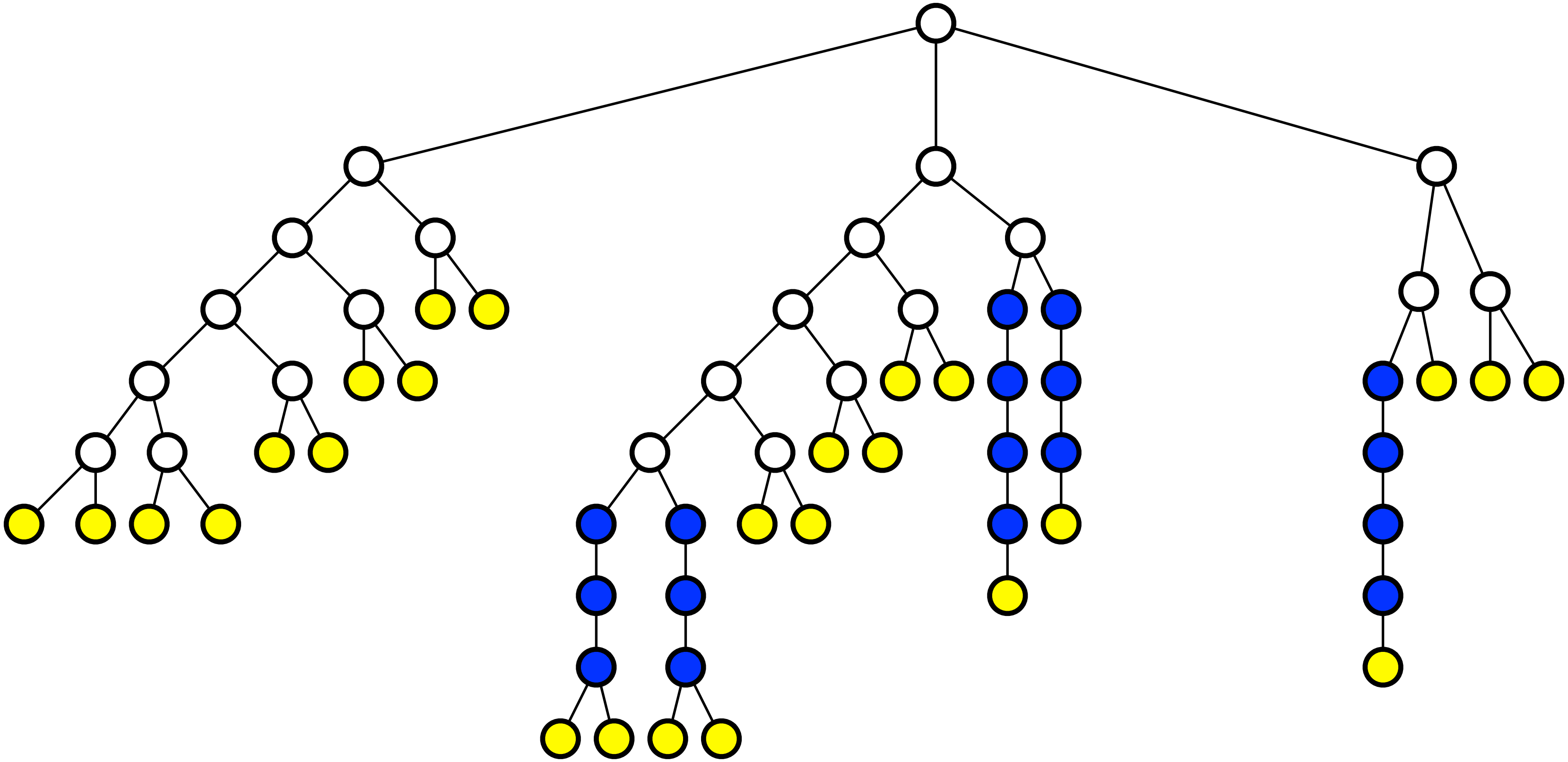
[Miller and Reif 1989]



- ***Rake***: remove all leaves

Rake and Compress

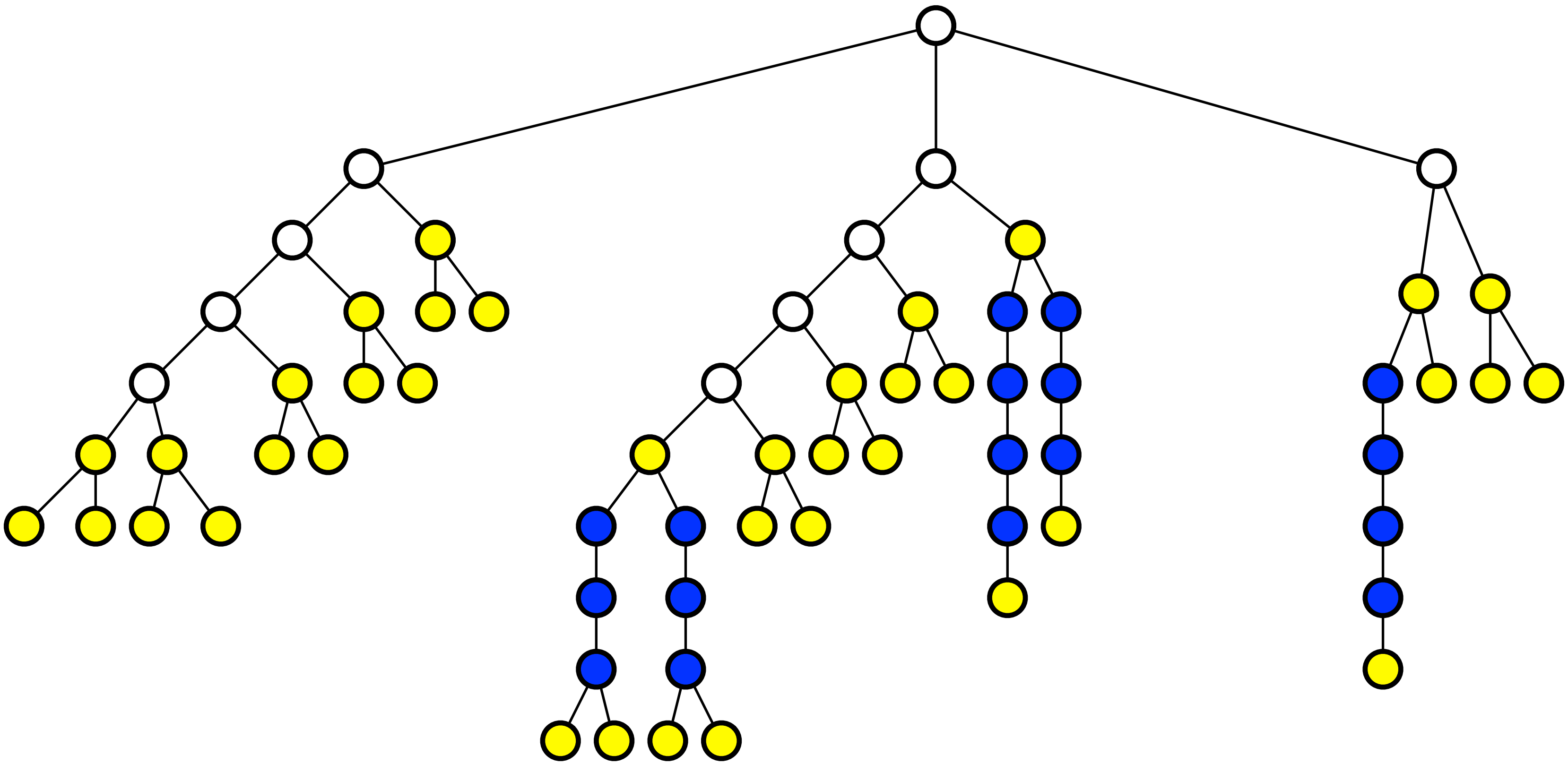
[Miller and Reif 1989]



- ***Rake***: remove all leaves
- ***Compress***: remove chains of degree-2 vertices.

Rake and Compress

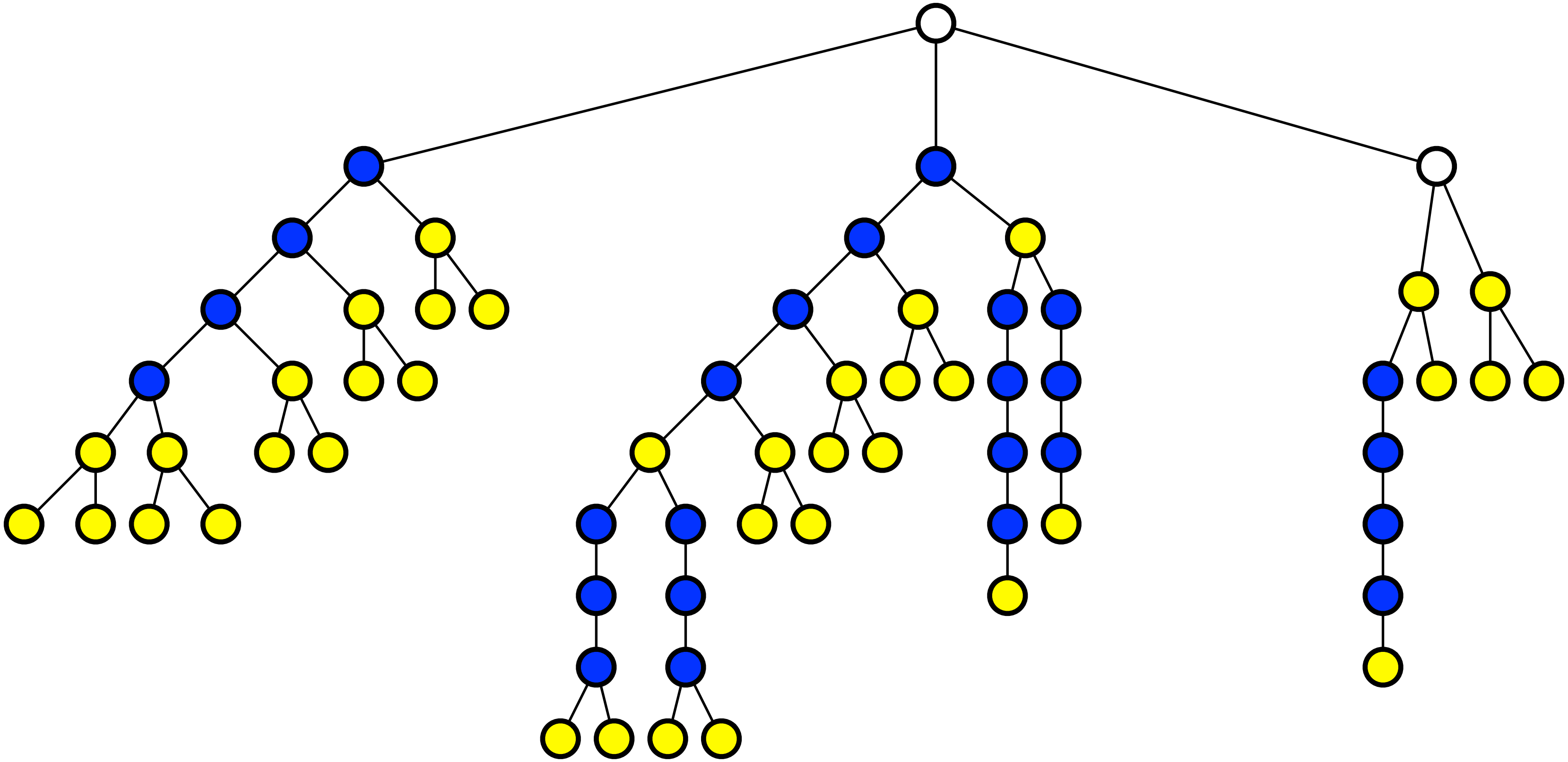
[Miller and Reif 1989]



- ***Rake***: remove all leaves
- ***Compress***: remove chains of degree-2 vertices.

Rake and Compress

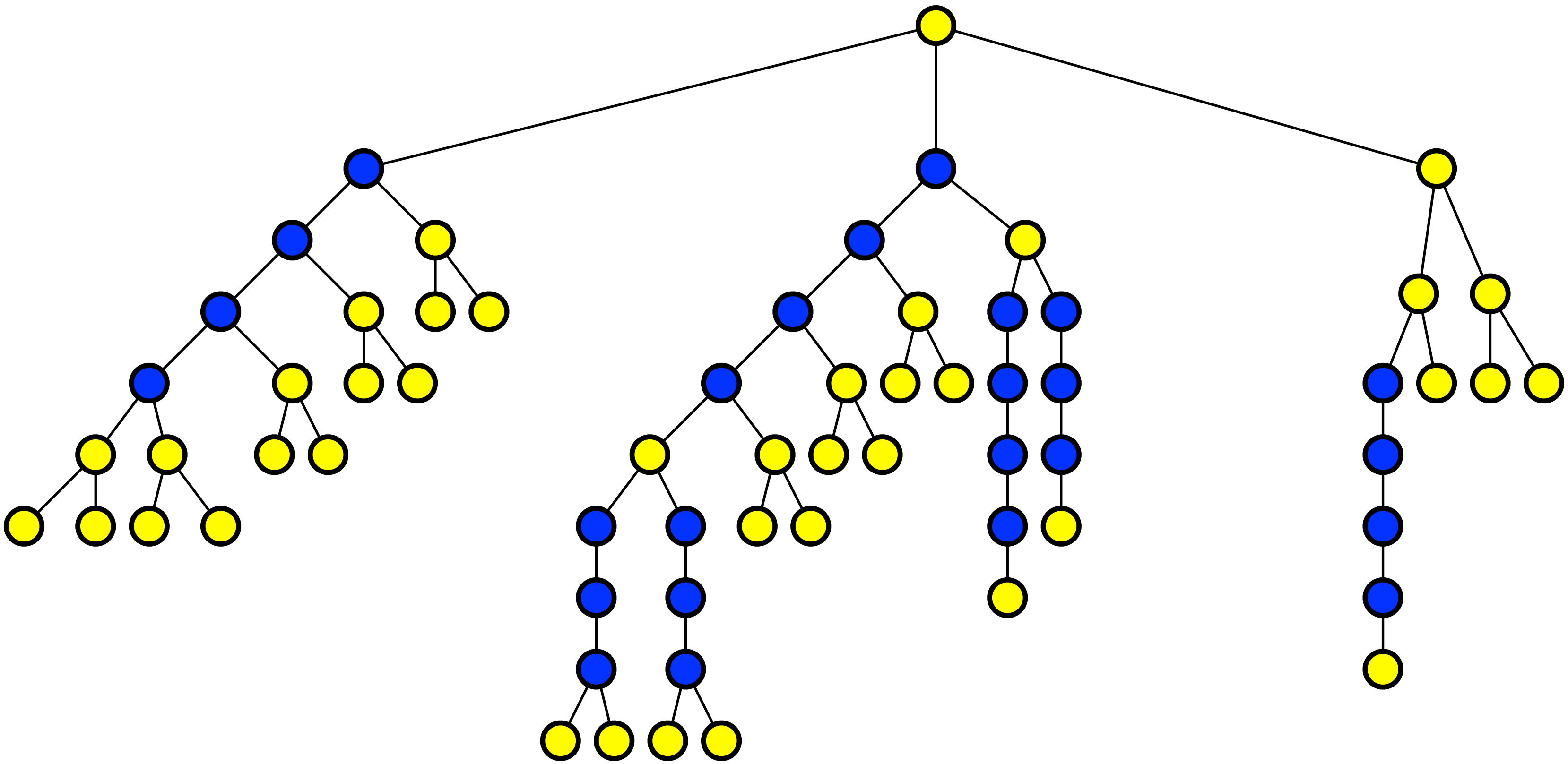
[Miller and Reif 1989]



- ***Rake***: remove all leaves
- ***Compress***: remove chains of degree-2 vertices.

Rake and Compress

[Miller and Reif 1989]



- ***Rake***: remove all leaves
- ***Compress***: remove chains of degree-2 vertices.
- O(log n) rakes & compresses suffice.

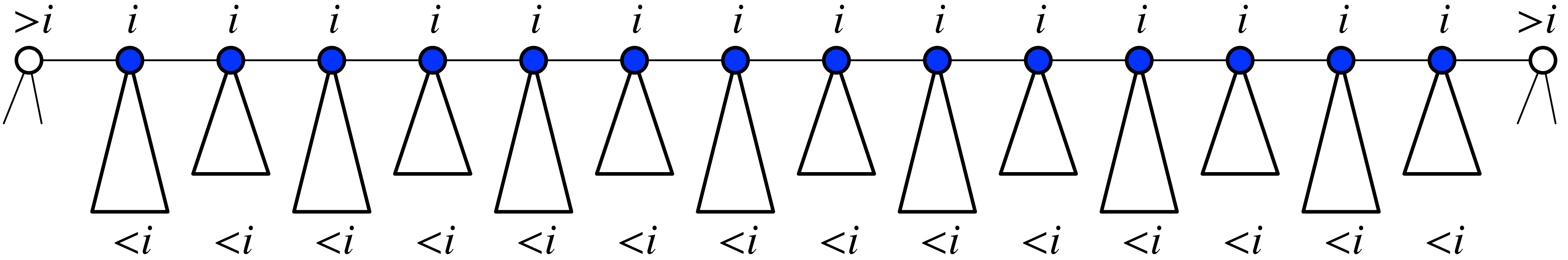
Rake and Compress

[Miller and Reif 1989]

- ***Case 1:*** O(log n) *rakes* suffice to decompose the tree
  - Diam. = O(log n); any LCL can be solved in O(log n) time.
- ***Case 2:*** *compress* occasionally removes $\omega(1)$-length paths.

Removing Long Paths

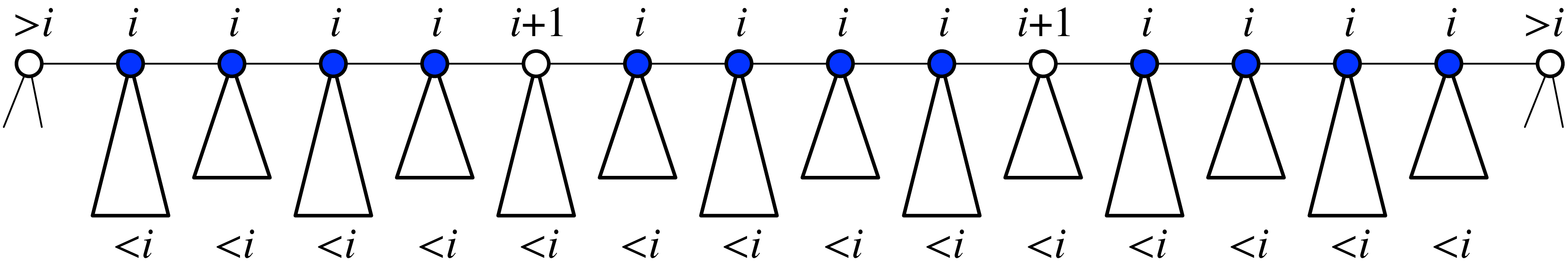
[Miller and Reif 1989]



- A sufficiently long path of ***nodes*** removed in i^{th} iteration
- Subtrees removed in iterations < i.
- Bookended by nodes removed in iteration > i.

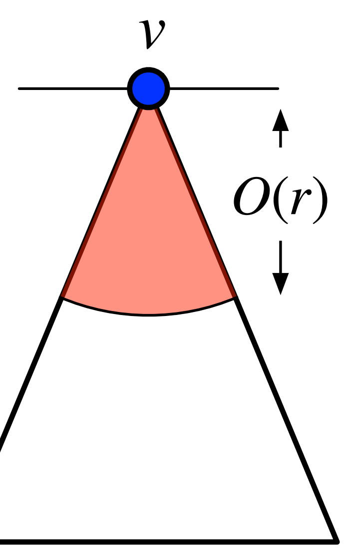
Removing Long Paths

[Miller and Reif 1989]



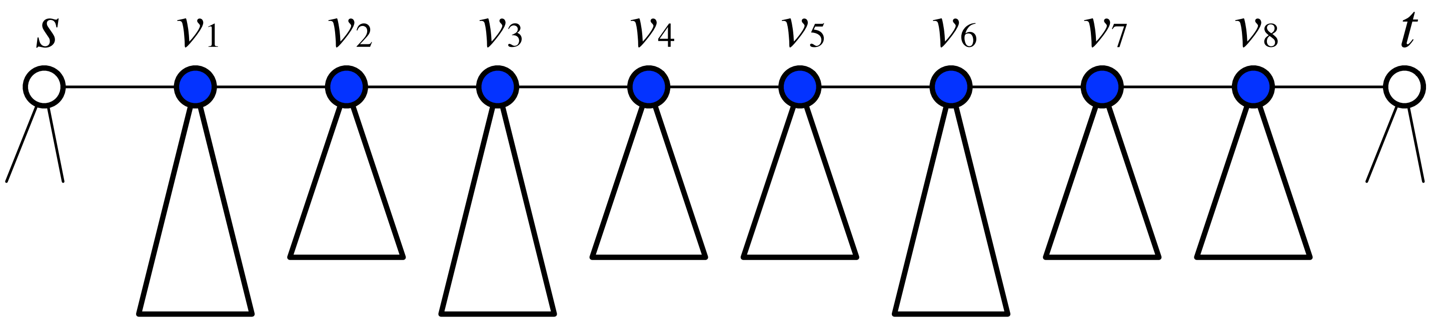
- A sufficiently long path of ***nodes*** removed in i^{th} iteration
- Subtrees removed in iterations < i.
- Bookended by nodes removed in iteration > i.
- Can assume w.l.o.g. that the path has ***length O(1)*** by "promoting" a well-spaced set of nodes to level-(i+1).

Class



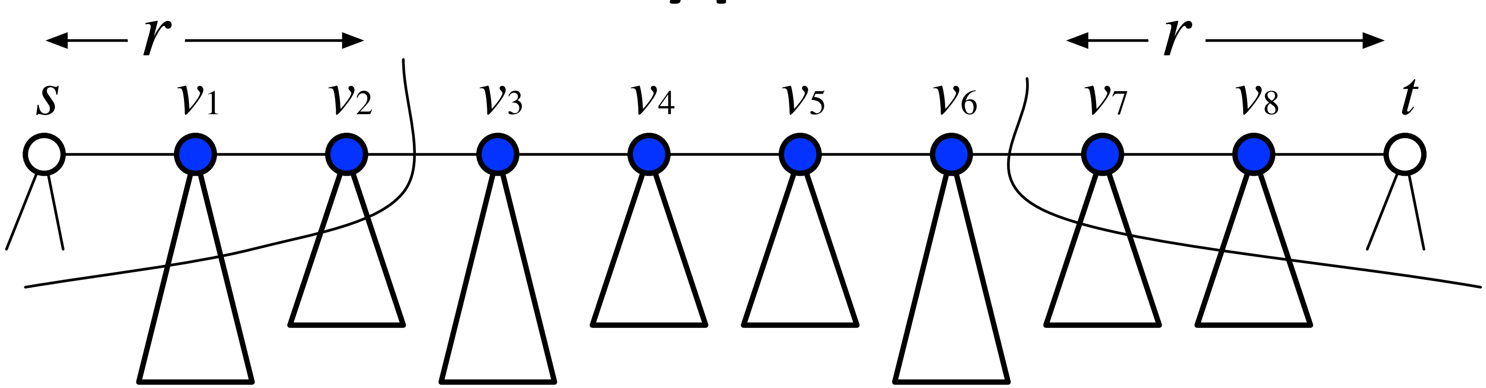
- ***Class(v)*** : the set of all labelings of $N^{O(r)}(v)$ that can be extended to the ***whole*** subtree rooted at v.
- # Classes = O(1). (Δ and Σ_{out} constants.)

Type



- c_i = Class(v_i)
- Relevant information (c_1, c_2, c_3, …)

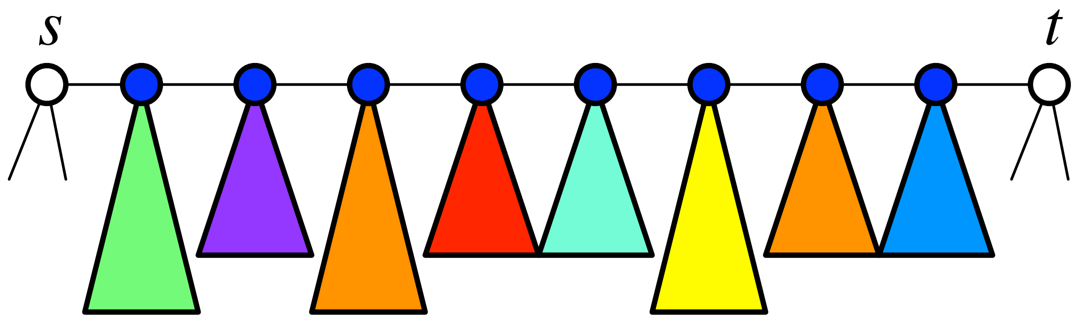
Type



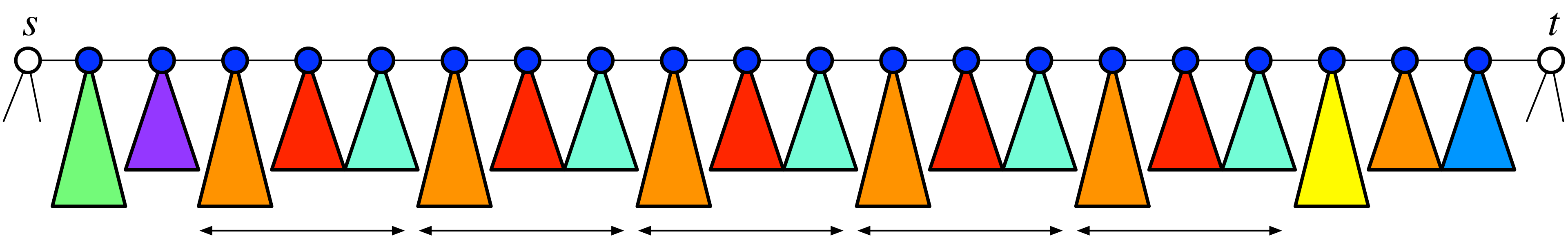
- c_i = Class(v_i)
- Relevant information (c_1, c_2, c_3, …)
- ***Type(s,t)*** : set of all labelings of $N^r(s) \cup N^r(t)$ that can be extended to subtrees of v_1, v_2, …
- Type(s,t) computable by a finite automaton that scans class vector (c_1, c_2, c_3, …).

Pumping Trees

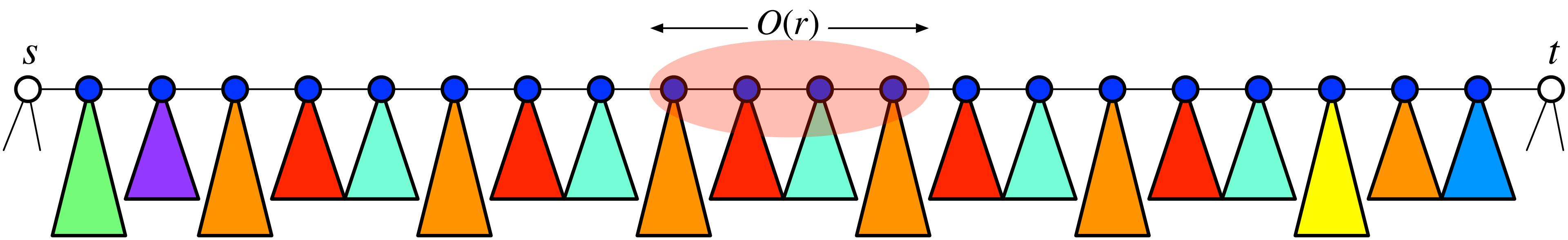
- If the path is sufficiently long, the automaton will enter some state twice.



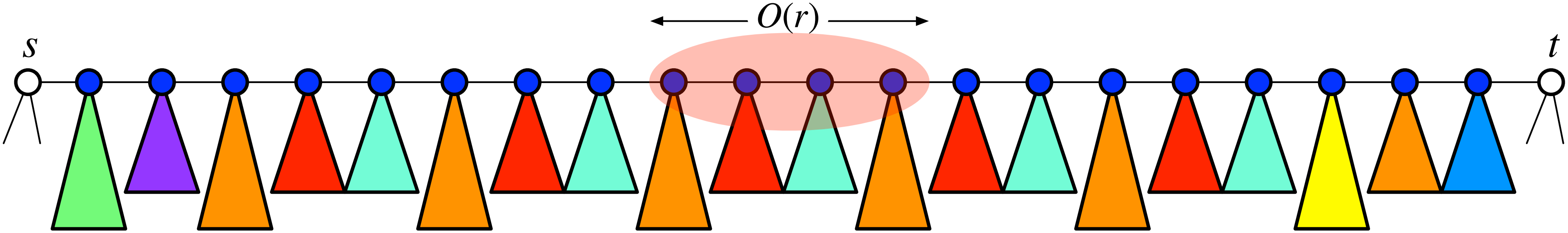
- Can create a “pumped” tree; Type(s,t) is unchanged.



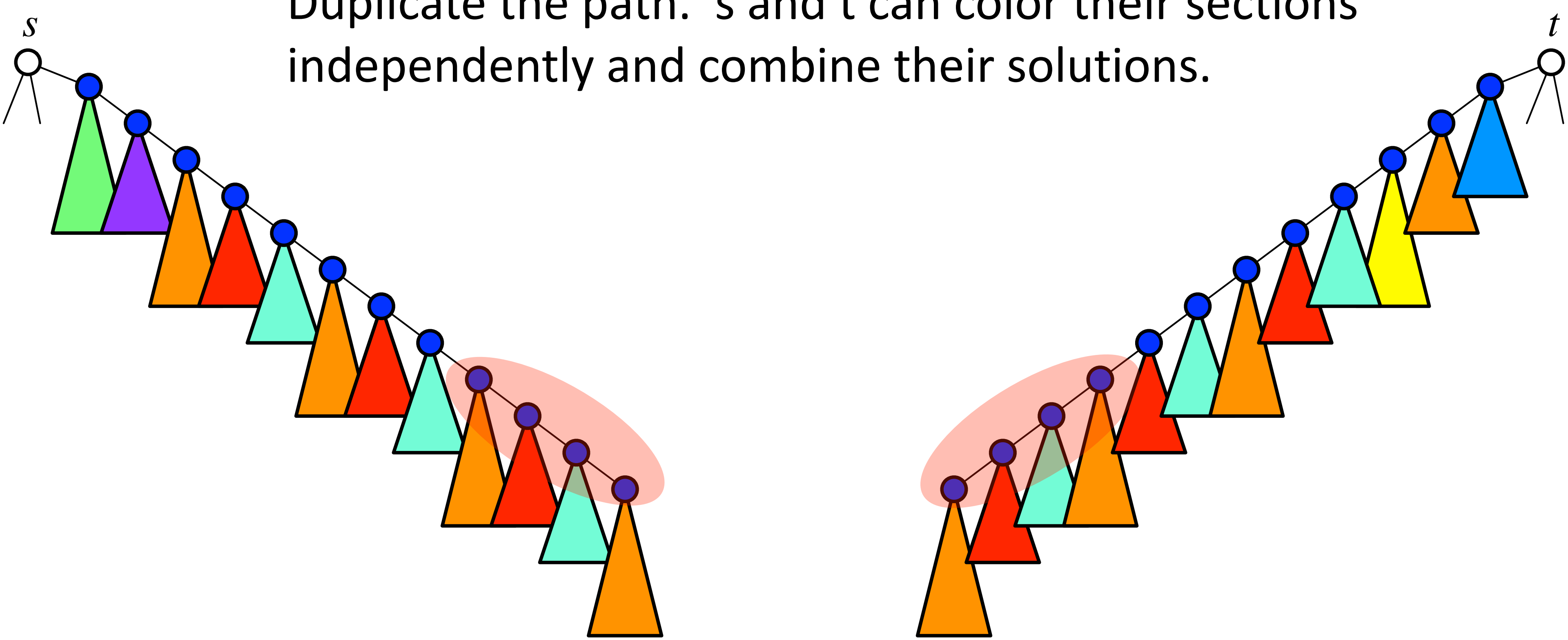
Pumped Trees



- Pump the path to be *very* long.
- Any $n^{o(1)}$-time algorithm run on the "middle" of the path does not depend on s nor t.
- Pre-commit to the output labeling of an O(r) neighborhood around the middle.



Duplicate the path. s and t can color their sections independently and combine their solutions.



- Apply ***pumping***, ***precommit***, and ***duplication*** to every Compress operation.
- Any subtree can be freely replaced by a (smaller) subtree of the same Class.

- The $n^{o(1)}$-neighborhood of any vertex in the final “imaginary” tree is a function of the $O(\log n)$-neighborhood in the actual tree.
- Any correct labeling in the imaginary tree can be converted to one in the actual tree.

Open Question

What is the LOCAL complexity of the LLL ?

- Probably need to solve rand. and det. complexities simultaneously. $\Theta(\log\log n)$ rand. and $\Theta(\log n)$ det.?
- Conceivable that the "original" LLL with criterion $ep(d+1) < 1$ is a harder problem.

Thank you!