Compact and Distributed Data Structures

Cyril Gavoille

7 décembre 2000

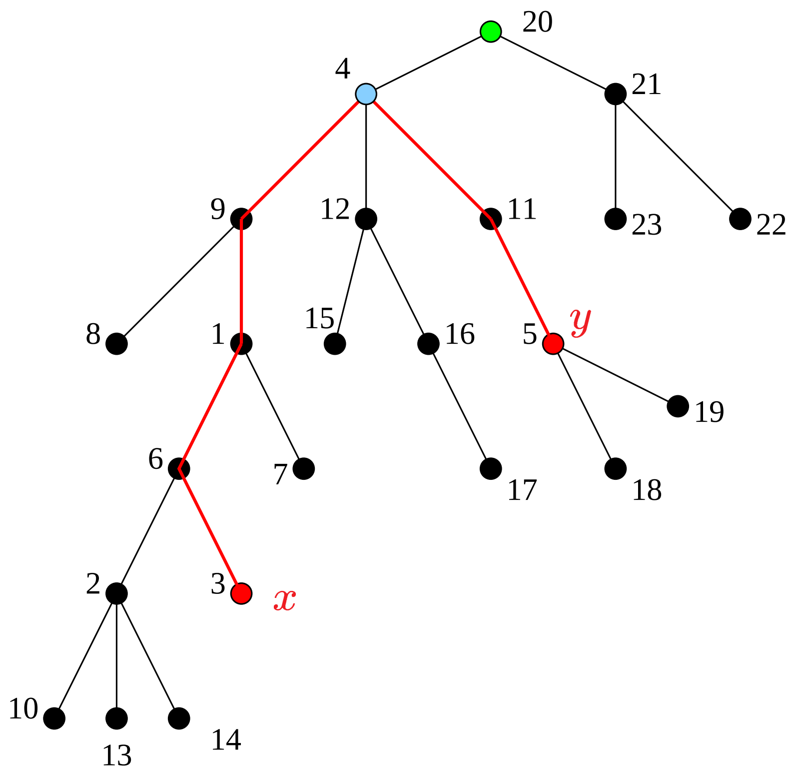
Contents

1. Example

Problem: Given a tree T and two nodes x, y, compute $\text{dist}_T(x, y)$?

T with

n nodes



node	1	2	3	4	5	6	7	8	9	10	...	n
father	**9**	**6**	**6**	**20**	**11**	**1**	**1**	**9**	**4**	**2**	**...**	**21**

Solution:

1. compute the path from x to the root, $(x, \text{father}(x), \text{father}(\text{father}(x)), \ldots, \text{root})$
2. compute the path from y to the root,
3. compute the *lowest common ancestor of x and y*
4. *sum difference of depths*

$$
\begin{array}{rcl}
(x, u_1, \ldots, \ldots, u_i, \ldots, u_{p-1}, u_p) & \rightarrow & O(n) \text{ time} \\
(y, v_1, \ldots, v_j, \ldots, v_{q-1}, v_q) & \rightarrow & O(n) \text{ time} \\
\hat{} \qquad \text{lca}(x, y) & \rightarrow & O(n) \text{ time} \\
\text{dist}_T(x, y) = i + j & \rightarrow & O(1) \text{ time} \\
\hline
\text{Total} & = & O(n) \text{ time}
\end{array}
$$

Remark: Given a linear time pre-processing on T, and using $O(n)$ space it is possible to compute $\mathsf{dist}_T(x,y)$ in $O(1)$ time.
("On-line queries on static tree")

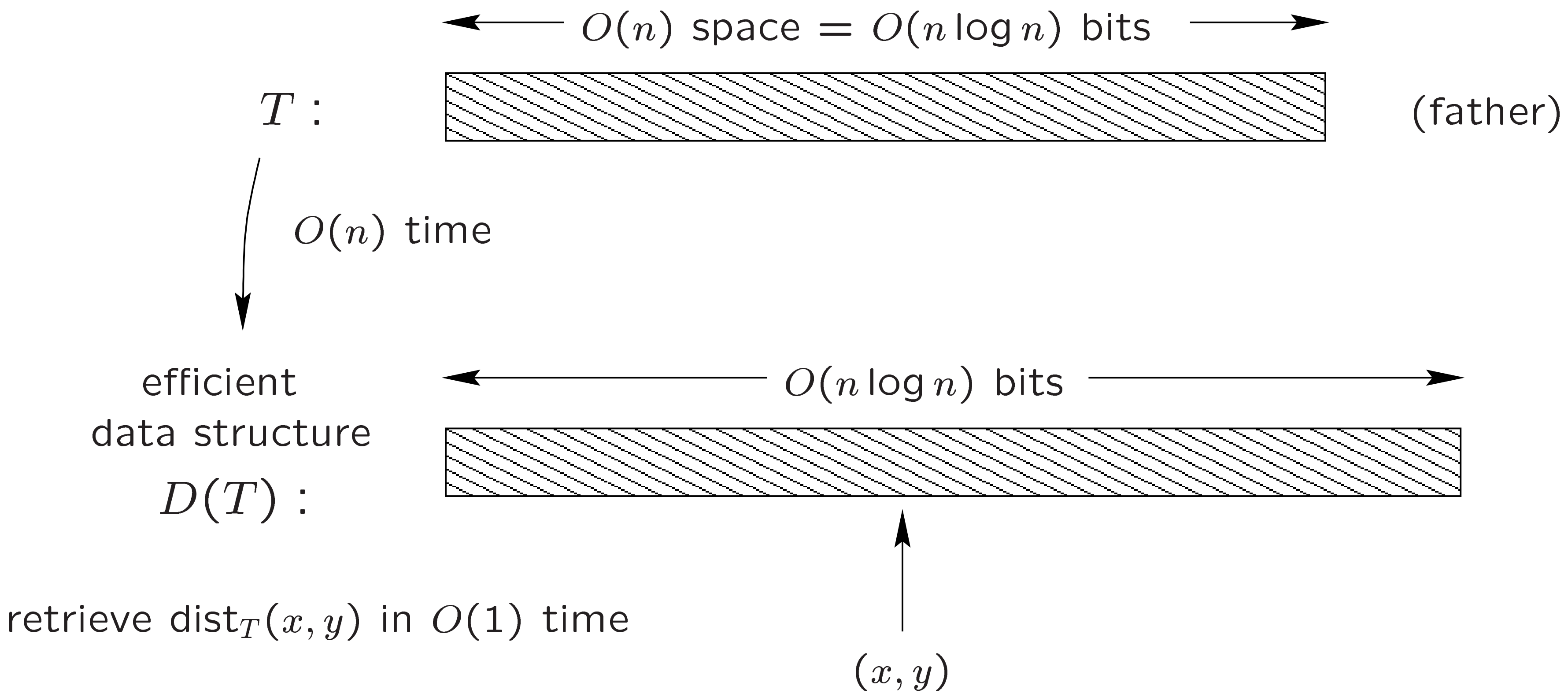
1.

node	1	2	3	4	5	6	7	8	9	10	...	n
depth	**3**	**5**	**5**	**1**	**3**	**4**	**4**	**3**	**2**	**6**	**...**	**2**

2. [Harel-Tarjan'84] $\mathsf{lca}(x,y)$ in constant time with $O(n)$ space (and $O(n)$ time pre-processing).

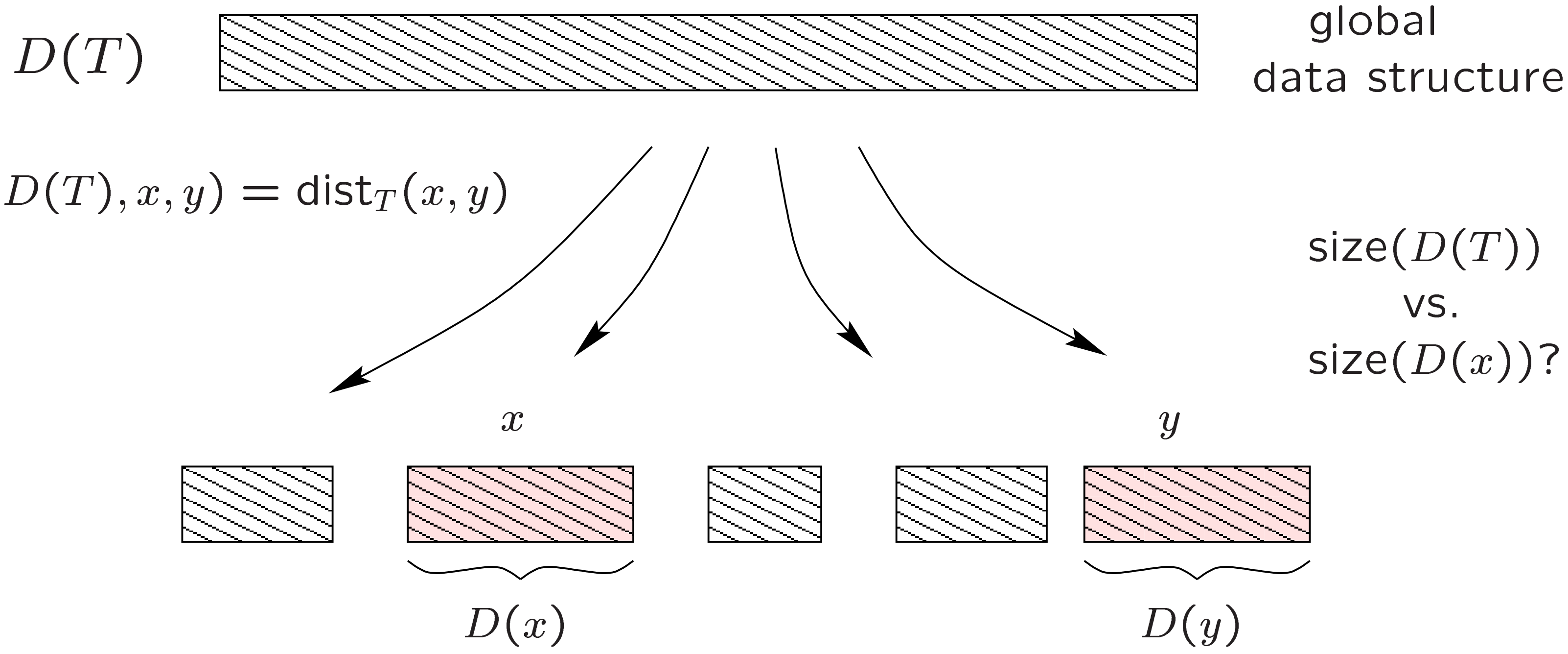
3. $z = \mathsf{lca}(x,y)$, $\mathsf{dist}_T(x,y) = \mathsf{depth}(x) + \mathsf{depth}(y) - 2\mathsf{depth}(z)$.

Thus,



$O(n)$ space $= O(n \log n)$ bits
T :
(father)
$O(n)$ time
$O(n \log n)$ bits
efficient data structure $D(T)$:
retrieve $\mathrm{dist}_T(x, y)$ in $O(1)$ time
(x, y)

Distributed Data Structures



$\text{query}(D(T), x, y) = \text{dist}_T(x, y)$

$\text{size}(D(T))$
vs.
$\text{size}(D(x))$?

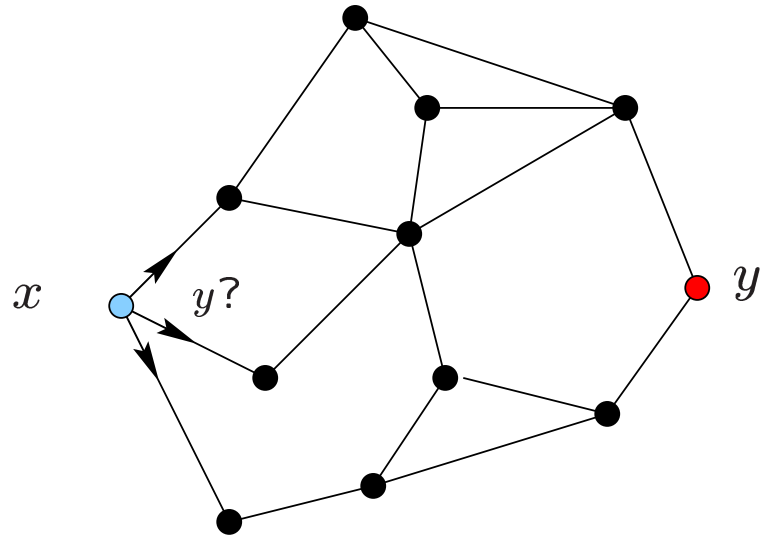
$f(D(x), D(y)) = \text{dist}_T(x, y)$

$\Rightarrow$ Applications to distributed computing

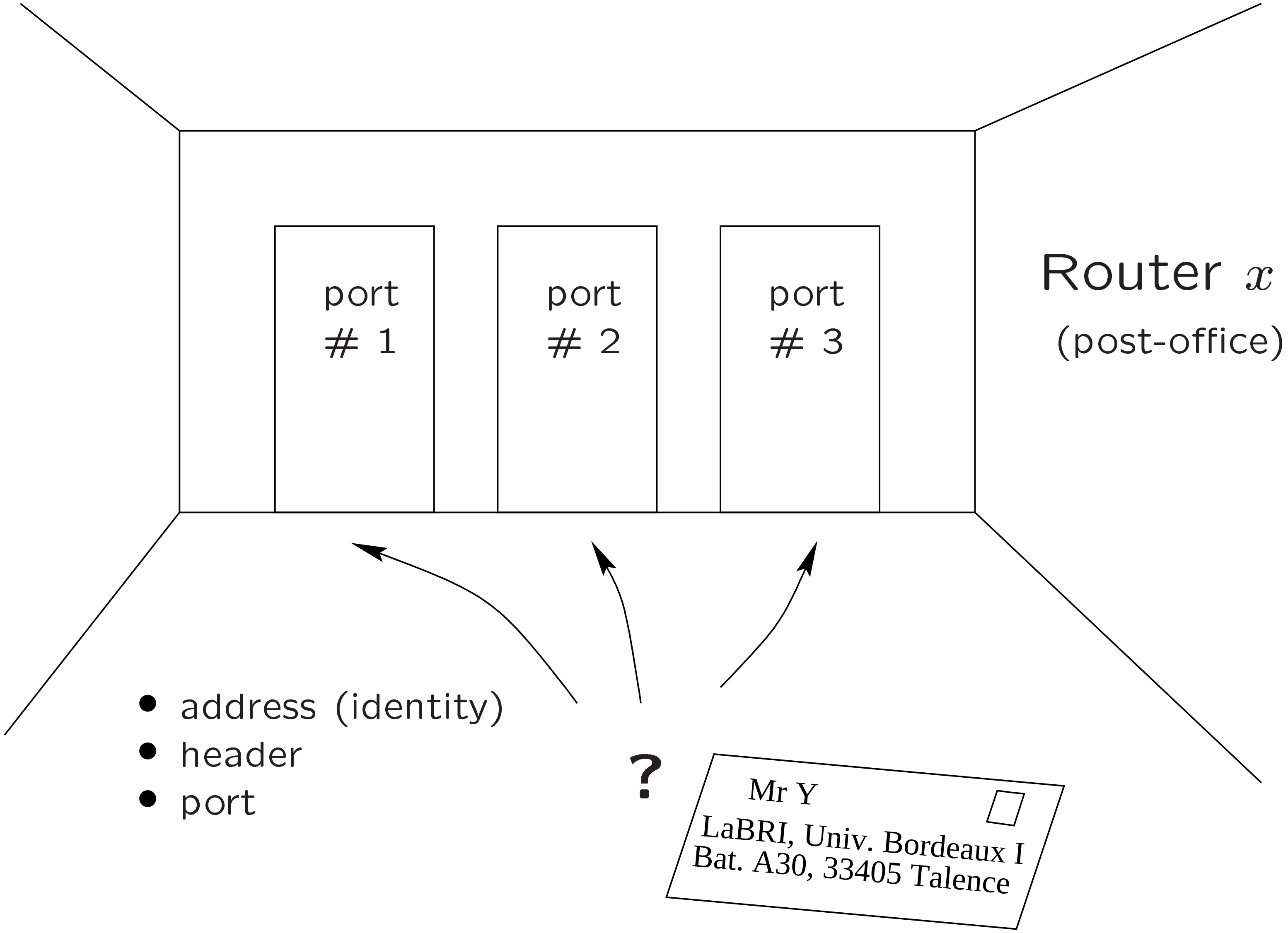
2. Routing

(with compact data structures)

Problem: Given a (connected) graph G, find an "efficient" distributed data structure on G for routing query type.

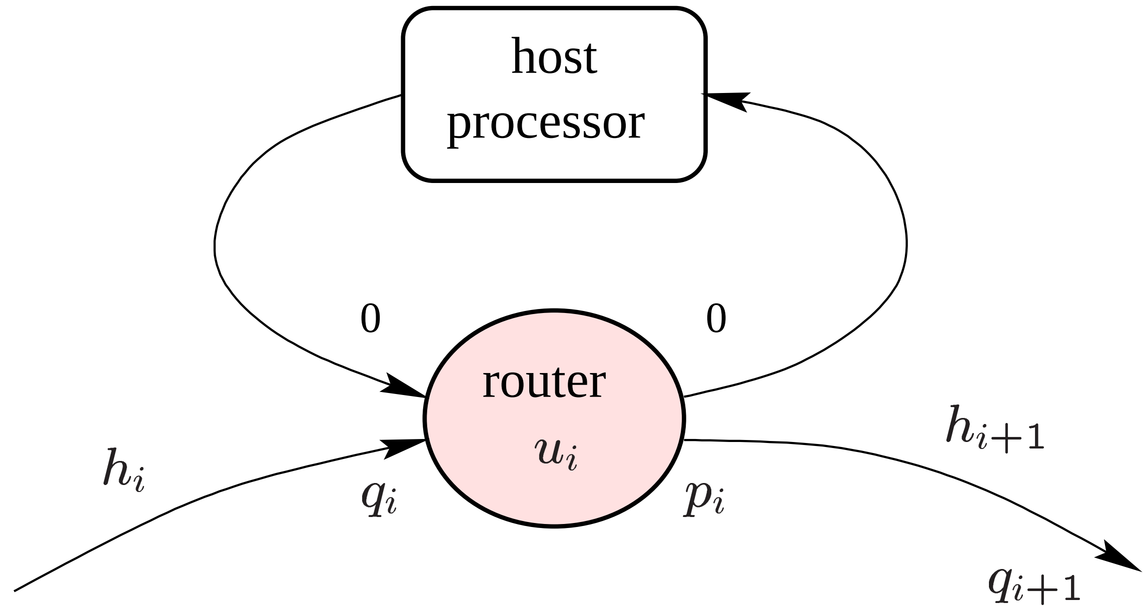


Terminology



- address (identity)
- header
- port

R_{u_i}: routing function $R_{u_i}(q_i, h_i) = (p_i, h_{i+1})$



Goal: Build efficient R_{u_i} (e.g., shortest path) with compact & local data structures, small header/address size, fast query time, ...

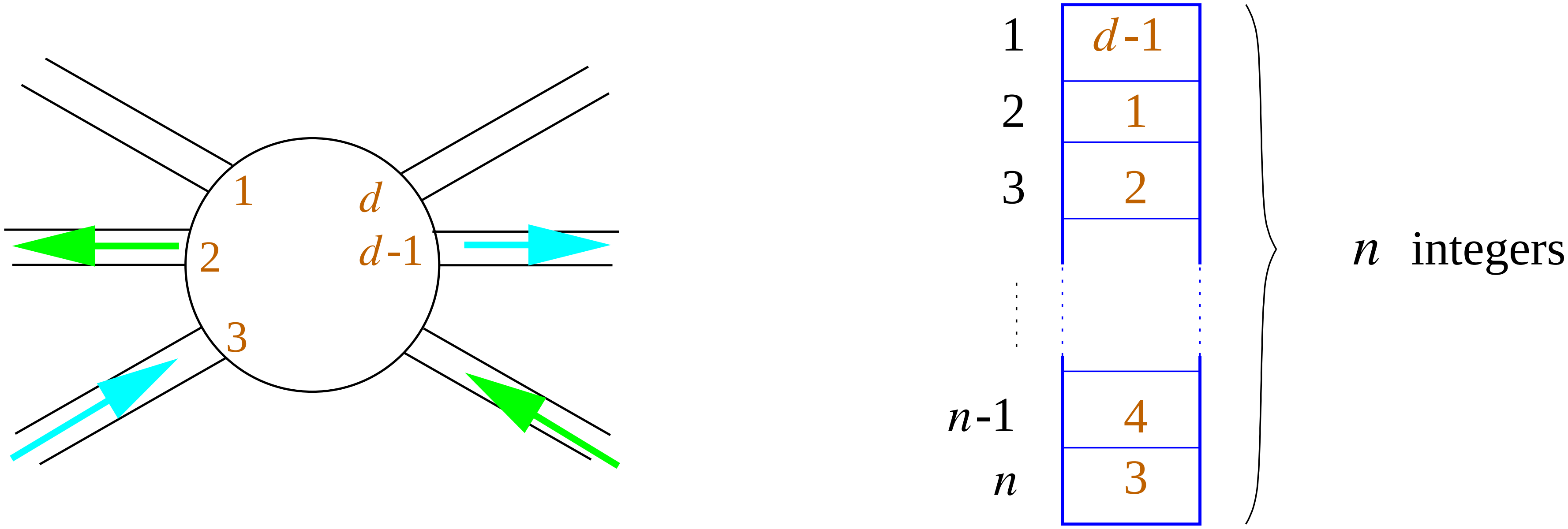
Selection of Results

(7 int'l journals, 15 int'l conferences)

$\Rightarrow$ Some key results (average/worst-case/planar/trees)
Some ideas for proofs
Some open problems

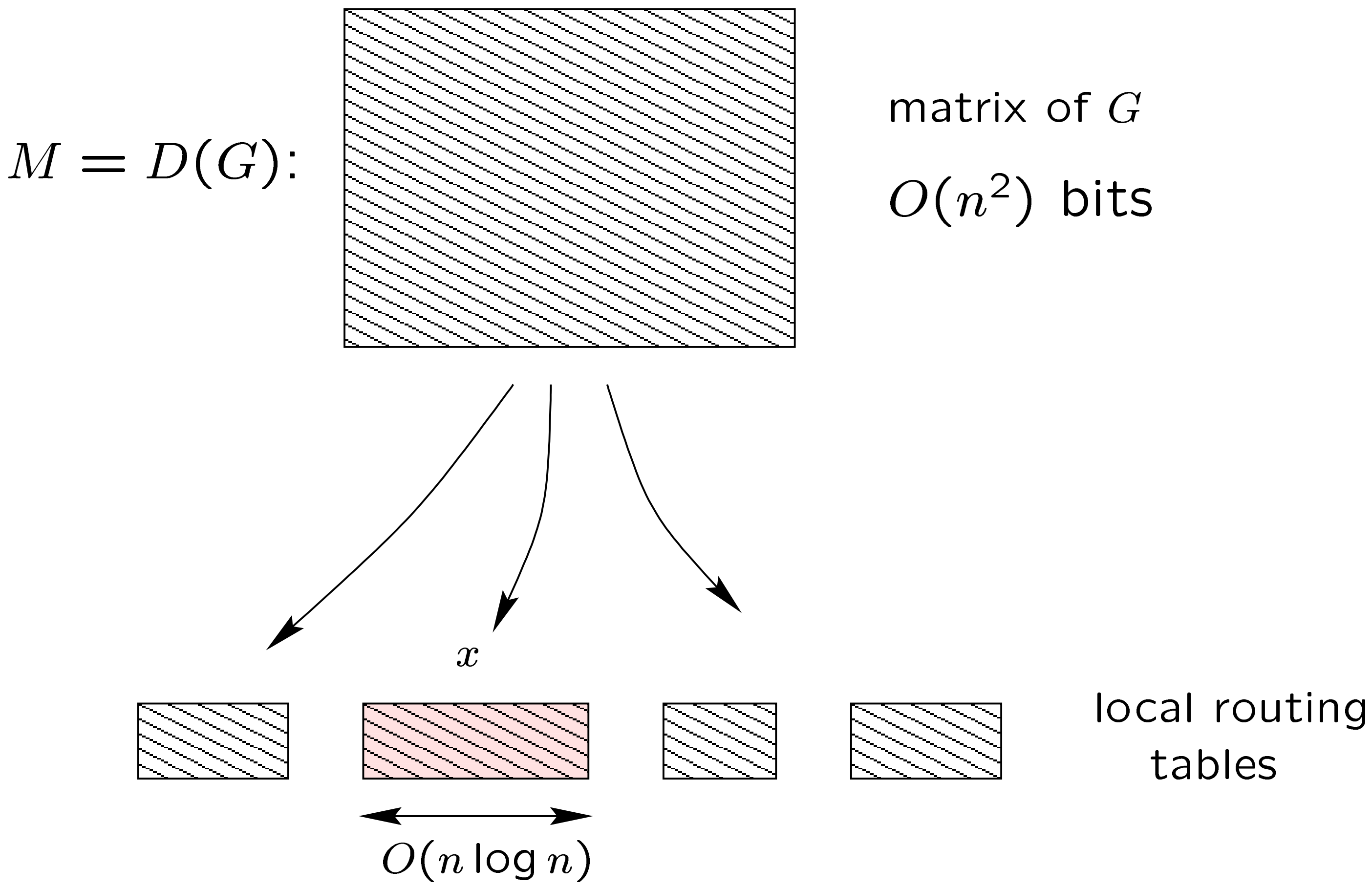
$\Rightarrow$ Shortest paths only
Routing tables model

Routing Tables



message towards node 3
message towards node 1

Result: For every graph, $O(n \log n)$ bits/node suffice.

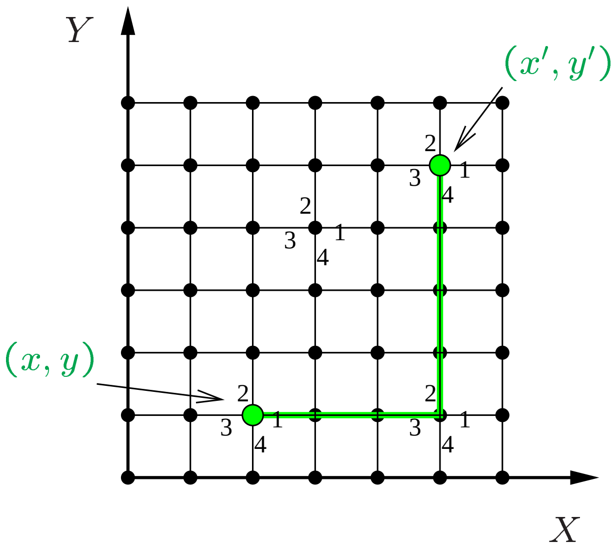


Can we do better than $O(n \log n)$ bits/node? $O(n)$? $o(n)$?

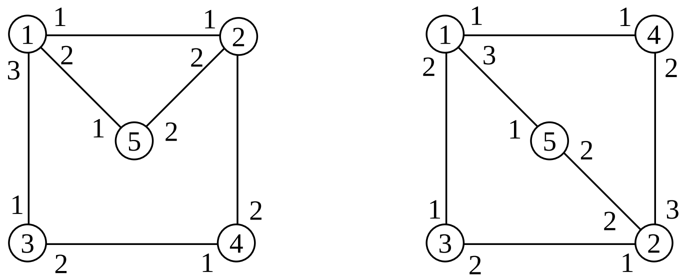
Warning!

$$\begin{aligned} 2^{\Theta(n^2)} \text{ graphs} &\Rightarrow \exists G_0 \text{ with } \Theta(n^2) \text{ bits} \\ &\Rightarrow 1 \text{ router with } \Omega(n) \text{ bits} \\ & \text{FALSE!} \end{aligned}$$

We can choose “good” sets of shortest paths, “good” addresses and port numbers.



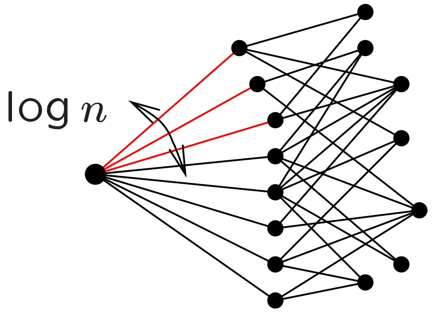
Two graphs may have the same shortest path routing tables [Peleg-Upfal’89]



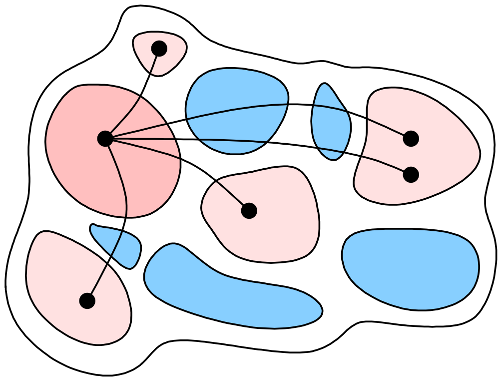
	1	2	3	4	5 (dest.)
1	-	1	3	1	2
2	1	-	1	3	2
3	1	2	-	2	1
4	1	2	1	-	2
5	1	2	1	2	-

$O(n)$ bits for the average case

$G \in \mathcal{G}_{n,\frac{1}{2}}$, results for a fraction of $1 - o(1)$ of all the n-node graphs.



- [Buhrman-Hoepman-Vitányi'96]
$3n + o(n)$ bits/node



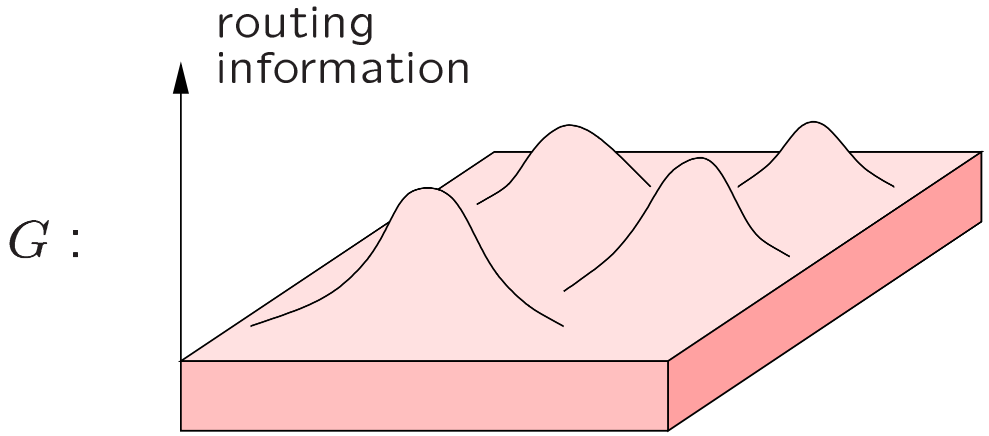
- [Gavoille-Peleg'99]
$n + O(\log^4 n)$ bits/node
(interval routing, clique decomposition)

Worst-Case

$O(n)$ bits/node for almost all graphs, but $\Omega(n \log n)$ in the worst-case! [Gavoille-Pérennès'96] (required for $\Theta(n)$ nodes).

1. Construction of graphs with $\pi = \Theta(n \log n)$ random bits

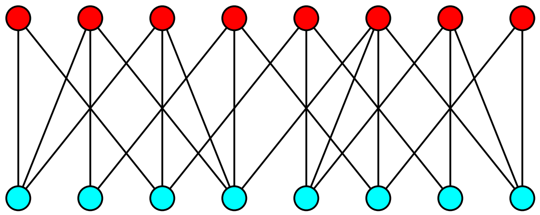
2. Any shortest path routing from 4 particular nodes to a set of $O(n)$ target nodes suffices to rebuild $\Omega(\pi)$ bits. (The routing functions of these 4 nodes "compute" a large part of π.)



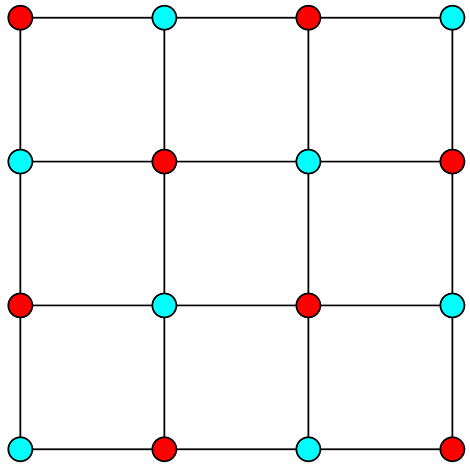
Routing in Planar Graphs

Does planarity help for routing?

Does a nice drawing help?

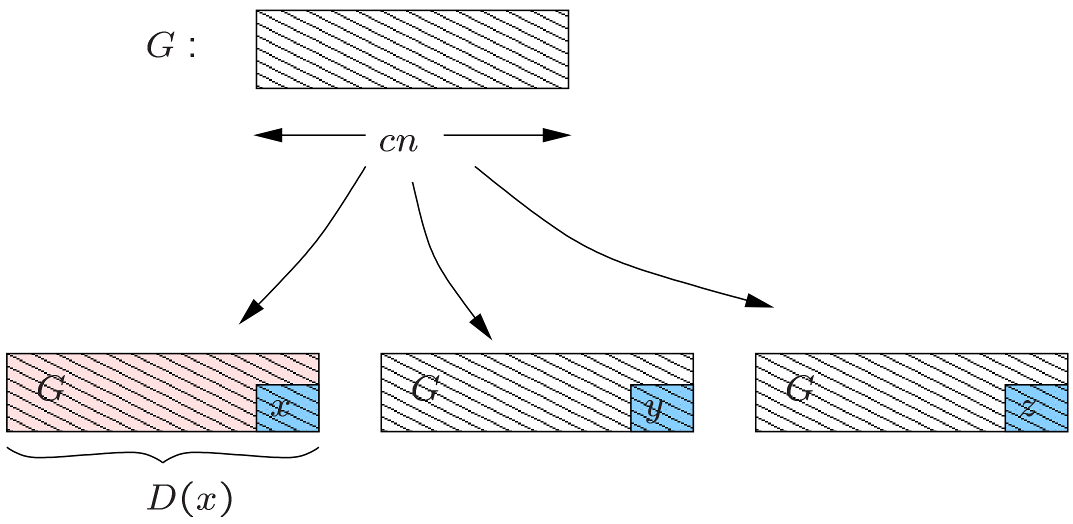


=



A Compact Solution

$O(n)$ bits suffice to code any (unlabeled) planar graph
(# planar $\sim 2^{cn+o(n)}$).

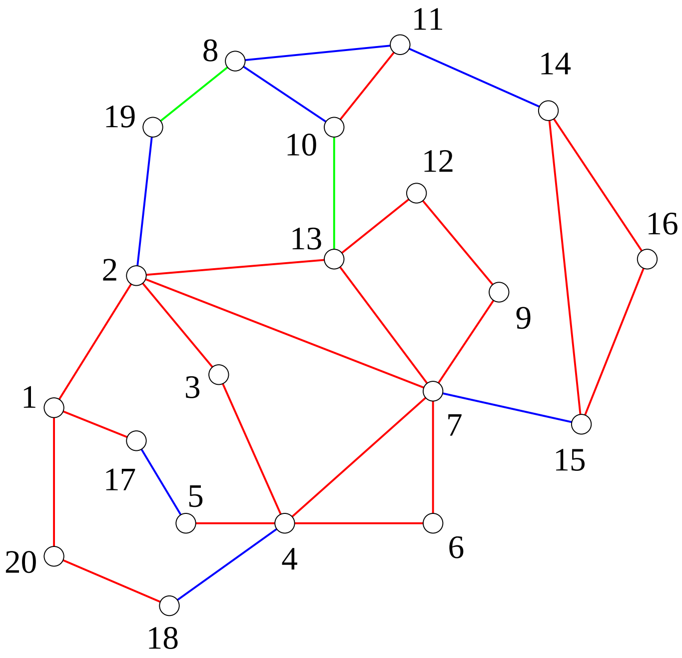


Problem: Query time per node?

1. time to rebuild G from $cn + o(n)$ bits

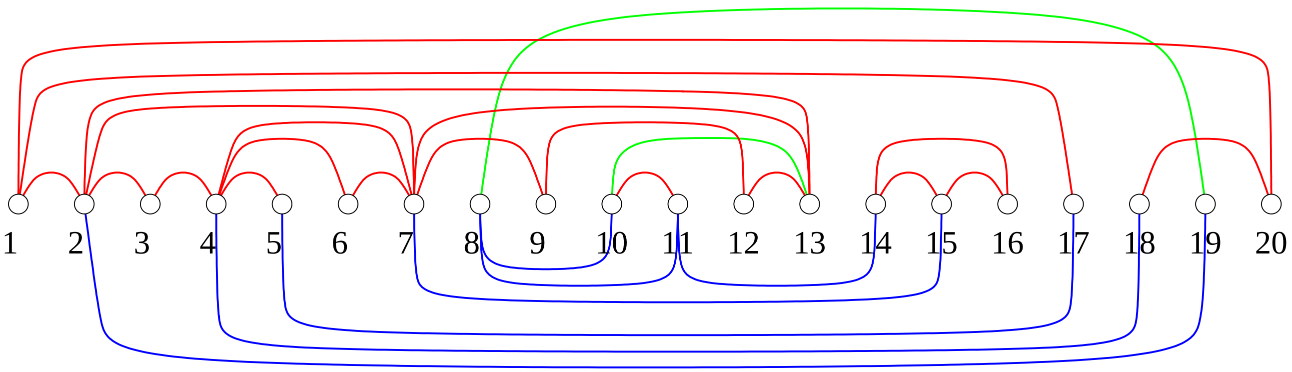
2. time to (locally) compute a shortest path $\rightarrow$ $\Omega(n)$ time !

k-Page Representation

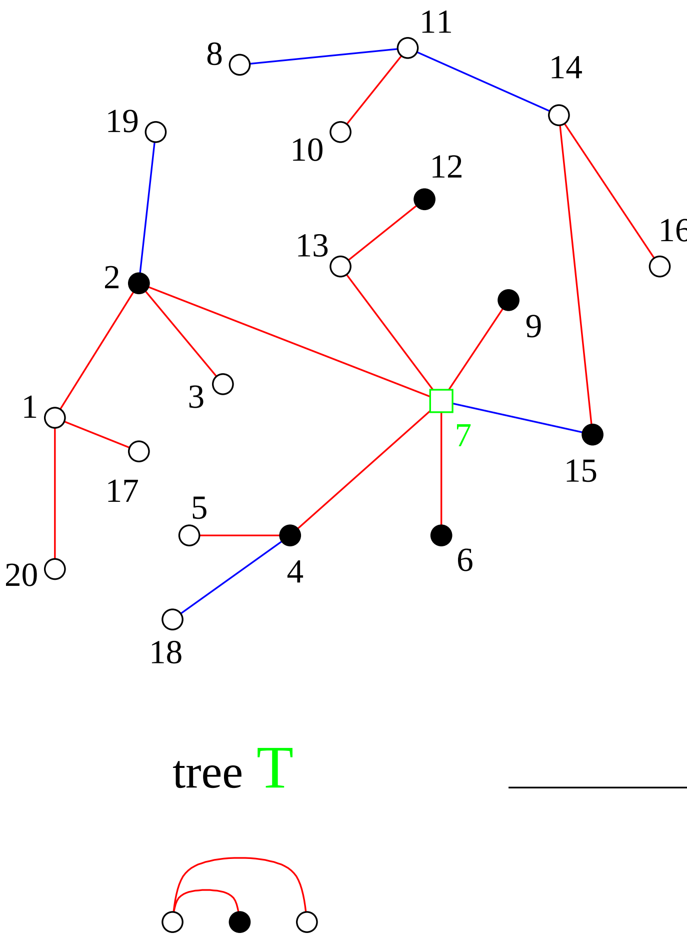


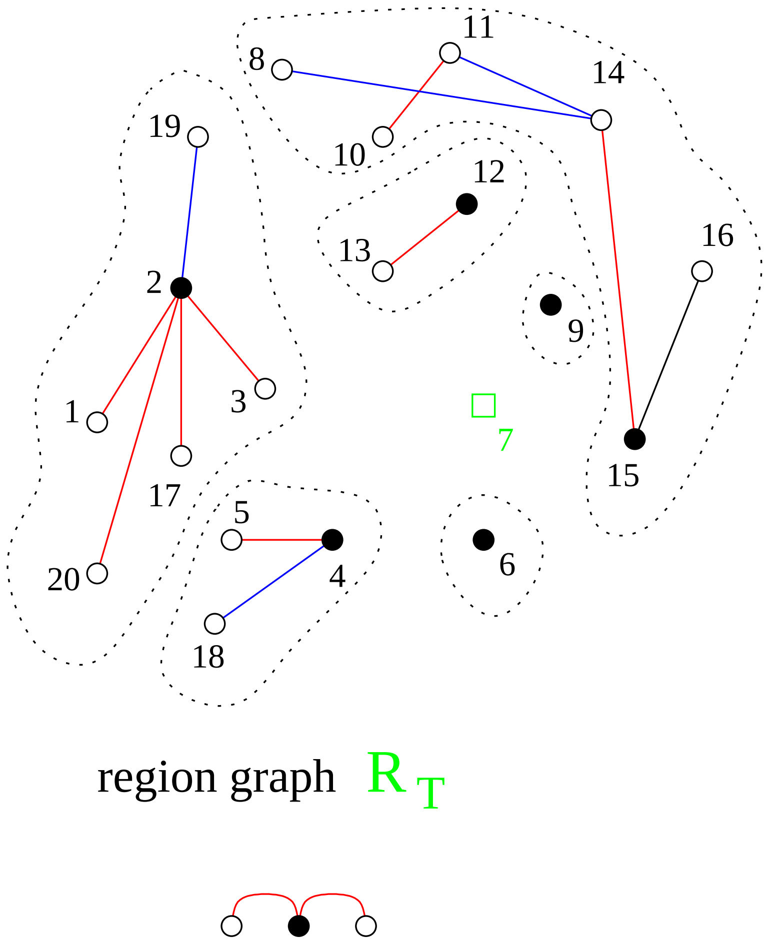
[Yannakakis'86]:

#page(planar) $\leqslant$ 4

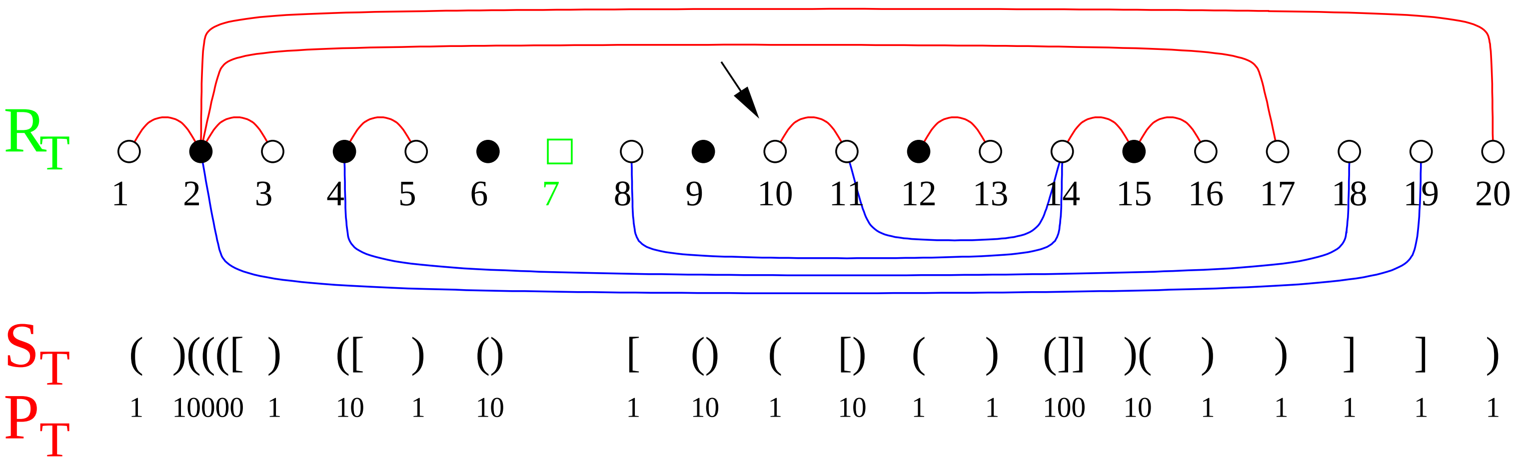


Coding of a Shortest Path Routing Tree in $u = 7$





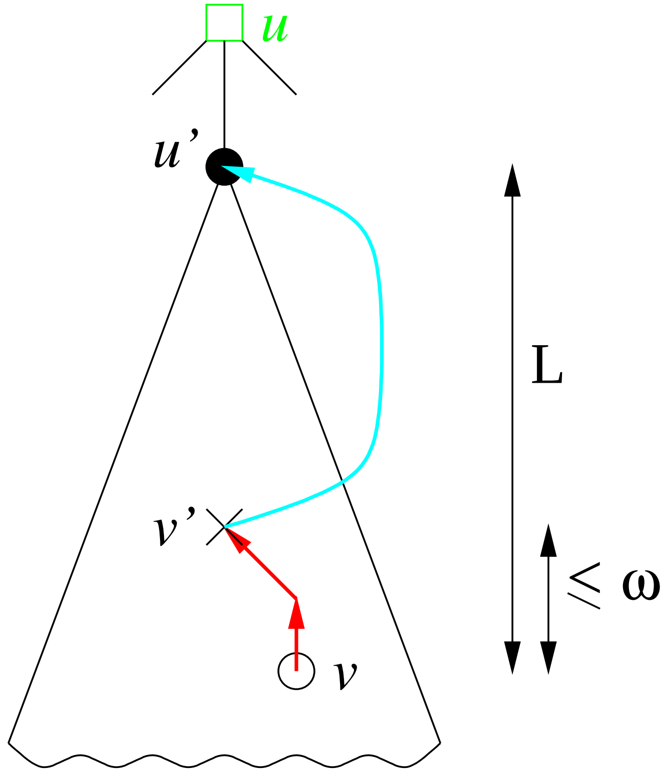
Parenthesis Coding - Compact Data Structures



Matching parenthesis in $O(1)$ time with $o(n)$ extra bits.
Rank($\cdot$) and Position($\cdot$) in $O(1)$ time with $o(n)$ extra bits.

Problem: Routing query time $=$ Length of route $= \Omega(n)$.

ω-Dominating Sets for $O(\log^{1+\epsilon})$ Query Time



We store the solution ($O(\log n)$ bits) for all the nodes in the ω-dominating set, i.e., for at most n/ω nodes. Choosing $\omega = \log^{1+\epsilon} n$, this represents $o(n)$ bit size table and $O(\omega)$ query time.

Hence,

$O(n \log k)$ bits/node with $O(\log^{1+\epsilon} n)$ query time,
$k =$#page(graph).

- $8n + o(n)$ bits/node for planar graphs;

- $n \log(g+1) + O(n)$ for genus g graphs ($k \leqslant O(\sqrt{g})$);

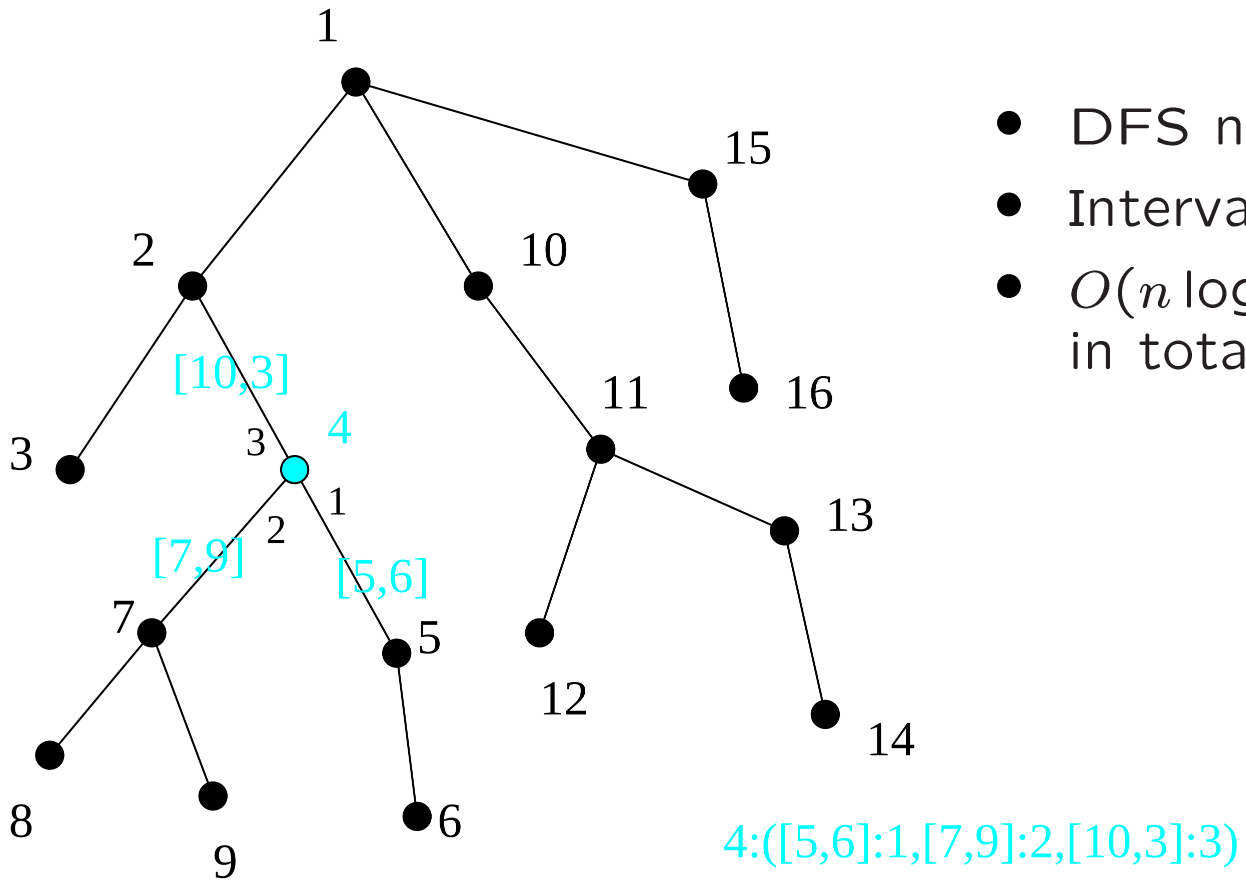
- holds even if G has $o(n/\log n)$ crossing edges.

Open problem: $o(n)$ bits for every planar graphs? $O(\sqrt{n})$?

Routing in a Tree (a basic problem)

Planar $\Rightarrow$ $8n$ bits/node

But,



- DFS numbering
- Interval Routing
- $O(n \log n)$ bits in total!

4:([5,6]:1,[7,9]:2,[10,3]:3)

Binary search $\Rightarrow$ routing in $O(\log n)$ time

Space $\Rightarrow$ $O(d \log n)$ bits, d = degree

Problem: Space $\Omega(n)$ whenever $d > n/\log n$

y $\downarrow$ (position 9)

Solution: 4:

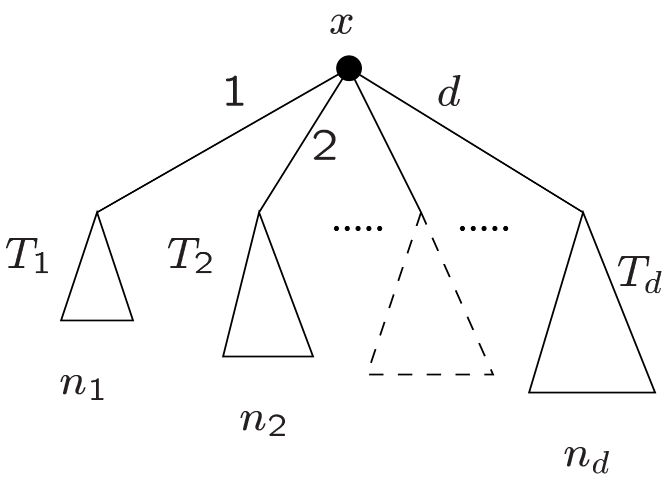
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
4:	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
					#1		#2			#3						

Routing: compute the number of 1's (mod d) before position y.

Space n bits, but $\Omega(n)$ query time! (In fact, we can do in $n + o(n)$ space and $O(1)$ query time using Position($\cdot$)).

Can we do better?

Lower bound: $\Omega(\sqrt{n})$ bits for at least one node.



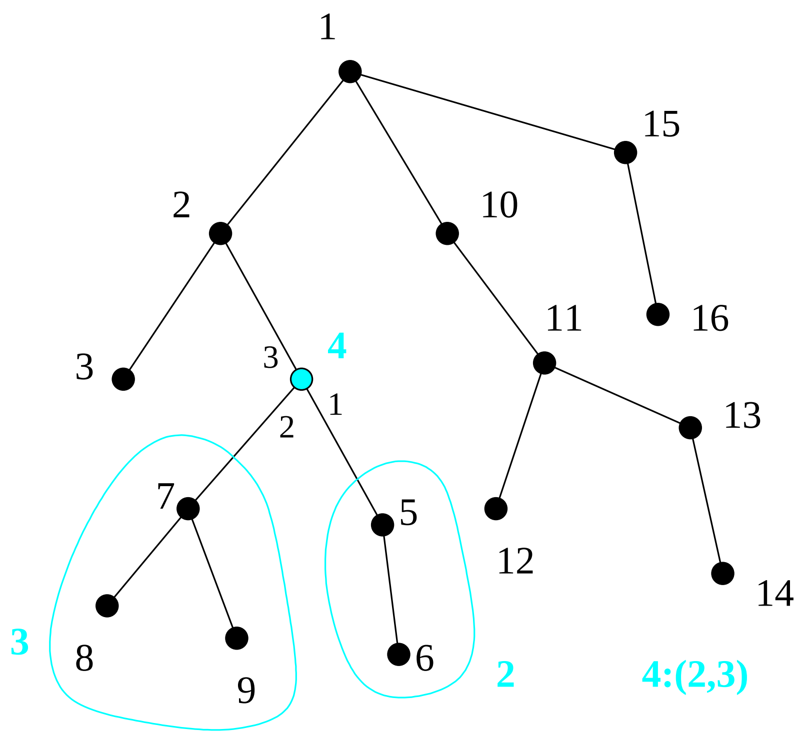
- test in x its routing function to all destinations $y \in \{1, \ldots, n\} \setminus \{x\}$
- sort answers by size $n_i = |T_i|$, $n_1 \leqslant \cdots \leqslant n_d$

$\sum_{i=1}^{d} n_i = n - 1$ and $1 \leqslant n_1 \leqslant \cdots \leqslant n_d$.

x "computes" $(n_1, \ldots, n_d)$ knowing n and x.

Number of partitions of n is $\sim 2^{c\sqrt{n}}$ $\Rightarrow$ $c\sqrt{n}$ bits for x.

Can we do better? (cont'd): YES



- DFS with respect to the number of descendants

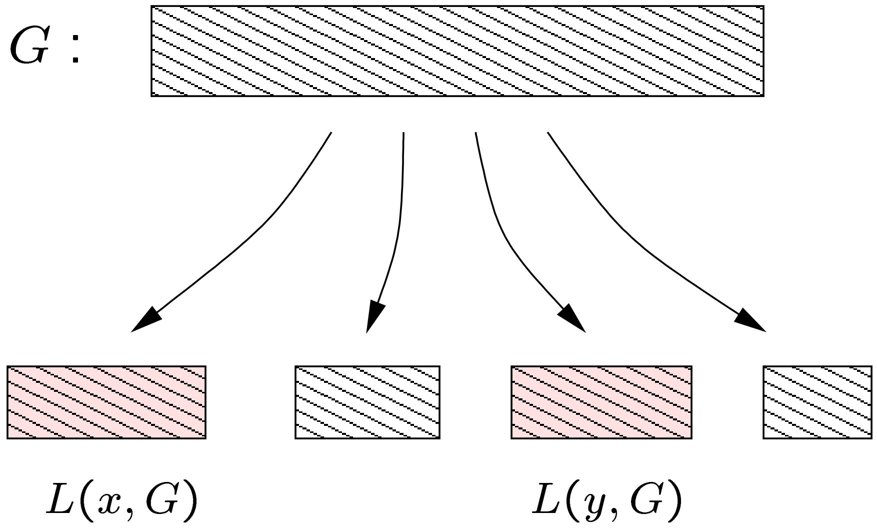
$4:(2,3) \Rightarrow$ theoretically, $O(\sqrt{n})$ bits storage.

$y \in]4, 4+2]$? $y \in]6, 6+3]$? if not route on 3.

Problem: Query time? What happens if addresses are longer than $\log n$ bits?

3. Distance Labeling

(a selection of results)



Query function f: $f(L(x,G), L(y,G)) = \mathrm{dist}_G(x,y)$

Problem: Find a pair $\langle L, f \rangle$ that provides short labels, and allows fast distance queries (small time complexity of f).

General Bounds

$O(n \log n)$ bit labels, $O(1)$ query time

	node	1	2	...	n
$L(u,G)$:	$\mathrm{dist}_G(u,\cdot)$	d_1	d_2	...	

$\longleftrightarrow$ $n \log n$ bits

Actually, [Winkler'84], Graham-Pollack Conjecture
$\Rightarrow$ $1.58n$ bit labels and $O(n)$ query time.

[Gavoille-Peleg-Pérennès-Raz'00]
$\Rightarrow$ $11n$ bit labels and $O(\log \log n)$ query time.

Lower Bounds

Let $\mathcal{F}_n$ be a family of n-node graphs (node labeled by $\{1,\ldots,n\}$), and $A \subseteq \{1,\ldots,n\}$. $\mathcal{F}_n$ is an A-family if $\forall\, G,H \in \mathcal{F}_n$, $\exists\, x,y \in A$ such that $\mathrm{dist}_G(x,y) \neq \mathrm{dist}_H(x,y)$. (Refinable).

Theorem. *Any $\langle L,f\rangle$ on an A-family $\mathcal{F}_n$ requires some labels of size*

$$\frac{1}{|A|} \cdot \log|\mathcal{F}_n| \ \text{bits.}$$

Proof. Sequence of labels: $a_i \in A$

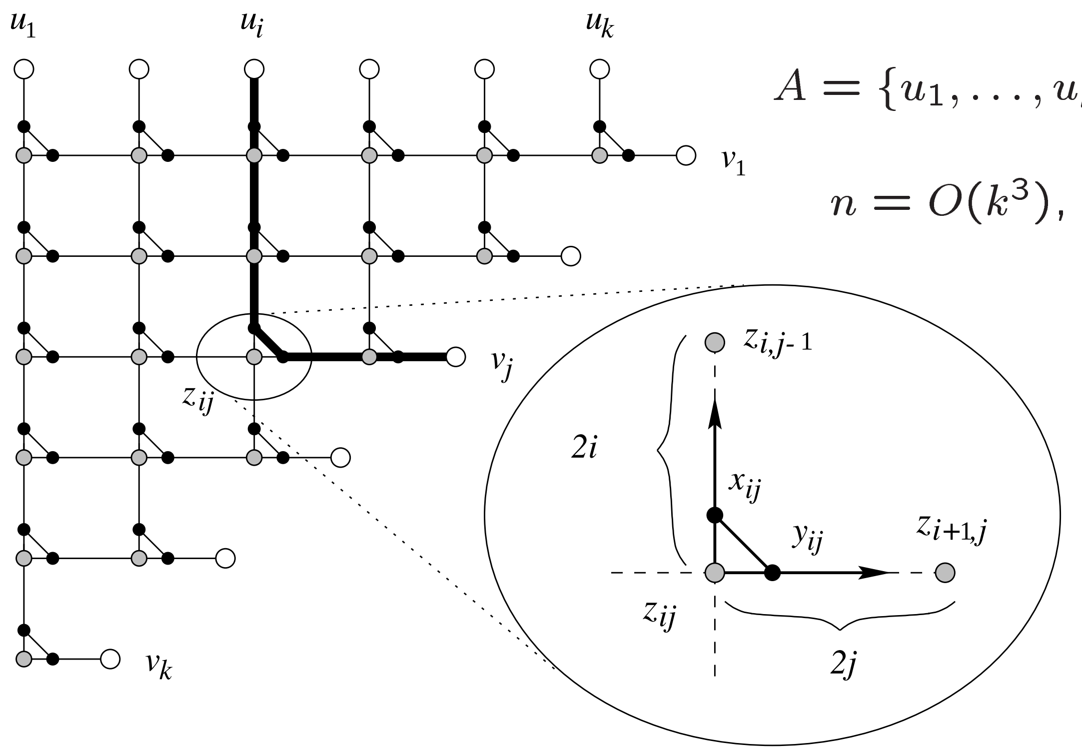
$$\begin{aligned} S(G) &= \langle L(a_1,G), L(a_2,G),\ldots)\rangle \\ S(H) &= \langle L(a_1,H), L(a_2,H),\ldots)\rangle \end{aligned}$$

$G \neq H \Rightarrow S(G) \neq S(H) \Rightarrow |\mathcal{F}_n|$ sequences
$\Rightarrow \exists a \in A$, $\exists G \in \mathcal{F}_n$, $|L(a,G)| \geqslant (\log|\mathcal{F}_n|)/|A|$. $\square$

Applications

- $\Omega(n)$ bit labels for general graphs (A = set of all the nodes)
- $\Omega(n^{1/3})$ bit labels for planar graphs

$|\mathcal{F}_n| = 2^{\Theta(k^2)}$

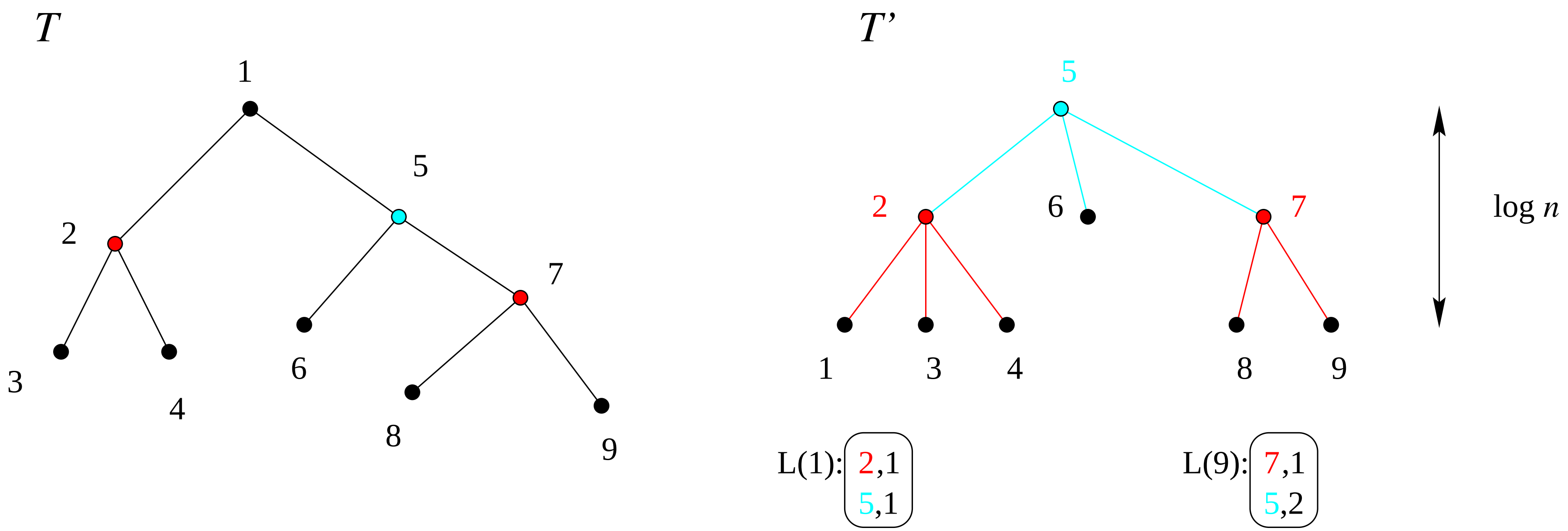


$A = \{u_1, \ldots, u_k, v_1, \ldots, v_k\}$

$n = O(k^3)$, $|A| = 2k$

- $\Omega(t(n))$ for graphs of treewidth bounded by $t(n)$ ...

Trees and Generalization



$2\log^2 n$ bit labels and $O(\log n)$ query time.

[Query time: $O(\log\log n)$, and $O(1)$ if allowing integer multiplication and division.]

Lower Bound for Trees?

$\mathcal{T}_n$ = trees with node set $\{1, \ldots, n\}$

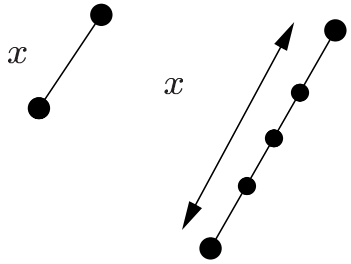
$|\mathcal{T}_n| = n^{n-2}$, A-family, $A = \{1, \ldots, n\}$

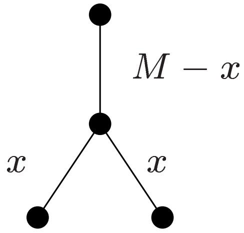
$$\Rightarrow \quad \frac{1}{n} \cdot \log |\mathcal{T}_n| \;=\; \log n - O(1) \;.$$

Lower Bound for Trees (cont'd)

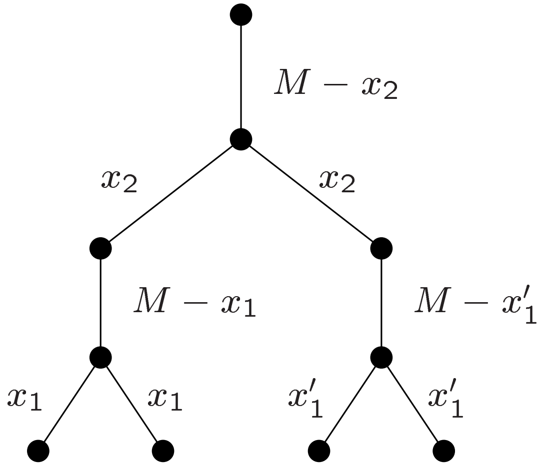
Family $\mathcal{T}_{h,M}$

$$M \sim \sqrt{n}$$
$$h \sim \log n$$





a $\mathcal{T}_{1,M}$ tree



#total of labels
for all binary trees
is at least $M^{h/2}$

$\mathcal{T}_{h,M}$

	1	2	3	...	n
T_1	ℓ_1	ℓ_2	ℓ_3	...	
T_2	ℓ_i	ℓ_1	ℓ_j	...	
T_3	ℓ_4	ℓ_i	ℓ_k	...	
$\vdots$		...	ℓ_t	...	

$$\Rightarrow \exists\ \ell_t \text{ of size } \frac{h}{2}\log M \geqslant \frac{1}{8}\log^2 n \text{ bits.}$$

Other Results

- Approximation of the distance

$$f(L(x,G),L(y,G)) \in [\mathrm{dist}_G(x,y),(1+\epsilon)\mathrm{dist}_G(x,y)]$$

  Trees (and bounded treewidth): $\Theta(\log n \cdot \log\log n)$ (optimal) for $\epsilon = 1/\log n \to 0$.

- $\exists$ graphs with possibly short labels ($3\log n$), but such that if labels are shorter than $n/2$ bits, then the query time is necessarily greater than

$$2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

  (bounded resource Turing Machine)

4. Perspectives

1. Generalization

2. New Applications

3. Fundamental Question

Generalization

Let π be a function defined on subset of nodes of a graph.

Ex: $\pi(x, y, G) = \mathrm{dist}_G(x, y)$.

$\langle L, f \rangle$ is a π-labeling for G if

$$f(L(x, G), L(y, G)) \; = \; \pi(x, y, G)$$

Ex: $L(x, G) = \langle \mathrm{address}(x), \mathrm{routing\_table}(x) \rangle$

Complexity Criteria

- Size of distributed data structure

$$\text{size}(\pi, G) \;=\; \max_{u \in V(G)} |L(u, G)|$$

- Query time to solve π from labels
  (worst-case time complexity of f)

Questions on π-Labeling Schemes

Complexity of size(π, G) for:

1. Classes of graphs: planar, chordal graphs, Cayley, bounded treewidth graphs, AT-free graphs, ...

2. Classes of problems: π of arity 2, $\pi \in$ predicate of a given logic (first order, ...), $\pi \in$ P, $\pi \in$ NP, ...

$\Rightarrow$ meta-theorems

Example: $\pi = \mathsf{dist}$

Meta-Theorem. *For every enumerable A-family $\mathcal{F}_n$ such that*

$$\frac{1}{|A|} \cdot \log |\mathcal{F}_n| \;\; \geqslant \;\; 3 \log n$$

$T_n(G) = \mathsf{size}(\mathsf{dist}, G)$ *is undecidable.*

Proof

Assume the reverse, $T_n(G) = \text{size}(\text{dist}, G)$.

T_n is a Turing Machine, a constant size program.

Let P_n the program:

```
1. i := 0
2. do
     i := i + 1
     G := Graph(F_n, i)
   while T_n(G) < 3 log n
3. Return i.
```

Let $G^* = \text{Graph}(\mathcal{F}_n, i)$.

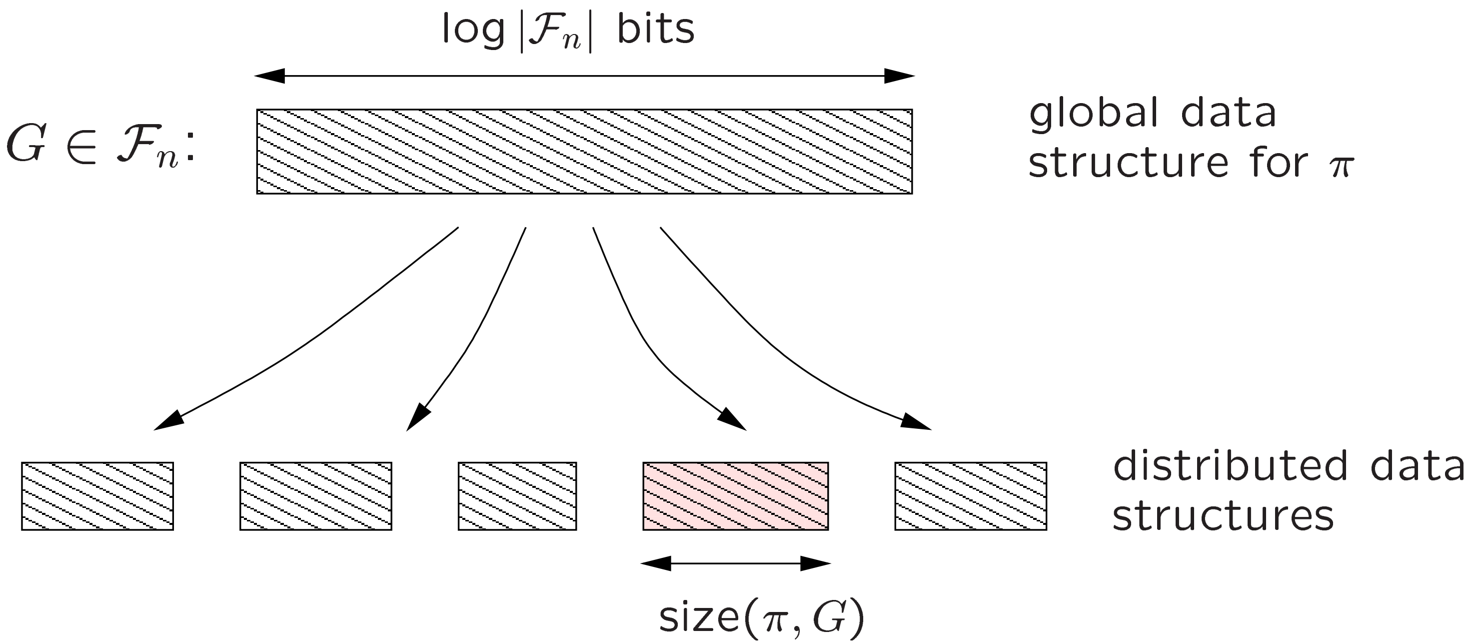
Define for every $G \neq G^*$:
$L(x, G) = \langle 0, x, \text{dist}_G(x, \cdot)\rangle$, and
$L(x, G^*) = \langle 1, x, P_n, T_n, \text{Graph}(\mathcal{F}_n, \cdot)\rangle$.
Given $L(x, G), L(y, G)$ one can compute $\text{dist}_G(x, y)$.
But, $|L(x, G^*)| \leqslant 2 \log n$. Contradiction. $\square$

New Applications

Find π that are interesting to solve.

- dist(x, y) for network setup connections
- adj$(x, y) = 1$ iff $(x, y) \in E$
- route$(x, y) =$ memory free routing
- sp$(x, y, z) = 1$ iff $z \in$ shortest path between x and y

Fundamental Question



$$\text{red}(\pi, \mathcal{F}_n) \;=\; \max_{G \in \mathcal{F}_n} \left\{ \frac{n \cdot \text{size}(\pi, G)}{\log|\mathcal{F}_n|} \right\} \quad (\text{redundancy})$$

$$\text{red}(\pi, \mathcal{F}_n) \;=\; O(1) \cdots O(n)\ .$$

- Classification of graph families $\mathcal{F}_n$

$$\mathsf{red}(\mathsf{dist}, \mathcal{T}_n) = \frac{n \cdot \Theta(\log^2 n)}{n \log n} = \Theta(\log n)$$

$$\mathsf{red}(\mathsf{dist}, \mathcal{G}_n) = \frac{n \cdot \Theta(n)}{\Theta(n^2)} \leqslant 3.16 \quad (\mathcal{G}_n\text{=all the graphs})$$

- Classification of problems π

$$\mathsf{red}(\mathsf{dist}, \mathcal{T}_n) = \Theta(\log n)$$
$$\mathsf{red}(\mathsf{adj}, \mathcal{T}_n) \leqslant 2$$