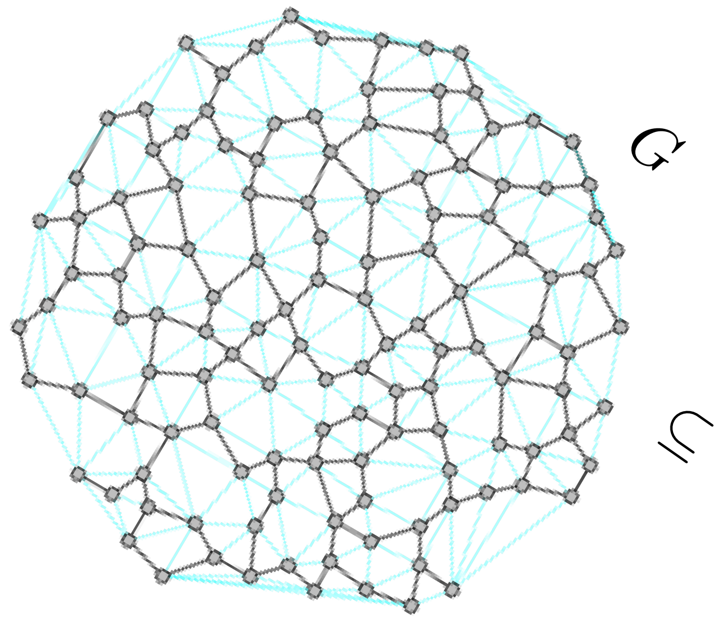
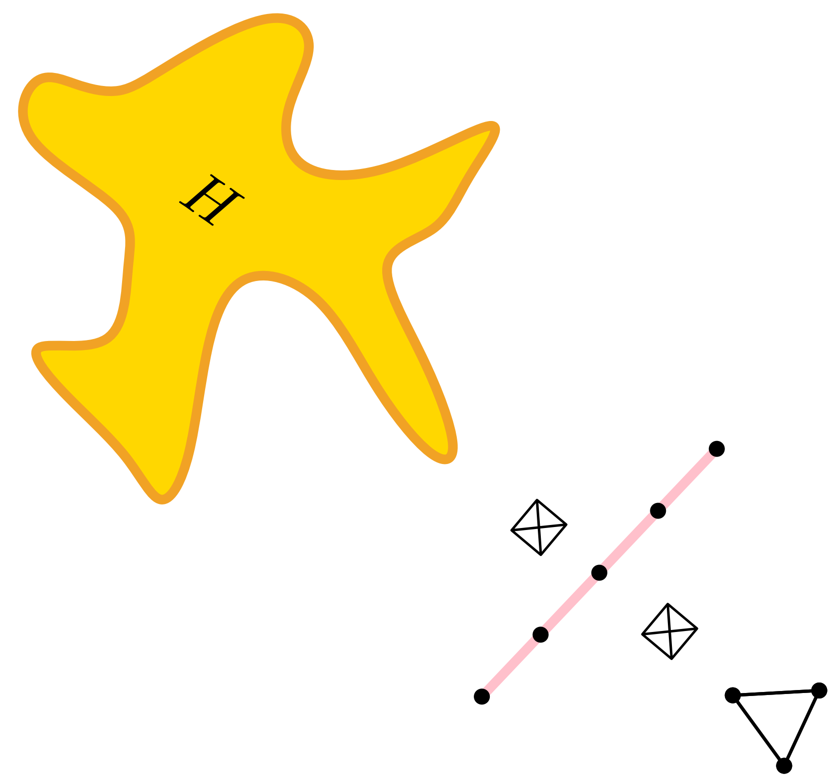
Product structure of planar graphs





Piotr Micek
Jagiellonian University

tutorial presentation for
9th Polish Combinatorial Conference
Będlewo, September 20, 2022

plan of tutorial

Part I	statements
	background
	proof
	variants / generalizations
Part II	quick applications
Part III	application: adjacency labelling scheme
	open problems / further research

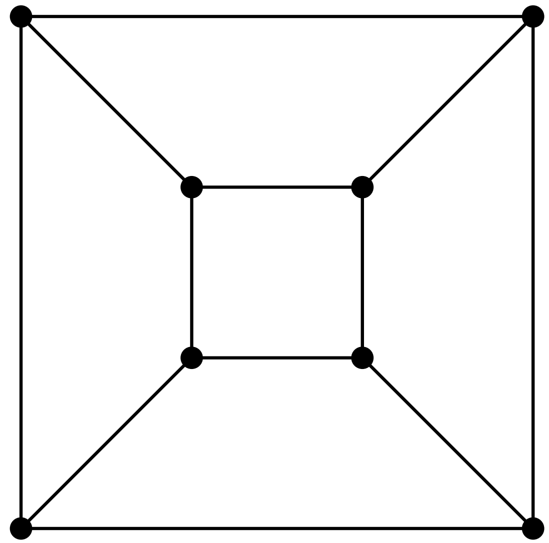
adjacency tester $A : (\{0,1\}^*)^2 \to \{0,1\}$

labelling function $\ell : V(G) \to \{0,1\}^*$

(G, ℓ) works with A if

$$A(\ell(v), \ell(w)) = \begin{cases} 0 & \text{if } vw \notin E(G) \\ 1 & \text{if } vw \in E(G) \end{cases}$$

G



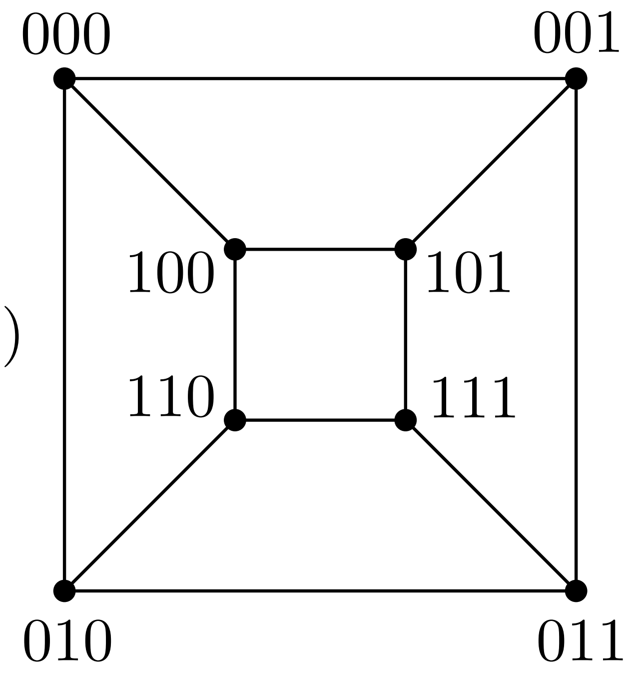
adjacency tester $A: (\{0,1\}^*)^2 \rightarrow \{0,1\}$

labelling function $\ell: V(G) \rightarrow \{0,1\}^*$

(G,ℓ) works with A if

$$A(\ell(v),\ell(w)) = \begin{cases} 0 & \text{if } vw \notin E(G) \\ 1 & \text{if } vw \in E(G) \end{cases}$$

(G,ℓ)

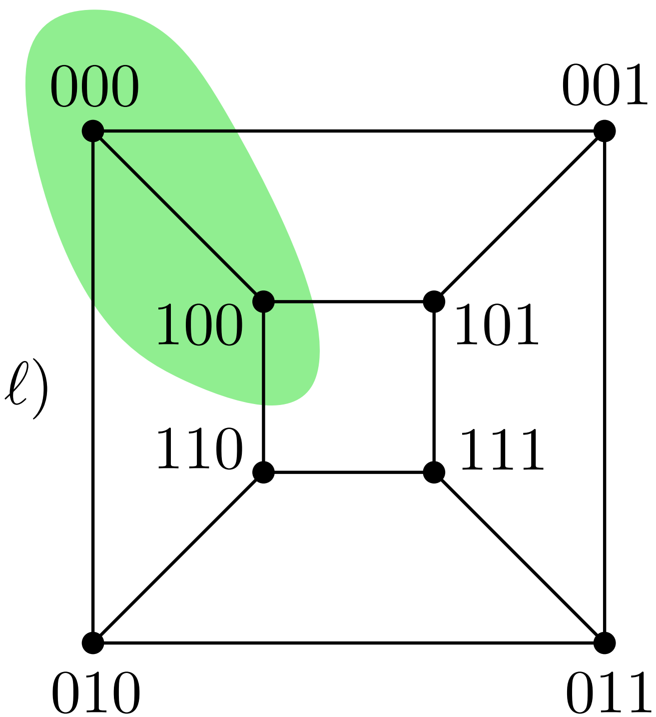


adjacency tester $\quad A: \; (\{0,1\}^*)^2 \; \rightarrow \; \{0,1\}$

labelling function $\quad \ell: \; V(G) \; \rightarrow \; \{0,1\}^*$

(G,ℓ) works with A if

$$A(\ell(v),\ell(w)) = \begin{cases} 0 & \text{if } vw \notin E(G) \\ 1 & \text{if } vw \in E(G) \end{cases}$$



(G,ℓ)

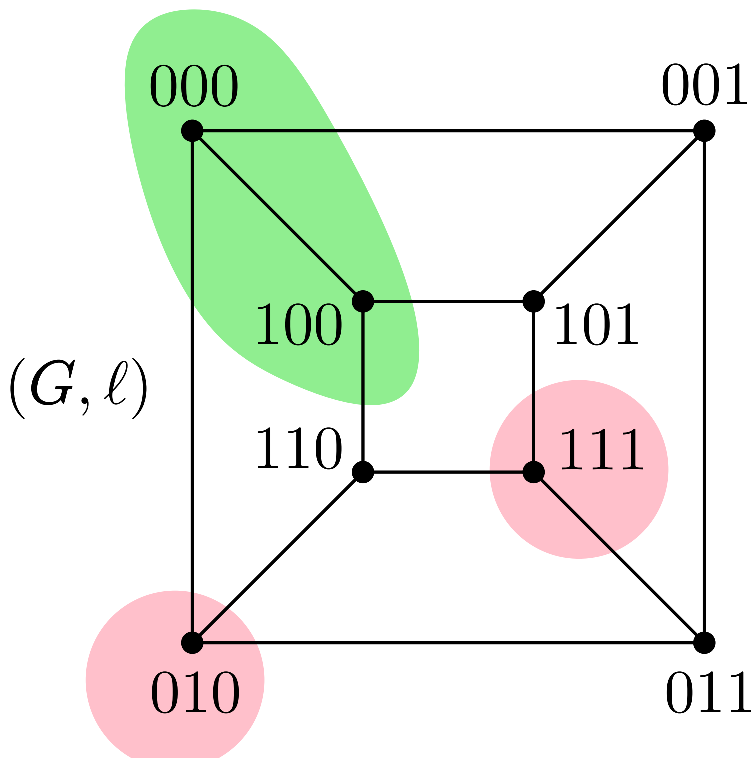
$A(000,100) = 1$

adjacency tester $A : (\{0,1\}^*)^2 \to \{0,1\}$

labelling function $\ell : V(G) \to \{0,1\}^*$

(G,ℓ) works with A if

$$A(\ell(v),\ell(w)) = \begin{cases} 0 & \text{if } vw \notin E(G) \\ 1 & \text{if } vw \in E(G) \end{cases}$$



$A(000,100) = 1$

$A(010,111) = 0$

A family of graphs $\mathcal{G}$ has an $f(n)$-**bit adjacency labelling scheme**

if $\exists$ a function $A : (\{0,1\}^*)^2 \to \{0,1\}$ such that

$\forall$ n-vertex graph $G \in \mathcal{G}$ $\exists$ $\ell : V(G) \to \{0,1\}^*$ such that

- $|\ell(v)| \leqslant f(n)$ for each v in G
- (G, ℓ) works with A

(Dujmović, Joret, Morin, PM, Ueckerdt, Wood 2019)

A family of graphs $\mathcal{G}$ has an $f(n)$**-bit adjacency labelling scheme**

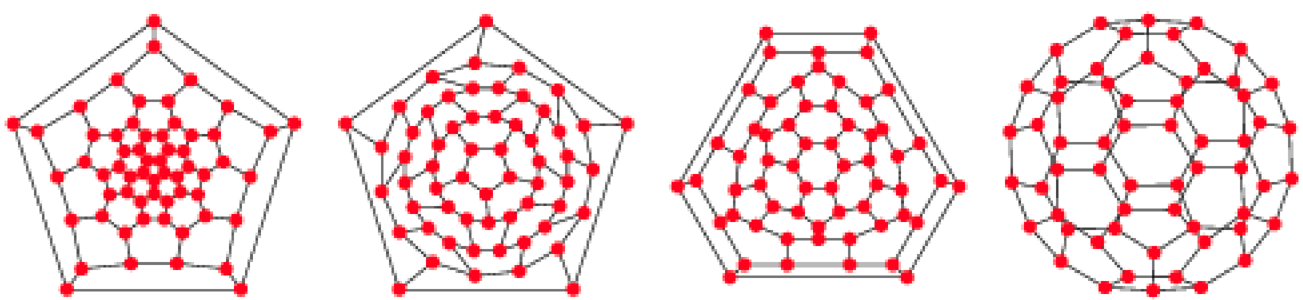
if $\exists$ a function $A : (\{0,1\}^*)^2 \to \{0,1\}$ such that

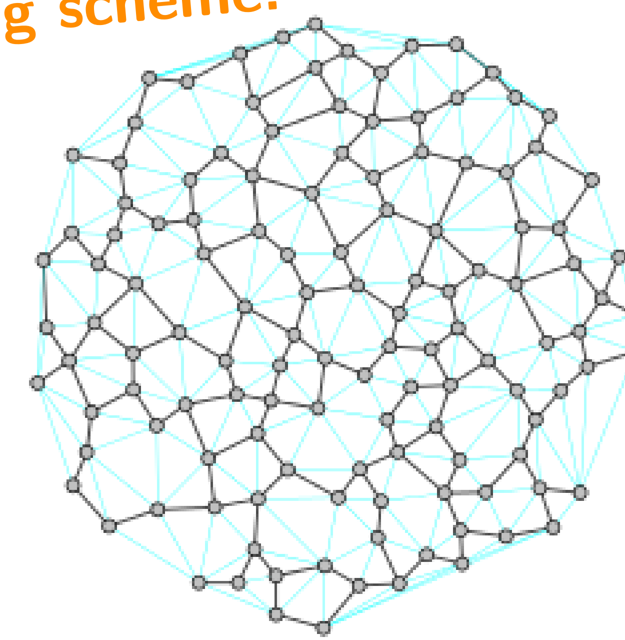
$\forall$ n-vertex graph $G \in \mathcal{G}$ $\exists$ $\ell : V(G) \to \{0,1\}^*$ such that

- $|\ell(v)| \leqslant f(n)$ for each v in G
- (G, ℓ) works with A

(Dujmović, Joret, Morin, PM, Ueckerdt, Wood 2019)

The family of **planar graphs** has a $(1 + o(1)) \log n$**-bit adjacency labelling scheme.**





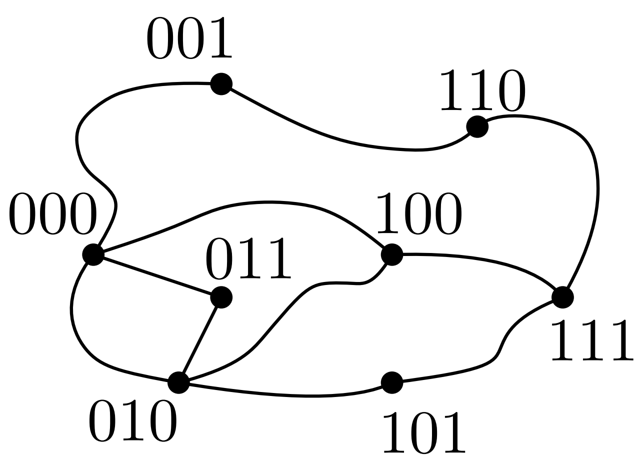
Examples

▷ when $\mathcal{G}$ contains a single n-vertex graph

labels $\equiv$ unique ids of length $\lceil \log n \rceil$

function $A \equiv$ adjacency matrix

G



A	000	001	010	011	100	101	$\cdots$
000	0	1	1	0	1	0	
001	1	0	0	0	0	0	
010	1	0	0	1	1	0	
011	0	0	1	0	0	0	
100	1	0	0	1	0	0	
101	0	1	0	0	0	0	
$\vdots$							$\ddots$

Examples

- ▷ when $\mathcal{G}$ contains a single n-vertex graph
  labels $\equiv$ unique ids of length $\lceil \log n \rceil$
  function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

Examples

▷ when $\mathcal{G}$ contains a single n-vertex graph

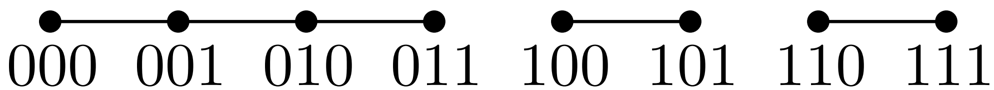
labels $\equiv$ unique ids of length $\lceil \log n \rceil$

function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

▷ when $\mathcal{G}$ is a family of linear forests

labels $\equiv$ unique ids assigned along the paths



plus an extra bit indicating . . .

Examples

▷ when $\mathcal{G}$ contains a single n-vertex graph

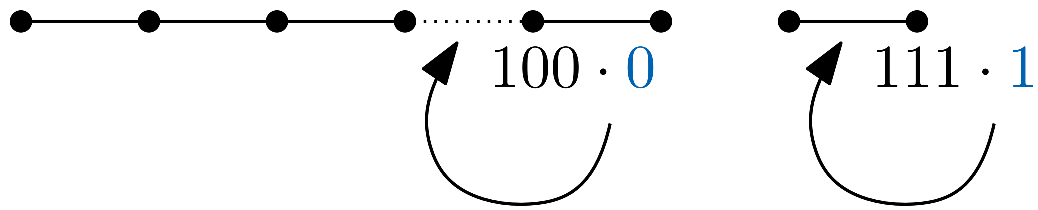
labels $\equiv$ unique ids of length $\lceil \log n \rceil$

function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

▷ when $\mathcal{G}$ is a family of linear forests

labels $\equiv$ unique ids assigned along the paths



plus an extra bit indicating ...

if a vertex is adjacent to a vertex to the left

$\log n + \mathcal{O}(1)$ scheme

Examples

▷ when $\mathcal{G}$ contains a single n-vertex graph

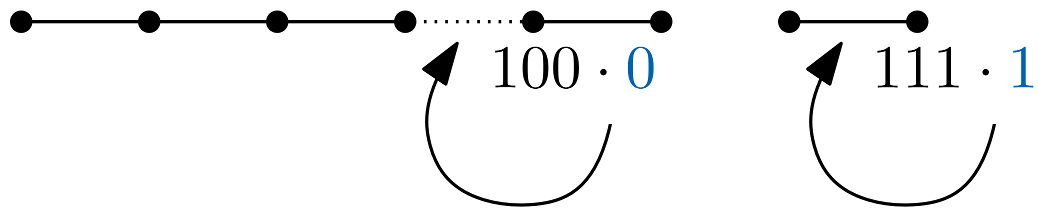
labels $\equiv$ unique ids of length $\lceil \log n \rceil$

function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

▷ when $\mathcal{G}$ is a family of linear forests

labels $\equiv$ unique ids assigned along the paths



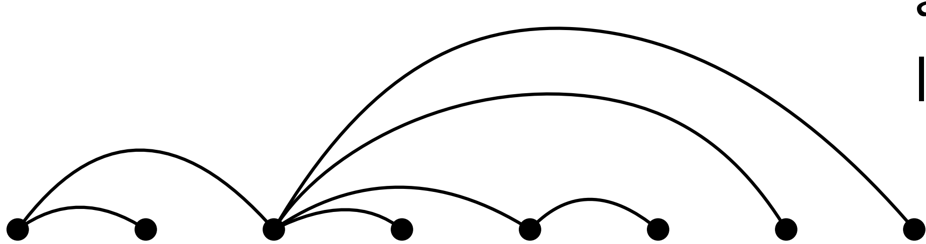
plus an extra bit indicating . . .

if a vertex is adjacent to a vertex to the left

$\log n + \mathcal{O}(1)$ scheme

▷ when $\mathcal{G}$ is a family of **trees**

root a tree and take any topological ordering of its vertices



assign unique ids

labels $\equiv$ concatenation of vertex id and id of its parent

$2\lceil \log n \rceil$ scheme

Examples

▷ when $\mathcal{G}$ contains a single n-vertex graph

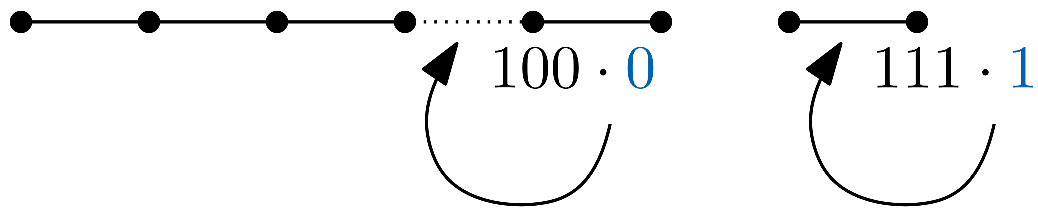
labels $\equiv$ unique ids of length $\lceil \log n \rceil$

function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

▷ when $\mathcal{G}$ is a family of linear forests

labels $\equiv$ unique ids assigned along the paths



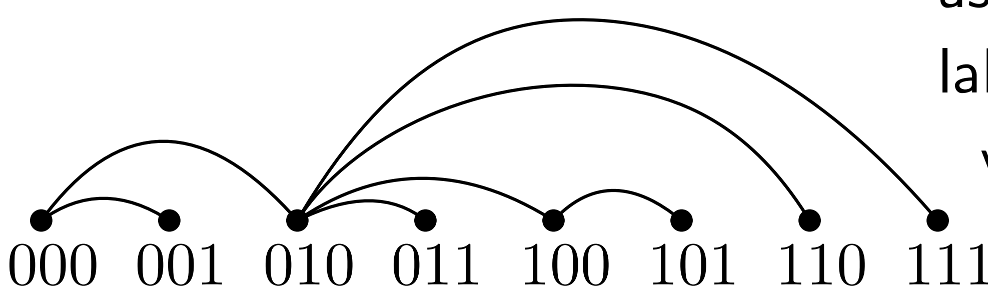
plus an extra bit indicating . . .

if a vertex is adjacent to a vertex to the left

$\log n + \mathcal{O}(1)$ scheme

▷ when $\mathcal{G}$ is a family of **trees**

root a tree and take any topological ordering of its vertices



assign unique ids

labels $\equiv$ concatenation of vertex id and id of its parent

$2\lceil \log n \rceil$ scheme

Examples

- when $\mathcal{G}$ contains a single n-vertex graph

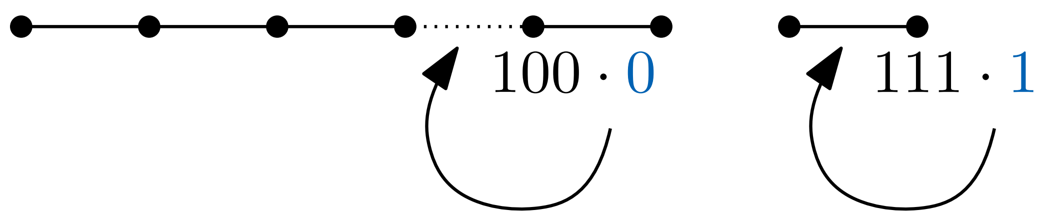
  labels $\equiv$ unique ids of length $\lceil \log n \rceil$

  function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

- when $\mathcal{G}$ is a family of linear forests

  labels $\equiv$ unique ids assigned along the paths



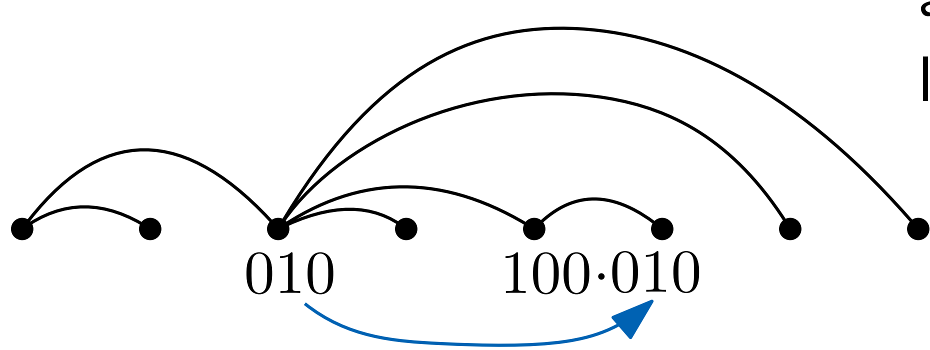
plus an extra bit indicating ...

if a vertex is adjacent to a vertex to the left

$\log n + \mathcal{O}(1)$ scheme

- when $\mathcal{G}$ is a family of **trees**

  root a tree and take any topological ordering of its vertices



assign unique ids

labels $\equiv$ concatenation of vertex id and id of its parent

$2\lceil \log n \rceil$ scheme

Examples

▷ when $\mathcal{G}$ contains a single n-vertex graph

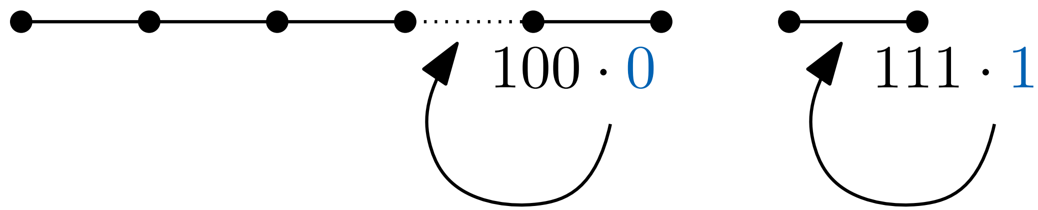
labels $\equiv$ unique ids of length $\lceil \log n \rceil$

function $A \equiv$ adjacency matrix

you cannot do better than $\lceil \log n \rceil$

▷ when $\mathcal{G}$ is a family of linear forests

labels $\equiv$ unique ids assigned along the paths



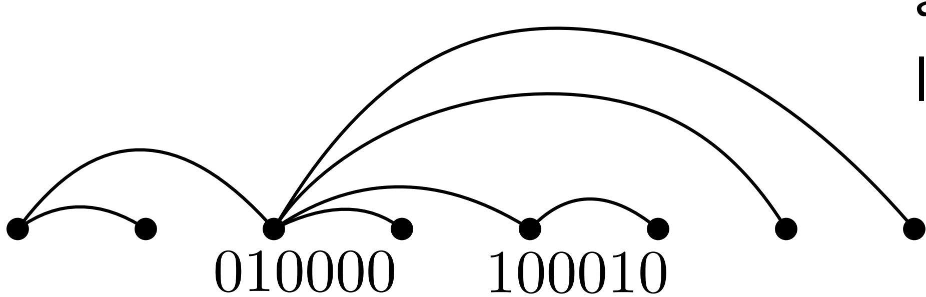
plus an extra bit indicating . . .

if a vertex is adjacent to a vertex to the left

$\log n + \mathcal{O}(1)$ scheme

▷ when $\mathcal{G}$ is a family of **trees**

root a tree and take any topological ordering of its vertices



assign unique ids

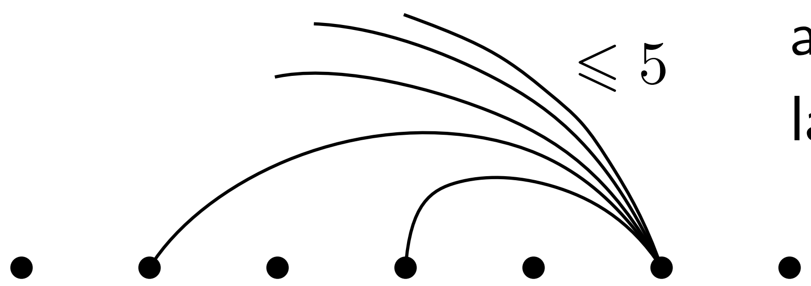
labels $\equiv$ concatenation of vertex id and id of its parent

$2\lceil \log n \rceil$ scheme

Examples

▷ when $\mathcal{G}$ is a family of **planar graphs**

take a vertex ordering witnessing that G is 5-degenerate



assign unique ids

labels $\equiv$ concatenation of

vertex id and ids of left neighbors

$6\lceil \log n \rceil$ scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log\log n)$-bit scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log \log n)$-bit scheme

(Alstrup, Rauhe 2006)

$\log n + \mathcal{O}(\log^* n)$-bit scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log \log n)$-bit scheme

(Alstrup, Rauhe 2006)

$\log n + \mathcal{O}(\log^* n)$-bit scheme

(Alstrup, Dahlgaard, Knudsen 2017)

$\log n + \mathcal{O}(1)$-bit scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log \log n)$-bit scheme

(Alstrup, Rauhe 2006)

$\log n + \mathcal{O}(\log^* n)$-bit scheme

(Alstrup, Dahlgaard, Knudsen 2017)

$\log n + \mathcal{O}(1)$-bit scheme

$\Longrightarrow$

planar graphs have arboricity 3

$3 \log n + \mathcal{O}(1)$-bit scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log\log n)$-bit scheme

(Alstrup, Rauhe 2006)

$\log n + \mathcal{O}(\log^* n)$-bit scheme

(Alstrup, Dahlgaard, Knudsen 2017)

$\log n + \mathcal{O}(1)$-bit scheme

Bounded treewidth graphs

(Gavoille, Labourel 2007)

$(1 + o(1)) \log n$-bit scheme

$\Longrightarrow$

planar graphs have arboricity 3

$3 \log n + \mathcal{O}(1)$-bit scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log\log n)$-bit scheme

(Alstrup, Rauhe 2006)

$\log n + \mathcal{O}(\log^* n)$-bit scheme

(Alstrup, Dahlgaard, Knudsen 2017)

$\log n + \mathcal{O}(1)$-bit scheme

$\Longrightarrow$ **planar graphs** have arboricity 3 $3\log n + \mathcal{O}(1)$-bit scheme

Bounded treewidth graphs

(Gavoille, Labourel 2007)

$(1 + o(1))\log n$-bit scheme

every graph with no K_t-**minor** can be edge 2-colored so that each monochromatic subgraph has bounded tw $\Longrightarrow$ $(2 + o(1))\log n$-bit scheme

history and related work

Forests

(Chung 1990)

$\log n + \mathcal{O}(\log\log n)$-bit scheme

(Alstrup, Rauhe 2006)

$\log n + \mathcal{O}(\log^* n)$-bit scheme

(Alstrup, Dahlgaard, Knudsen 2017)

$\log n + \mathcal{O}(1)$-bit scheme

$\Longrightarrow$ **planar graphs** have arboricity 3 $3\log n + \mathcal{O}(1)$-bit scheme

Bounded treewidth graphs

(Gavoille, Labourel 2007)

$(1 + o(1))\log n$-bit scheme

every graph with no K_t-**minor** can be edge 2-colored so that each monochromatic subgraph has bounded tw $\Longrightarrow$ $(2 + o(1))\log n$-bit scheme

Planar graphs

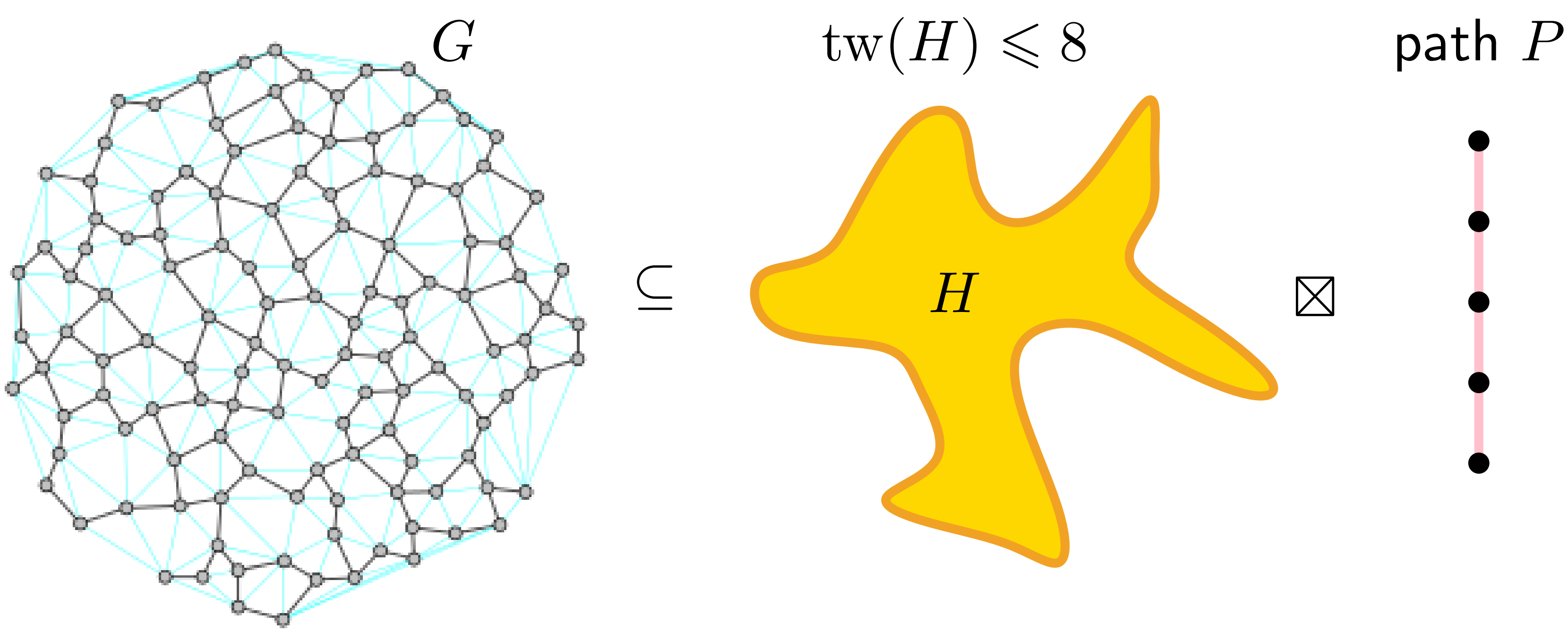
(Bonamy, Gavoille, Mi. Pilipczuk 2020)

$\left(\frac{4}{3} + o(1)\right)\log n$-bit scheme

product structure theorem

(Dujmović, Joret, Morin, PM, Ueckerdt, Wood 2020)

Every **planar graph** G is a subgraph of a **strong product** $H \boxtimes P$ where H is a graph of treewidth at most 8 and P is a path.

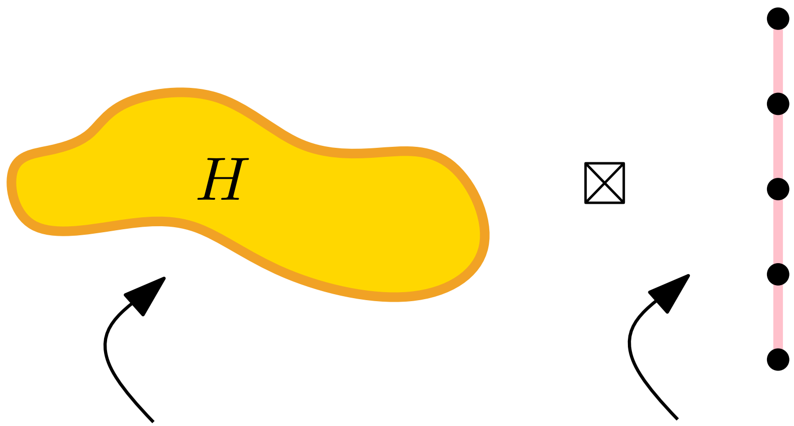


labelling scheme through the Product Structure Thm

$\mathrm{tw}(H) \leqslant 8$ path P

n vertices m vertices h vertices

$G \subseteq H \boxtimes P$



$(1+o(1))\log m$-bit $(1+o(1))\log h$-bit schemes

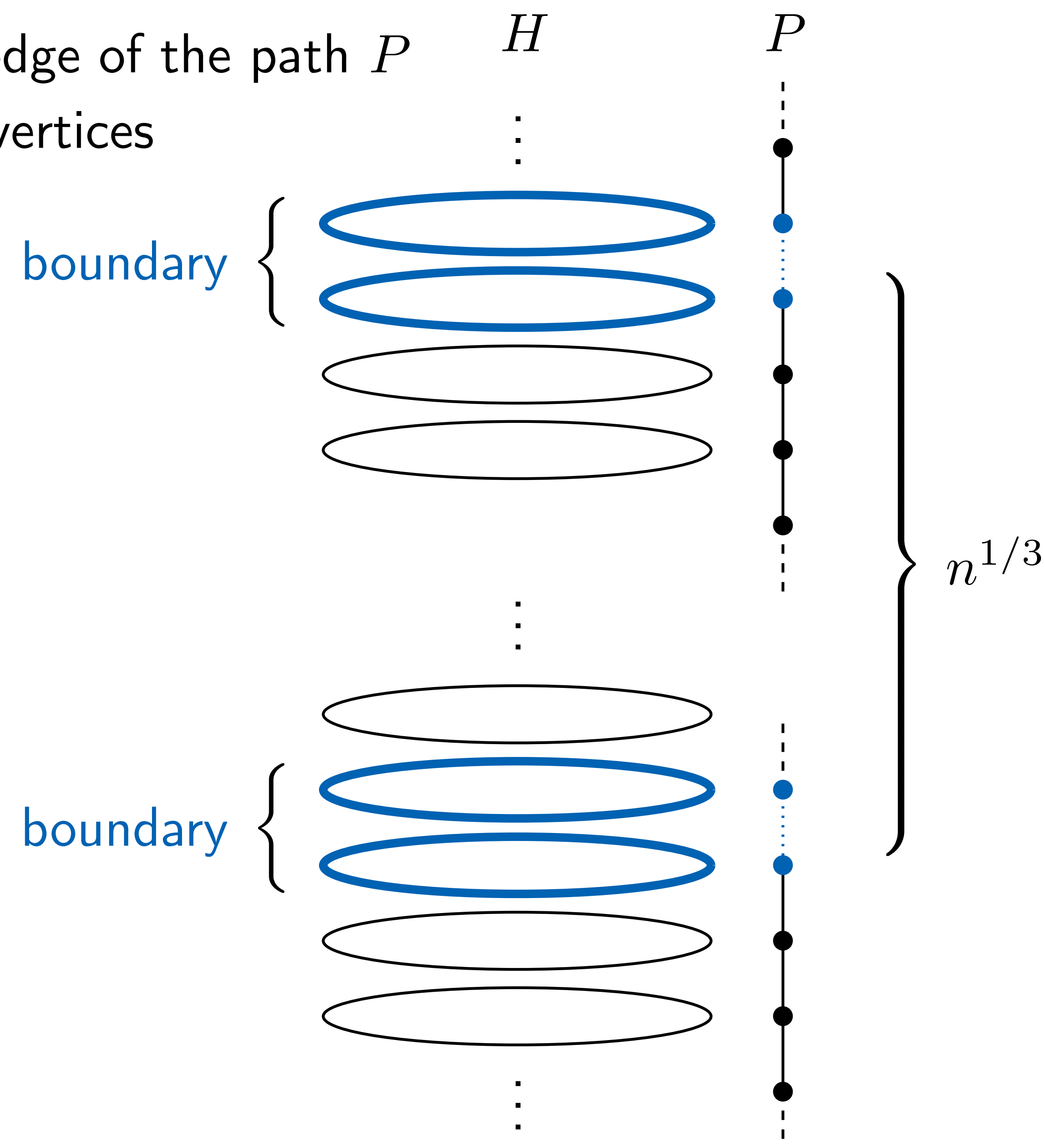
$(1+o(1))\log(m \cdot h)$ plus $8 \cdot 3$ bits to check if edge in $H \boxtimes P$ is present in G

all we can say is $m \leqslant n$ and $h \leqslant n$, so we get

$$(1+o(1))\log(n \cdot n)\text{-bit}$$
$$|||$$
$$(2+o(1))\log(n)\textbf{-bit}$$

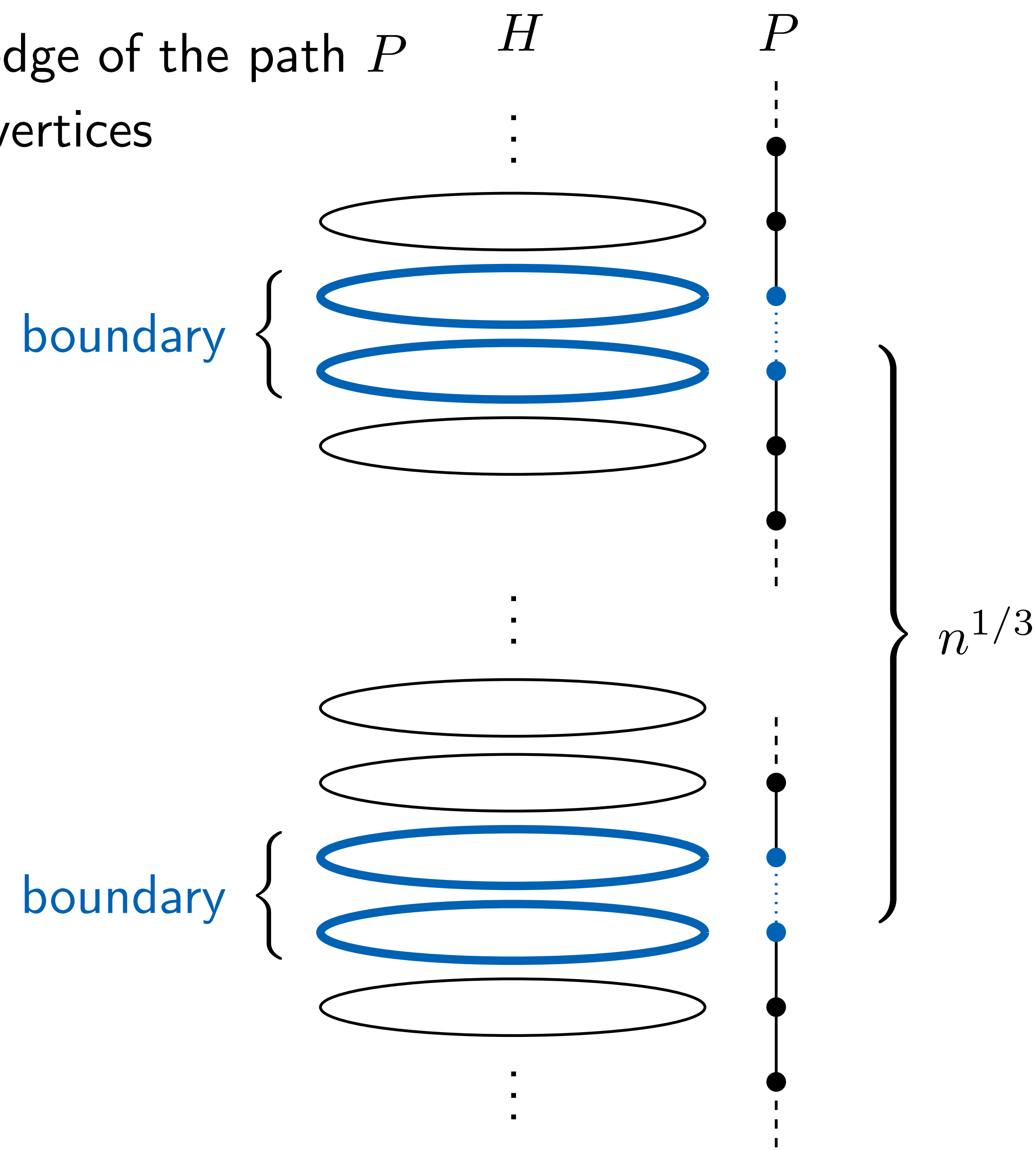
$\left(\frac{4}{3}+o(1)\right)\log n$-**bit scheme**

- "remove" every other $n^{1/3}$ edge of the path P so that there are $\mathcal{O}(n^{2/3})$ vertices in **boundary layers**



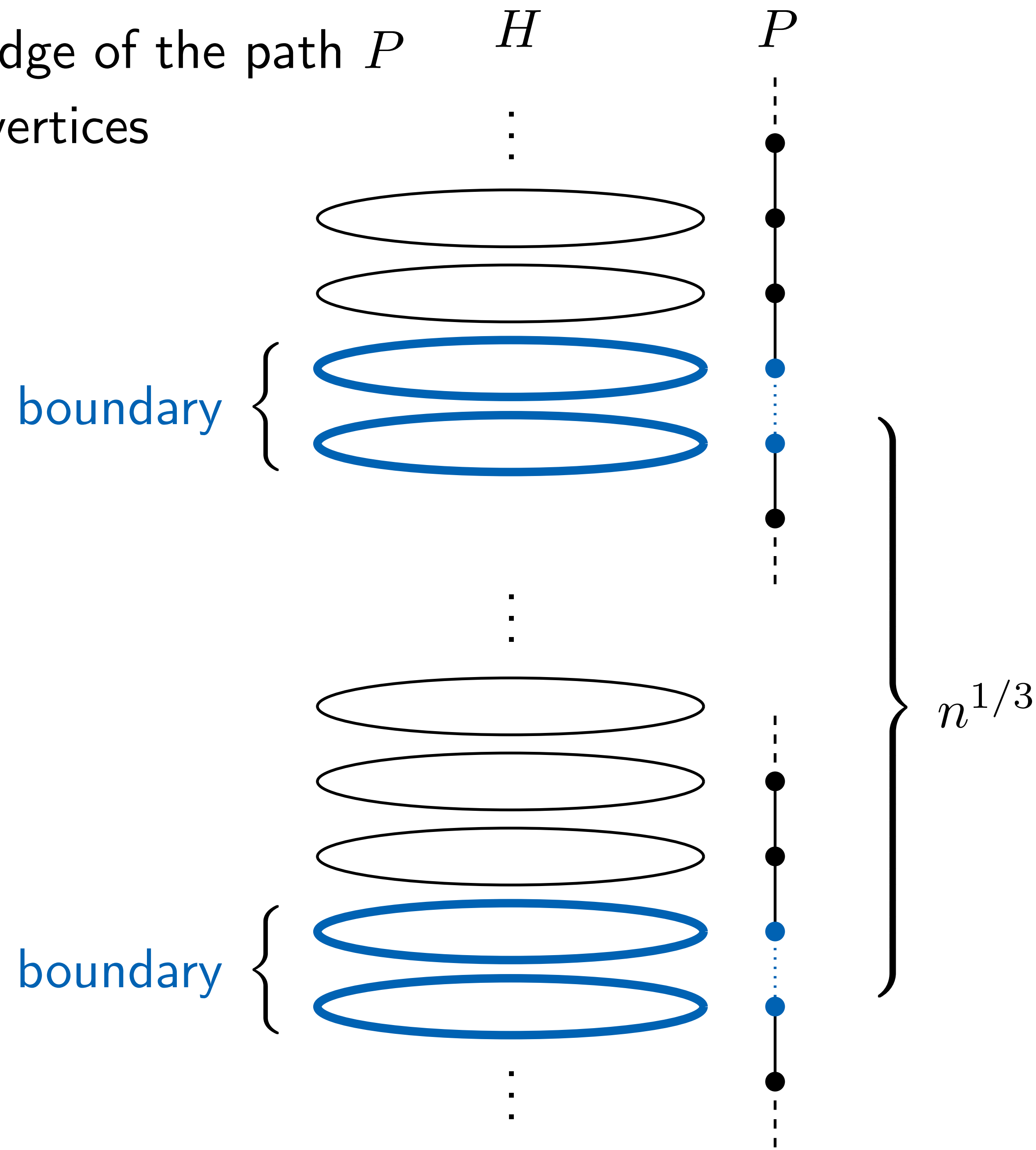
$\left(\frac{4}{3} + o(1)\right) \log n$-bit scheme

- "remove" every other $n^{1/3}$ edge of the path P so that there are $\mathcal{O}(n^{2/3})$ vertices in **boundary layers**



$\left(\frac{4}{3}+o(1)\right)\log n$-bit scheme

- "remove" every other $n^{1/3}$ edge of the path P so that there are $\mathcal{O}(n^{2/3})$ vertices in **boundary layers**

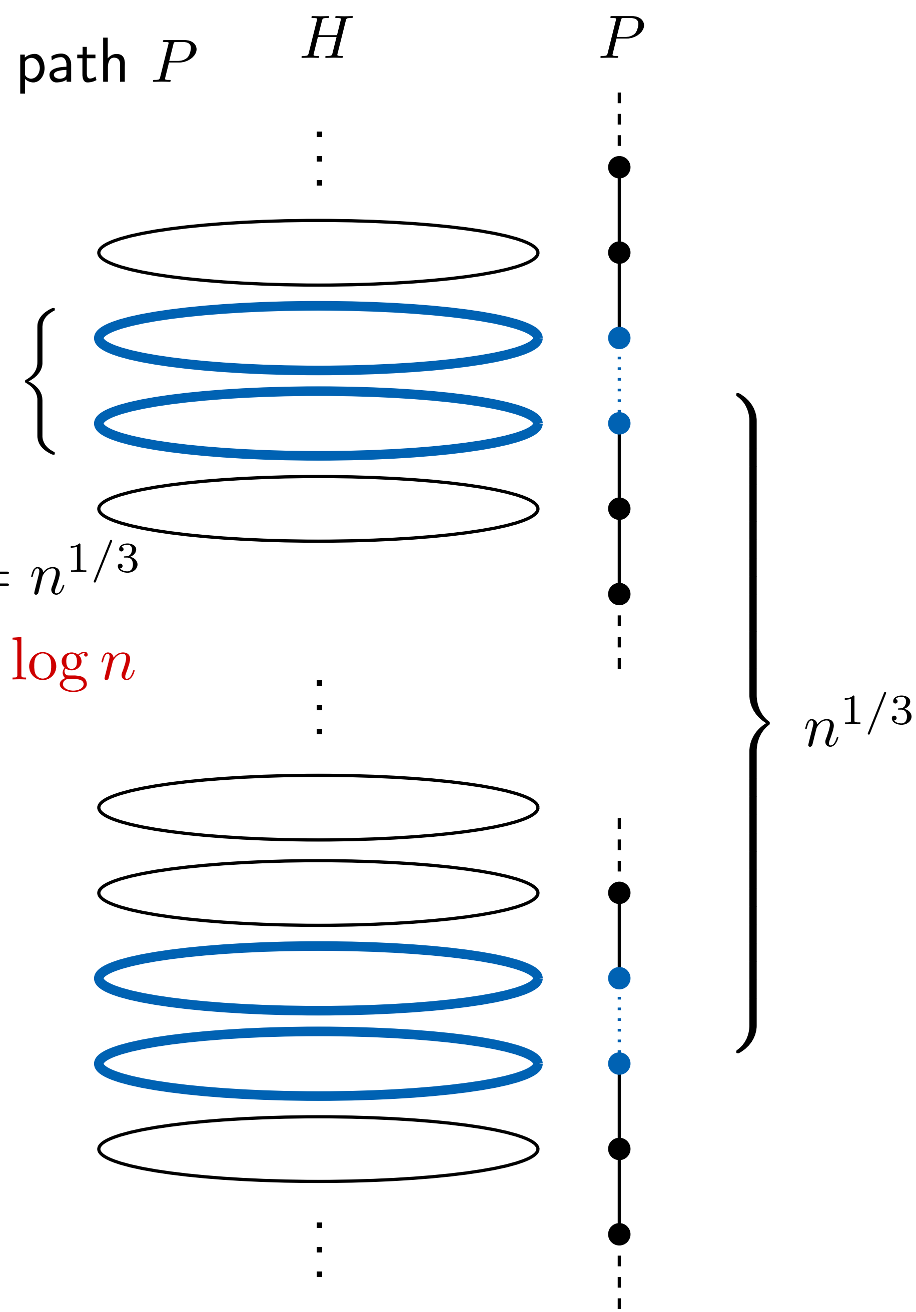


$\left(\frac{4}{3} + o(1)\right) \log n$-bit scheme

▷ "remove" every other $n^{1/3}$ edge of the path P so that there are $\mathcal{O}(n^{2/3})$ vertices in **boundary layers**

▷ each piece between the cuts is a subgraph of $H \boxtimes P'$ where $|P'| = n^{1/3}$

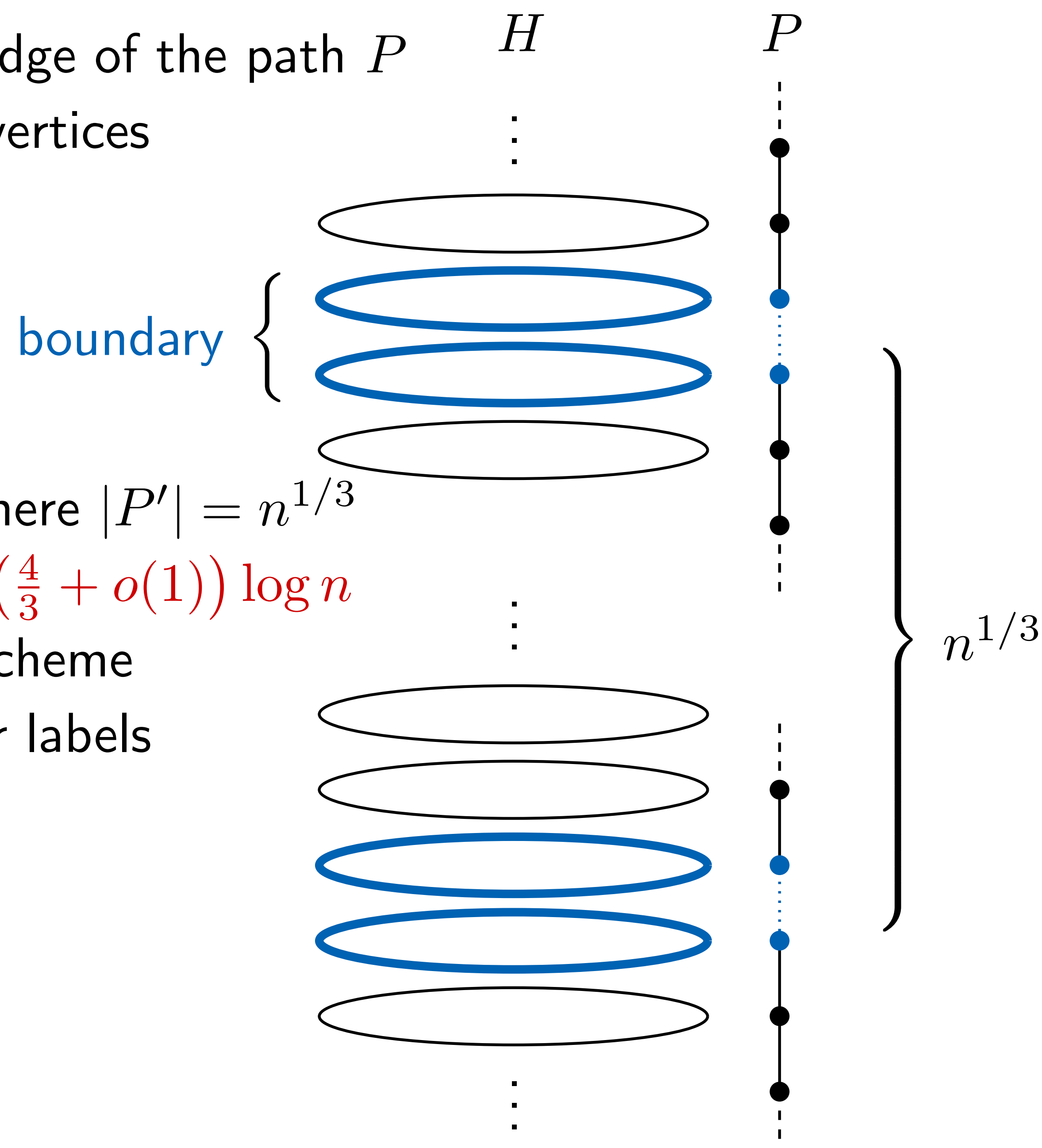
$(1 + o(1)) \log(n \cdot n^{1/3}) \equiv \left(\frac{4}{3} + o(1)\right) \log n$ -bit scheme



$\left(\frac{4}{3}+o(1)\right)\log n$-bit scheme

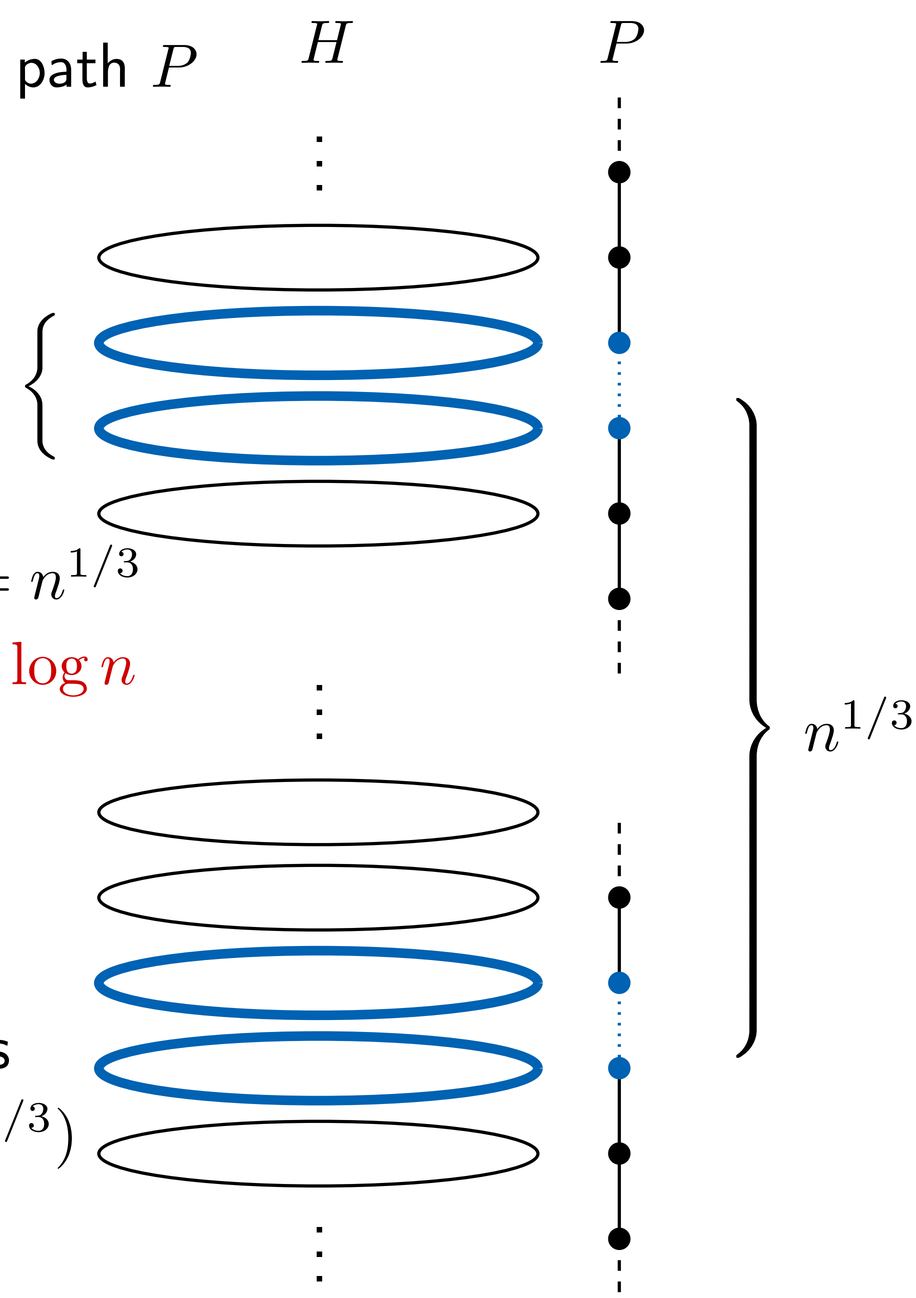
- "remove" every other $n^{1/3}$ edge of the path P so that there are $\mathcal{O}(n^{2/3})$ vertices in **boundary layers**
- each piece between the cuts is a subgraph of $H \boxtimes P'$ where $|P'| = n^{1/3}$

  $(1+o(1))\log(n \cdot n^{1/3}) \equiv \left(\frac{4}{3}+o(1)\right)\log n$ -bit scheme
- boundary vertices get shorter labels $\left(\frac{2}{3}+o(1)\right)\log n$-bit length



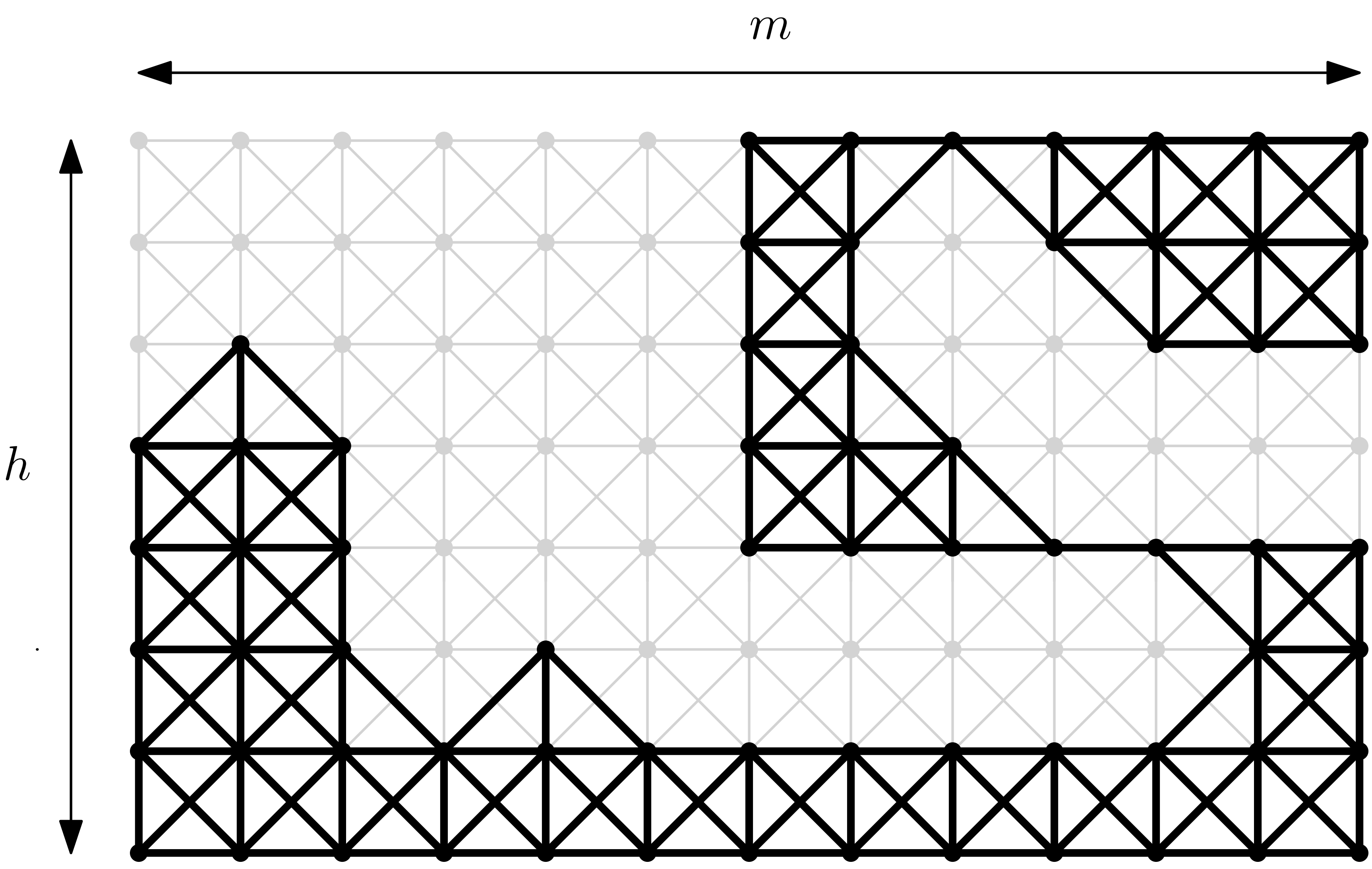
$\left(\frac{4}{3}+o(1)\right)\log n$-**bit scheme**

- "remove" every other $n^{1/3}$ edge of the path P
  so that there are $\mathcal{O}(n^{2/3})$ vertices
  in **boundary layers**
- each piece between the cuts
  is a subgraph of $H \boxtimes P'$ where $|P'| = n^{1/3}$
  $(1+o(1))\log(n \cdot n^{1/3}) \equiv \left(\frac{4}{3}+o(1)\right)\log n$ -bit scheme
- boundary vertices get shorter labels
  $\left(\frac{2}{3}+o(1)\right)\log n$-bit length
- the graph induced by boundary vertices
  has bounded treewidth and size $\mathcal{O}(n^{2/3})$
  $\left(\frac{2}{3}+o(1)\right)\log n$-bit scheme



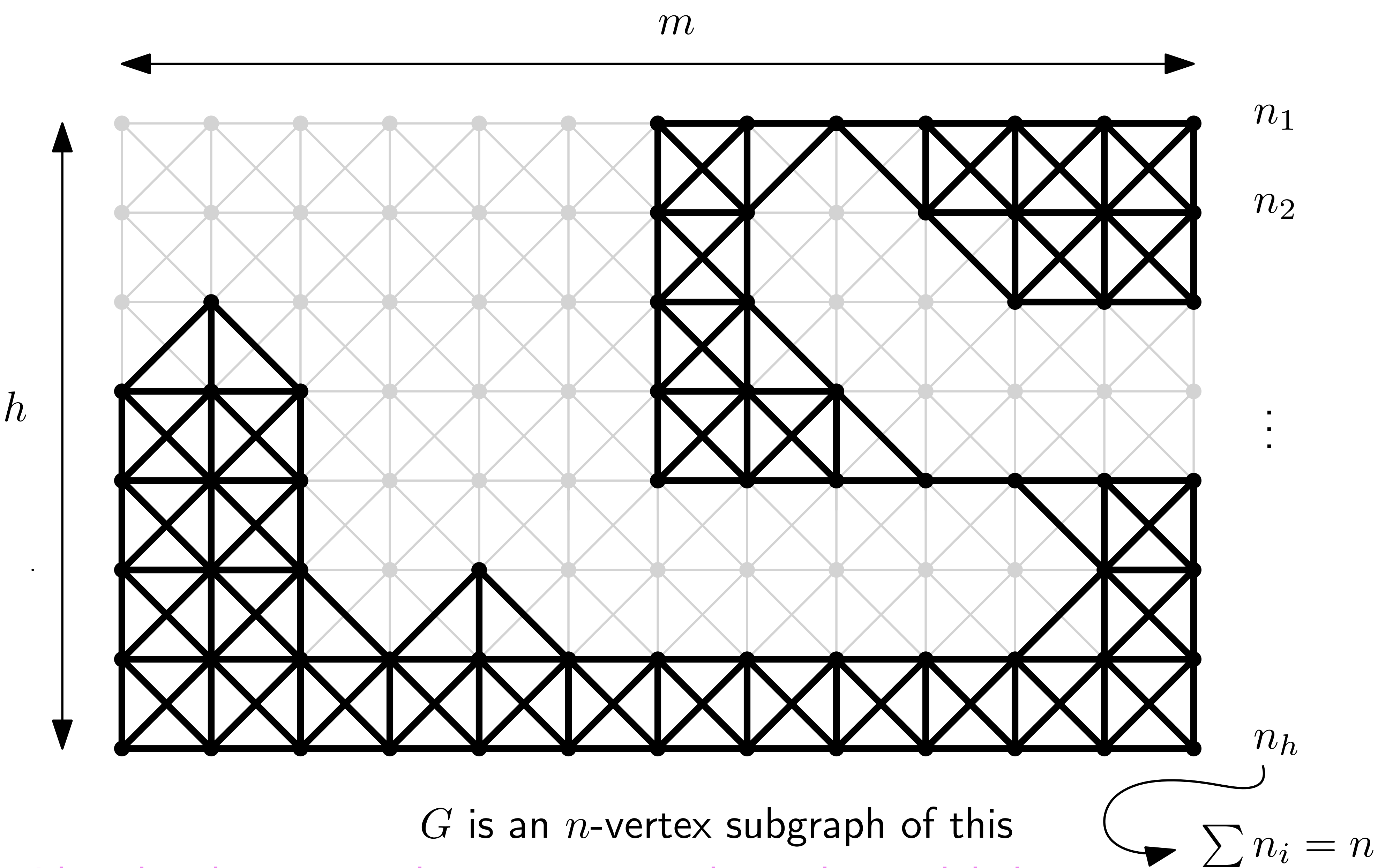
in total: $\left(\frac{4}{3}+o(1)\right)\log n$-**bit scheme**

special case: subgraphs of $P \boxtimes P$



G is an n-vertex subgraph of this

special case: subgraphs of $P \boxtimes P$



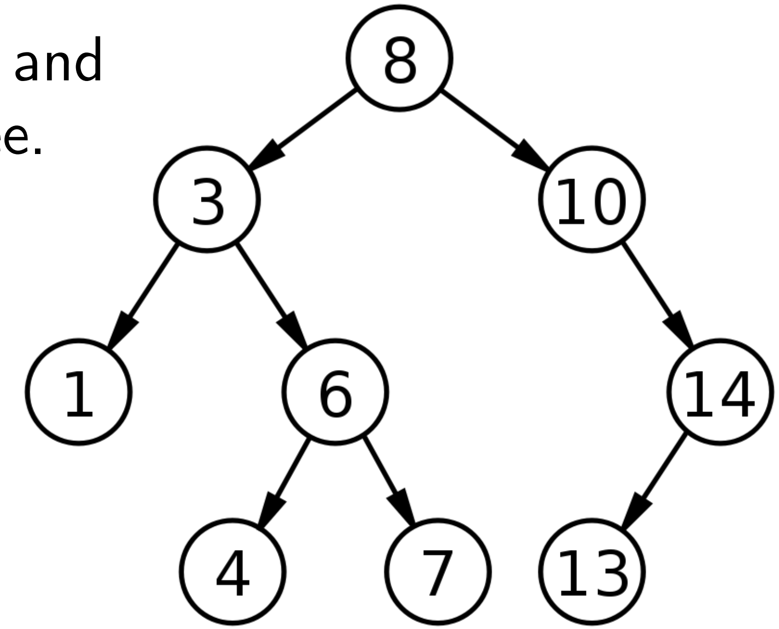
G is an n-vertex subgraph of this

Idea: let the rows with many vertices have shorter labels

weighted scheme for paths: preliminaries (1)

A binary search tree T is a binary tree whose node set $V(T)$ consists of distinct real numbers and that has the property:

For each node x in T,
  $z < x$ for each node z in x's left subtree and
  $z > x$ for each node z in x's right subtree.



$$x < y$$

$$\Updownarrow$$

$\sigma_T(x)$ is lexicographically less than $\sigma_T(y)$

weighted scheme for paths: preliminaries (2)

S finite subset of $\mathbb{R}$

$w : S \to \mathbb{R}^+$ weight function

$$W = \sum_{s \in S} w(s)$$

Observation

There exists a binary search tree T with $V(T) = S$ such that

$$d_T(y) \leqslant \log(W) - \log(w(y)), \text{ for each } y \in S.$$

weighted scheme for paths: preliminaries (2)

S finite subset of $\mathbb{R}$

$w : S \to \mathbb{R}^+$ weight function

$$W = \sum_{s \in S} w(s)$$

Observation

There exists a binary search tree T with $V(T) = S$ such that

$$d_T(y) \leqslant \log(W) - \log(w(y)), \text{ for each } y \in S.$$

To construct the tree:

- ▷ choose the root of T to be the unique node $s \in S$ such that
  $$\sum_{\substack{z \in S \\ z < s}} w(z) \leqslant W/2 \quad \text{and} \quad \sum_{\substack{z \in S \\ z > s}} w(z) < W/2$$
- ▷ then recurse on $\{z \mid z \in S \text{ and } z < s\}$ and $\{z \mid z \in S \text{ and } z < s\}$ to obtain the left and right subtrees of s, respectively.

weighted scheme for paths: preliminaries (3)

x, y nodes in bst T such that $x < y$ and there is no z in T with $x < z < y$, so x and y are consecutive in the sort of $V(T)$.

Then

- ▷ if y has no left child, $\sigma_T(x)$ is obtained from $\sigma_T(y)$ by removing all trailing 0's and the last 1;
- ▷ if y has a left child, $\sigma_T(x)$ is obtained from $\sigma_T(y)$ by appending a 0 followed by $d_T(y) - d_T(x) - 1$ 1's.

Thus, there exists a universal function $D : (\{0,1\}^*)^2 \to \{0,1\}^*$ such that for every bst T with x, y being consecutive in $V(T)$, there exists $\delta_T(y) \in \{0,1\}^*$ with $|\delta_T(y)| = \mathcal{O}(\log h(T))$ such that

$$D(\sigma_T(y), \delta_T(y)) = \sigma_T(x).$$

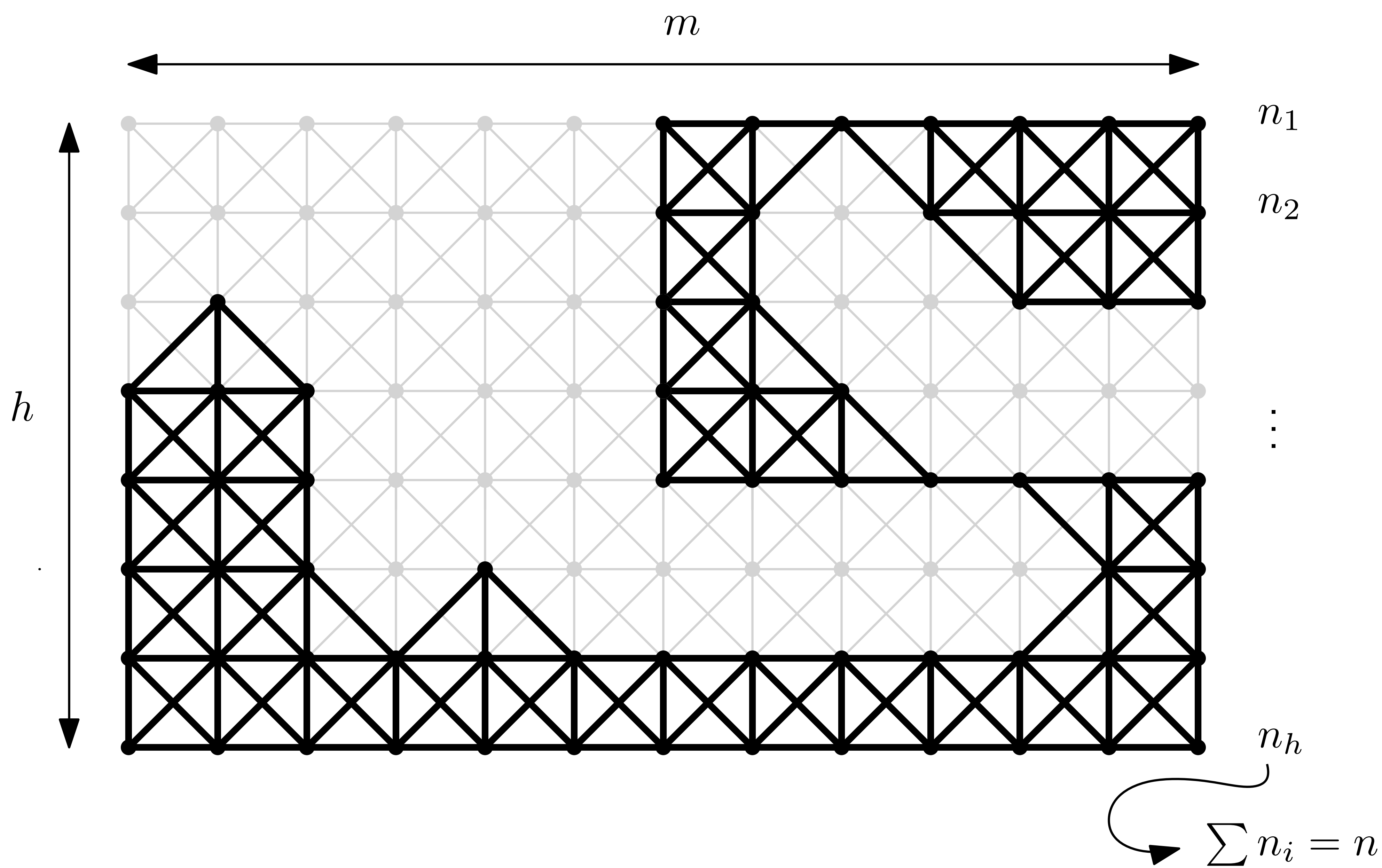
weighted scheme for paths

There exists a universal function $A : (\{0,1\}^*)^2 \to \{-1,0,1,\perp\}$ such that,
for any $h \in \mathbb{N}$, and any weight function $w : \{1,\ldots,h\} \to \mathbb{R}^+$
there is a prefix-free code $\alpha : \{1,\ldots,h\} \to \{0,1\}^*$ such that

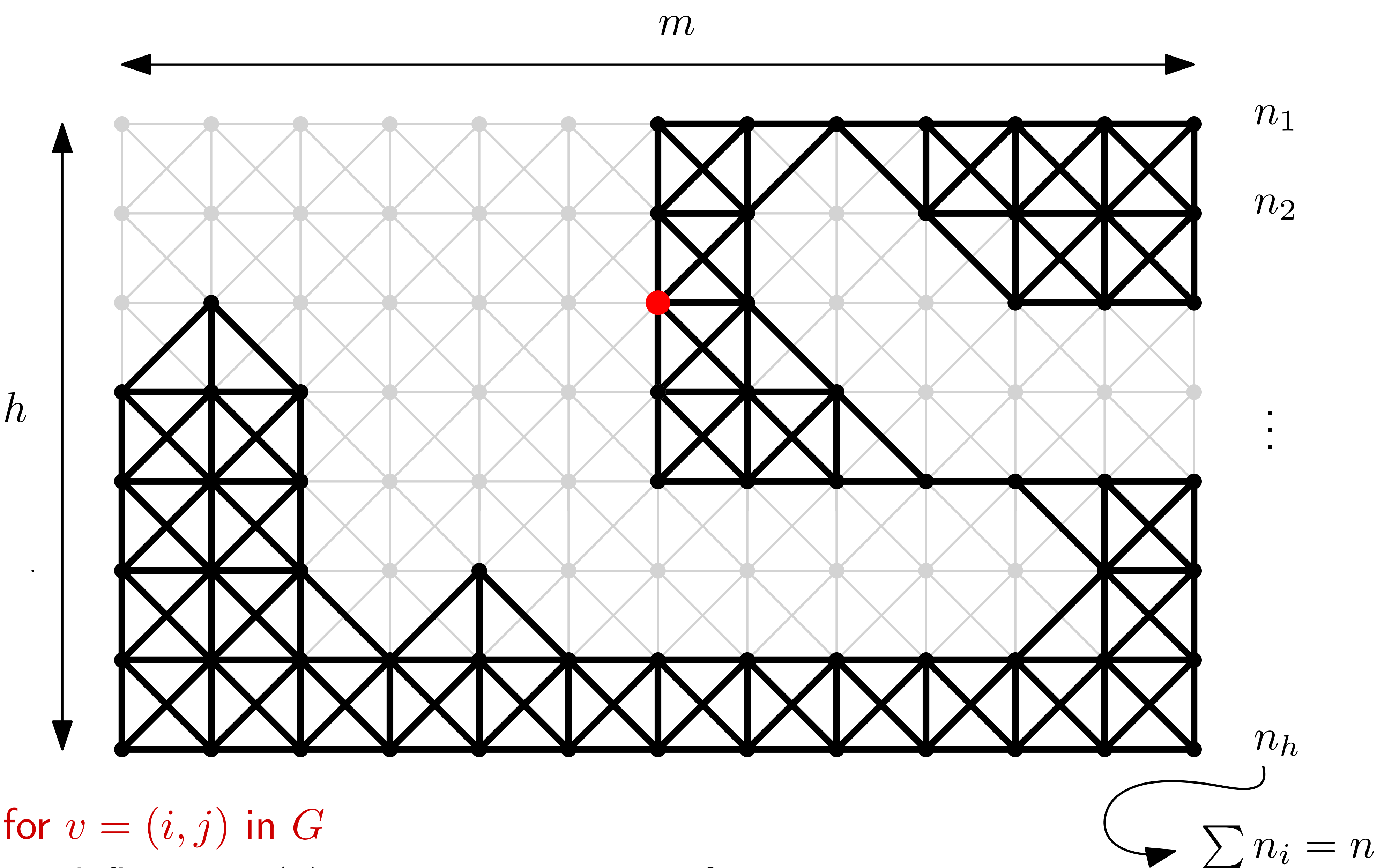
- ▷ for each $i \in \{1,\ldots,h\}$, $|\alpha(i)| = \log W - \log w(i) + \mathcal{O}(\log\log h)$;
- ▷ for any $i,j \in \{1,\ldots,h\}$,

where $W = \sum_{i=1}^{h} w(i)$

$$A(\alpha(i),\alpha(j)) = \begin{cases} 0 & \text{if } j = i; \\ 1 & \text{if } j = i+1; \\ -1 & \text{if } j = i-1; \\ \perp & \text{otherwise.} \end{cases}$$



m
h
n_1
n_2
$\vdots$
n_h
$\sum n_i = n$



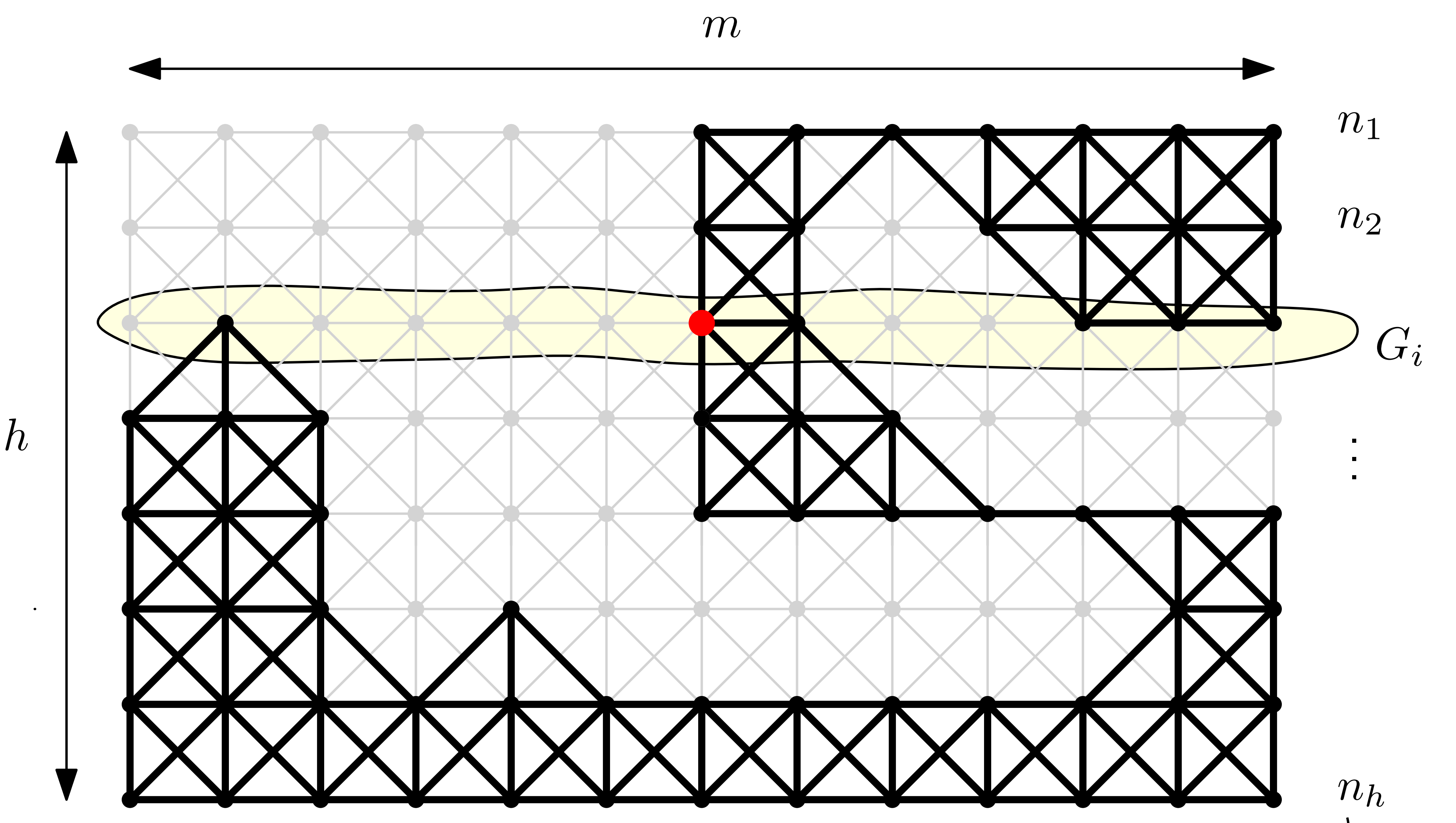
for $v = (i, j)$ in G

define $\mathrm{label}(v)$ as a concatenation of

row label: $\log n - \log n_i + o(\log n)$

column label: $\log n_i + o(\log n)$

weighted scheme



$\sum n_i = n$

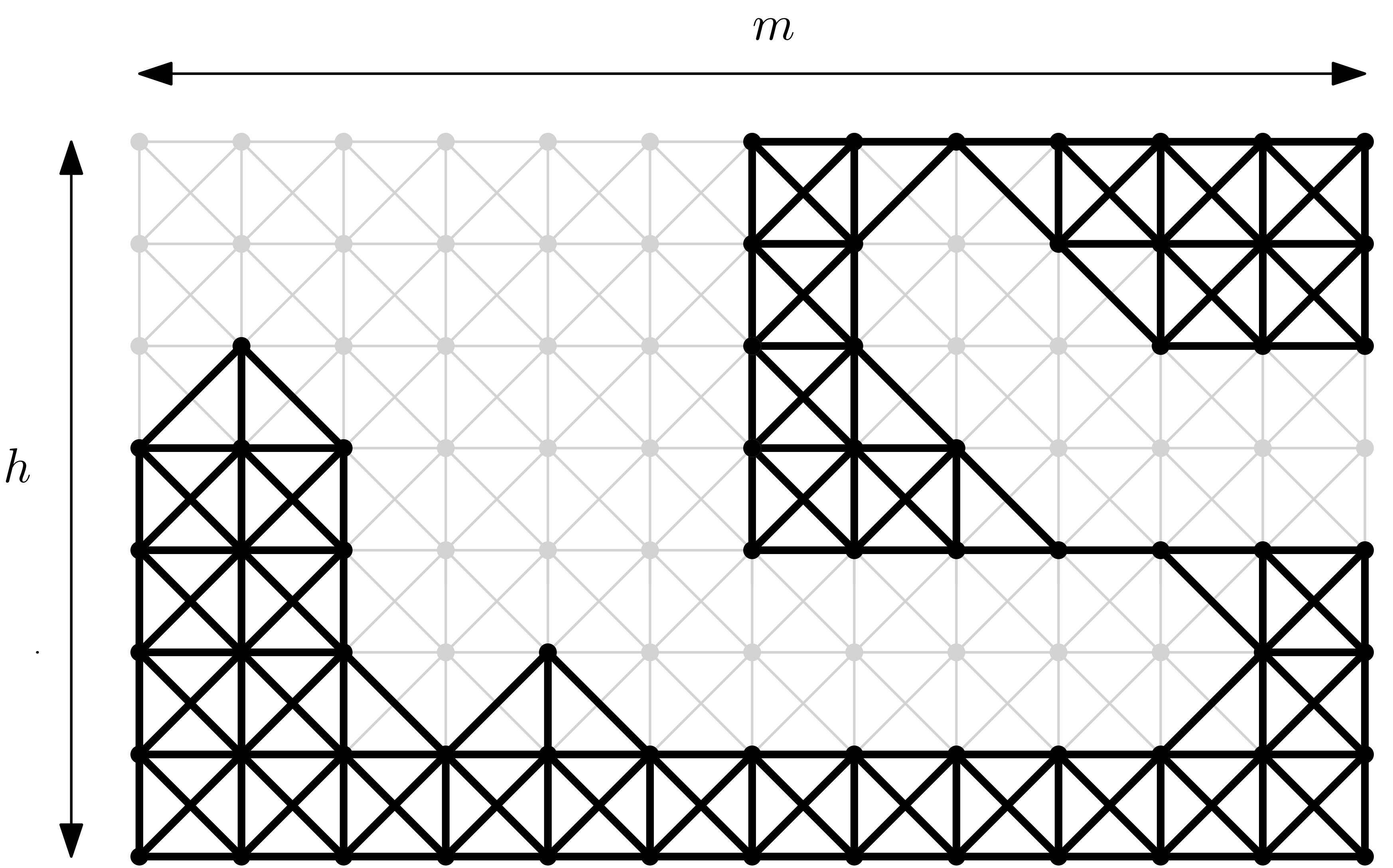
for $v = (i, j)$ in G

define $\text{label}(v)$ as a concatenation of

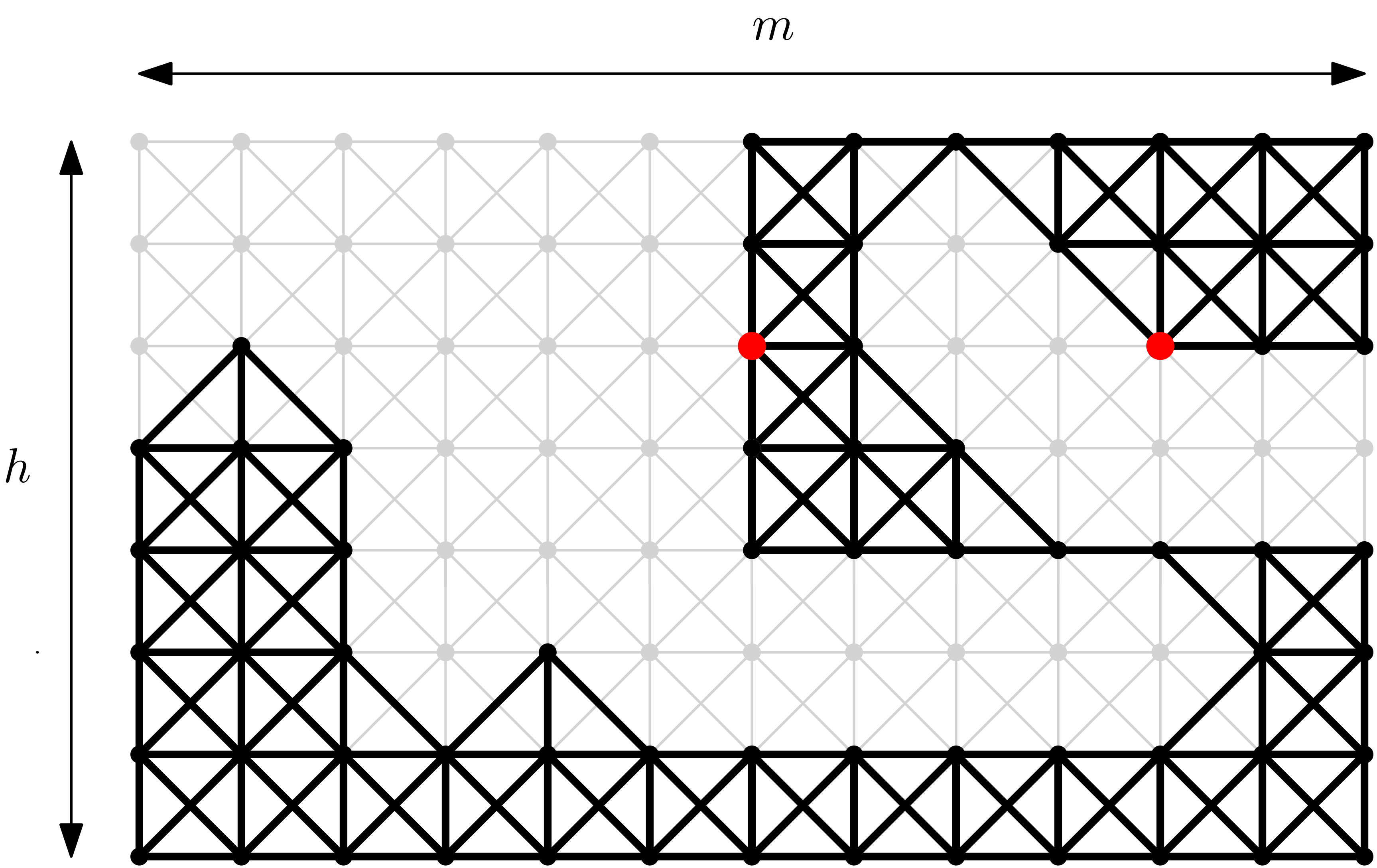
row label: $\log n - \log n_i + o(\log n)$

column label: $\log n_i + o(\log n)$

weighted scheme



given label(v), label(w)

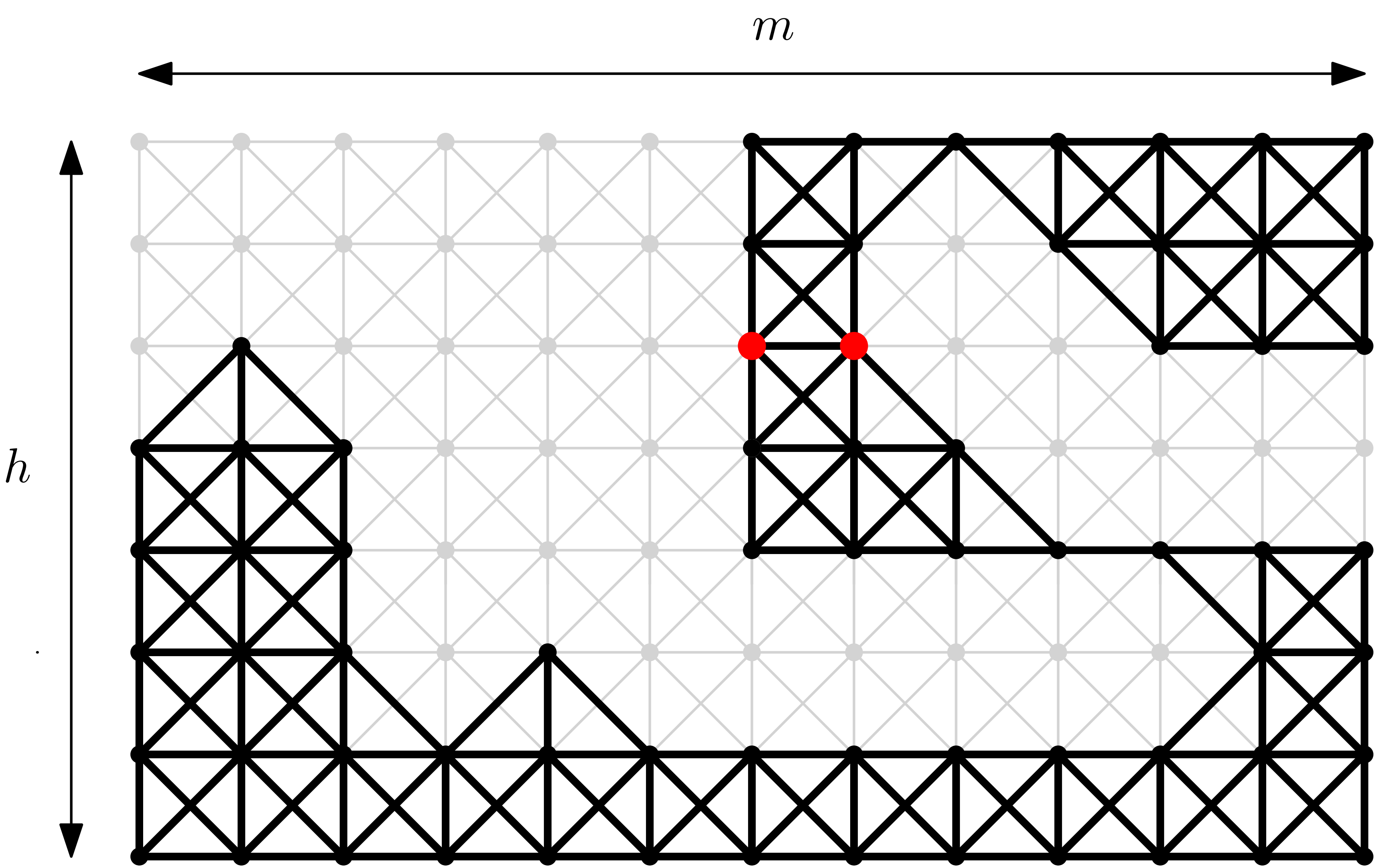


given $\mathrm{label}(v)$, $\mathrm{label}(w)$

$\triangleright$ if $\mathrm{row}(v) = \mathrm{row}(w)$ then column labels will do

YES

NO

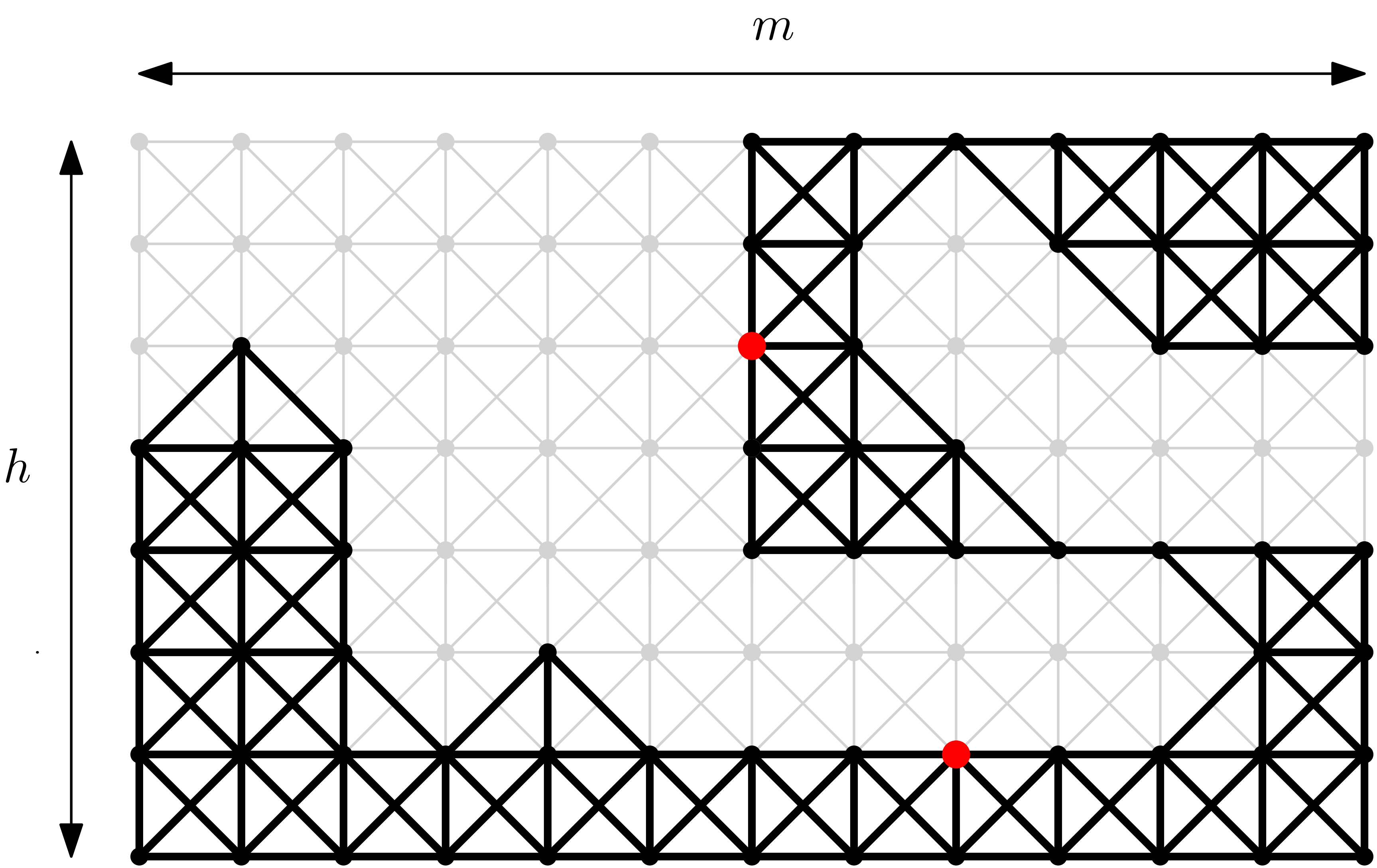


given $\mathrm{label}(v)$, $\mathrm{label}(w)$

▷ if $\mathrm{row}(v) = \mathrm{row}(w)$ then column labels will do

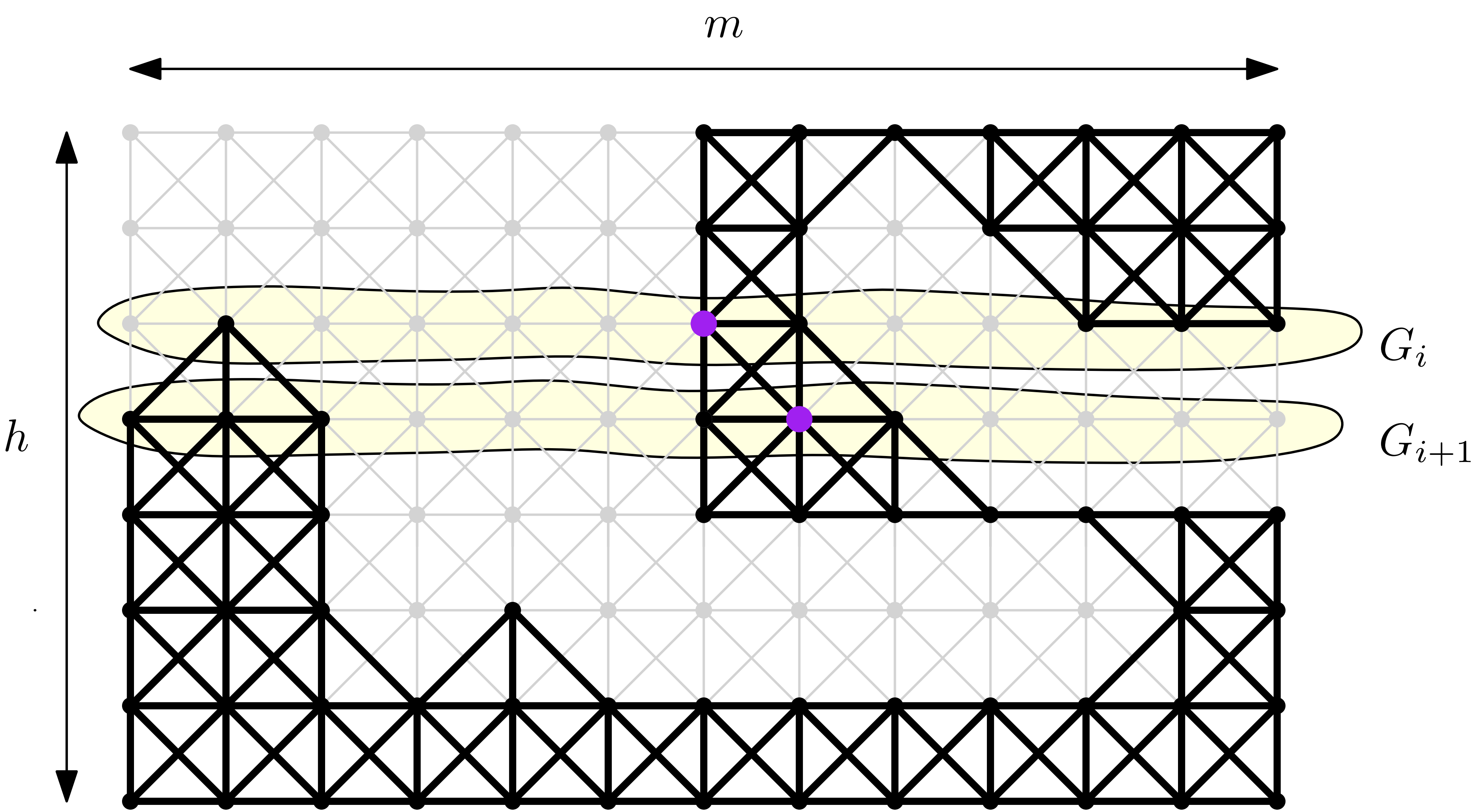
YES

NO



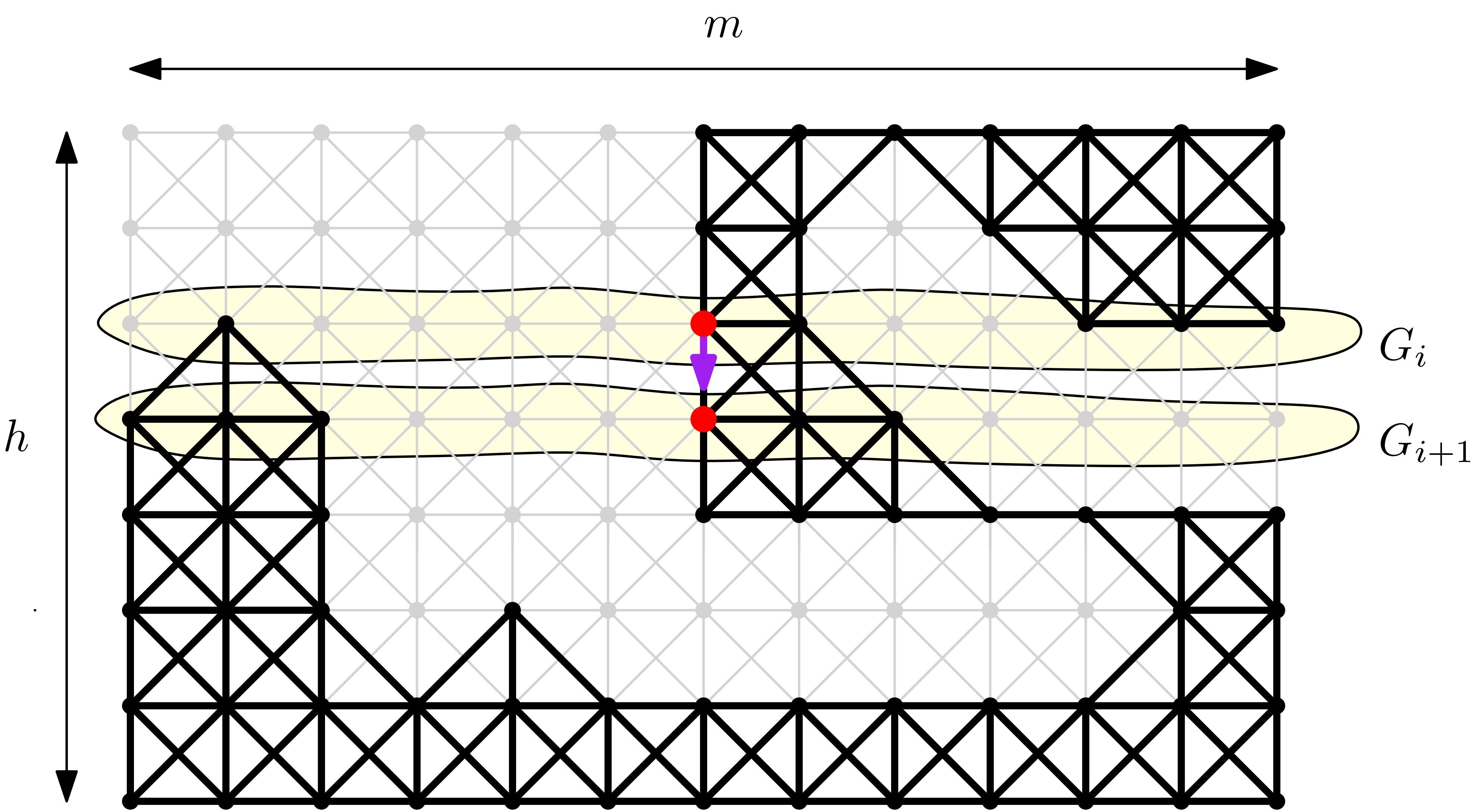
given $\mathrm{label}(v)$, $\mathrm{label}(w)$

- ▷ if $\mathrm{row}(v) = \mathrm{row}(w)$ then column labels will do YES NO
- ▷ if $|\,\mathrm{row}(v) - \mathrm{row}(w)| > 1$ then output NO



given $\mathrm{label}(v)$, $\mathrm{label}(w)$

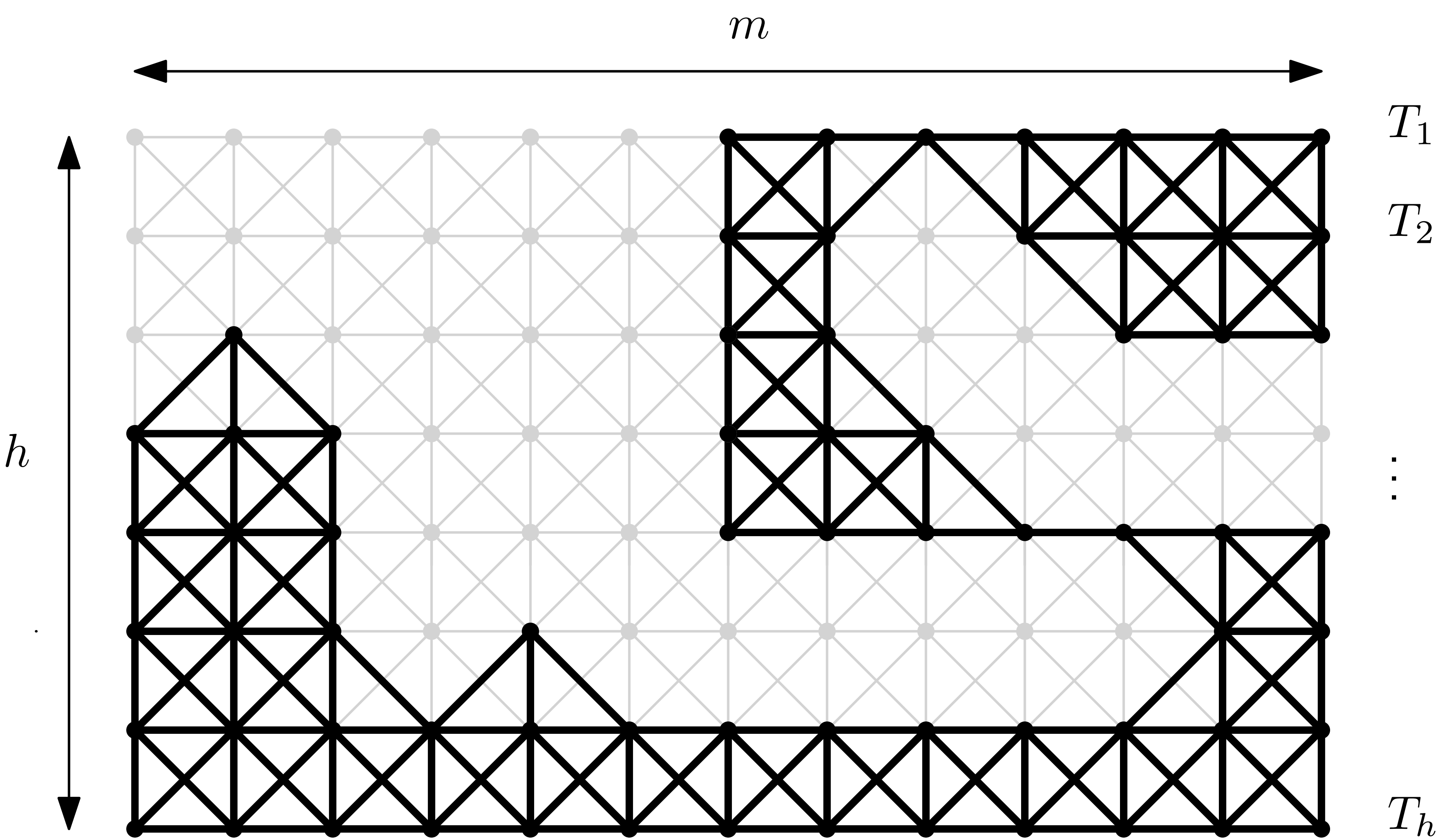
- ▷ if $\mathrm{row}(v) = \mathrm{row}(w)$ then column labels will do ↗ YES ↘ NO
- ▷ if $|\mathrm{row}(v) - \mathrm{row}(w)| > 1$ then output NO
- ▷ if $|\mathrm{row}(v) - \mathrm{row}(w)| = 1$ then ???



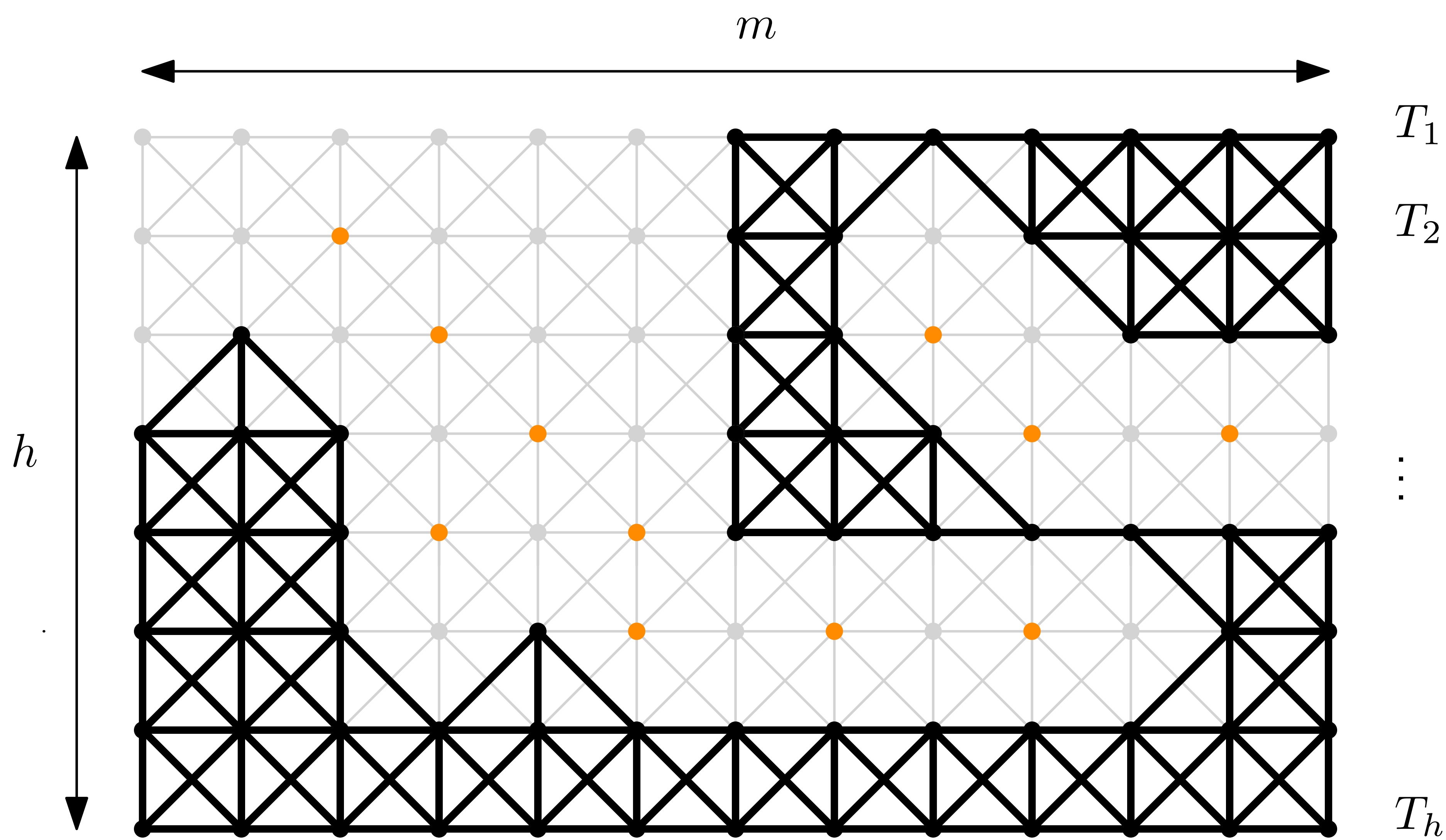
row label: $\log n - \log n_i + o(\log n)$

column label: $\log n_i + o(\log n)$

transition label: $o(\log n)$



$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree



$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree

fractional cascading

a-chunking sequence

$X, Y \subset \mathbb{R} \quad a \geqslant 1$

X a-chunks Y if, for any $a+1$-element subset $S \subseteq Y$, there exists $x \in X$, such that

$$\min S \leqslant x \leqslant \max S$$

$V_1, \ldots, V_h$ is a-chunking if V_y a-chunks V_{y+1} and V_{y+1} a-chunks V_y

a-chunking sequence

$X, Y \subset \mathbb{R} \qquad a \geqslant 1$

X a-chunks Y if, for any $a+1$-element subset $S \subseteq Y$, there exists $x \in X$, such that

$$\min S \leqslant x \leqslant \max S$$

$V_1, \ldots, V_h$ is a-chunking if V_y a-chunks V_{y+1} and V_{y+1} a-chunks V_y

Lemma For any finite sets $S_1, \ldots, S_h \subset \mathbb{R}$ and any integer $a \geqslant 1$, there exist sets $V_1, \ldots, V_h \subset \mathbb{R}$ such that

- $V_y \supseteq S_y$, for each $y \in \{1, \ldots, h\}$;
- $V_1, \ldots, V_h$ is a-chunking;
- $\sum |V_y| \leqslant \left(\frac{a+1}{a}\right)^2 \cdot \sum |S_y|$.

a-chunking sequence

$X, Y \subset \mathbb{R} \quad a \geqslant 1$

X a-chunks Y if, for any $a+1$-element subset $S \subseteq Y$, there exists $x \in X$, such that

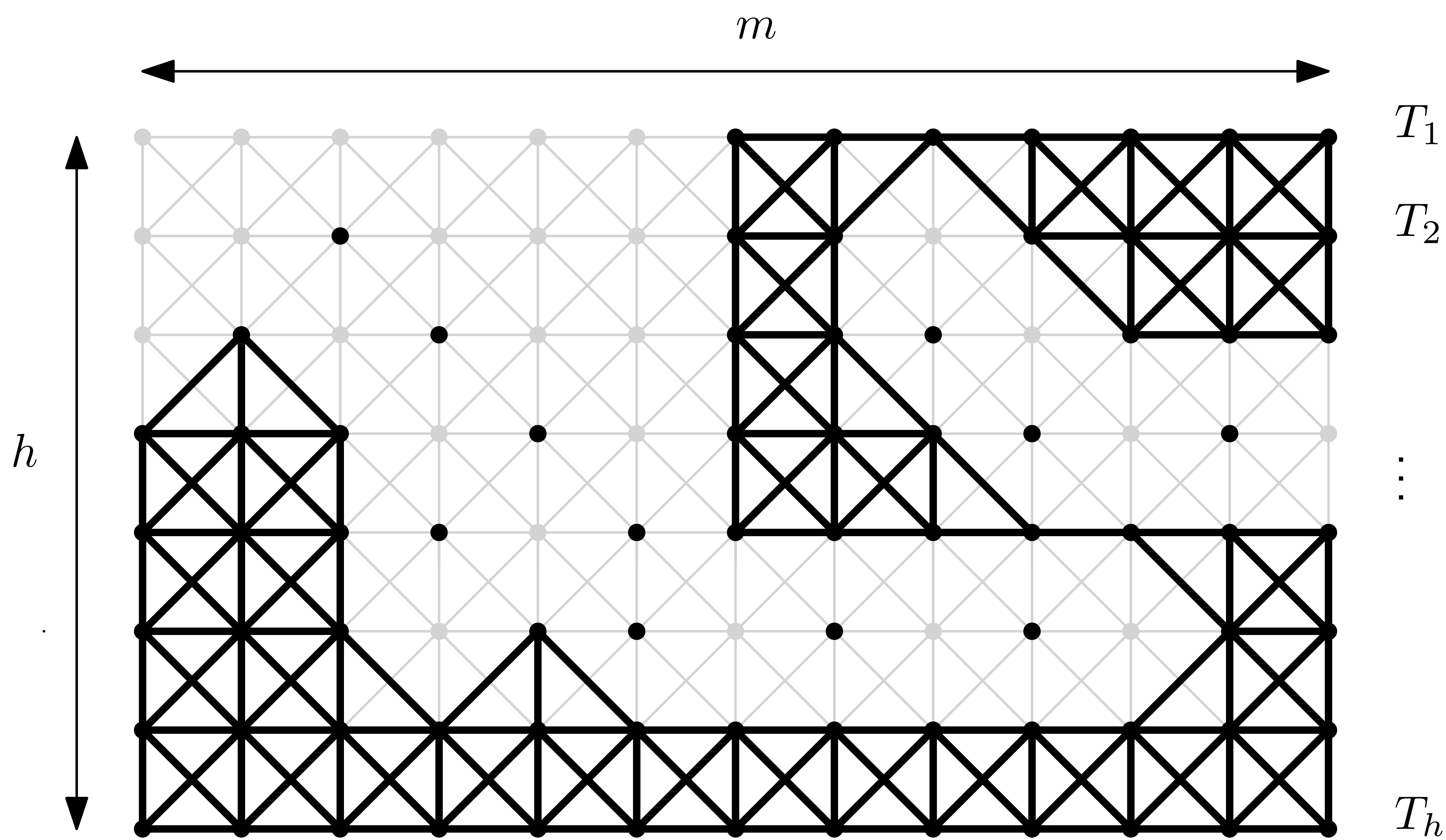
$$\min S \leqslant x \leqslant \max S$$

$V_1, \ldots, V_h$ is a-chunking if V_y a-chunks V_{y+1} and V_{y+1} a-chunks V_y

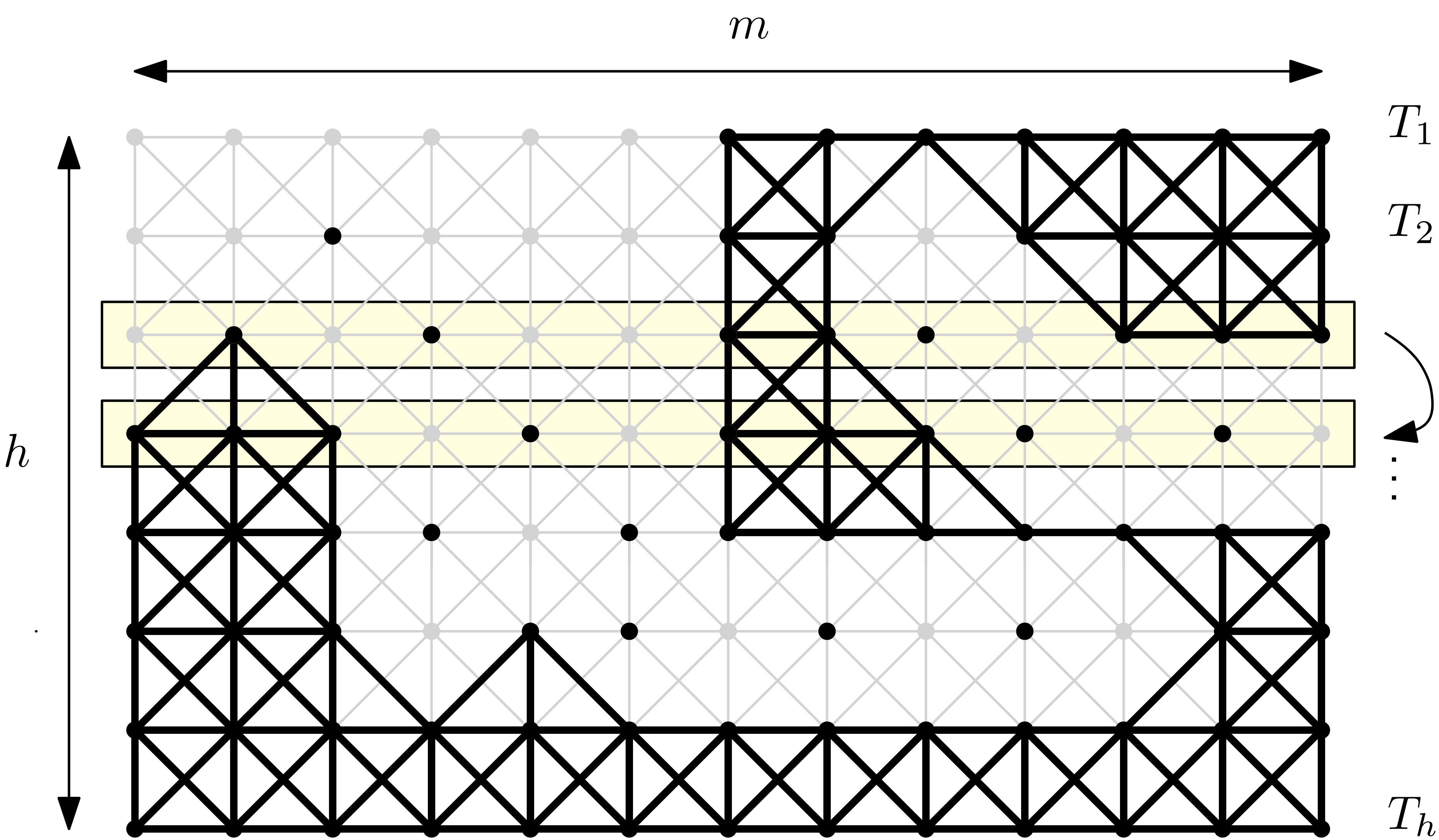
Lemma For any finite sets $S_1, \ldots, S_h \subset \mathbb{R}$ and any integer $a = 1$, there exist sets $V_1, \ldots, V_h \subset \mathbb{R}$ such that

- ▷ $V_y \supseteq S_y$, for each $y \in \{1, \ldots, h\}$;
- ▷ $V_1, \ldots, V_h$ is 1-chunking;
- ▷ $\sum |V_y| \leqslant 4 \cdot \sum |S_y|$.

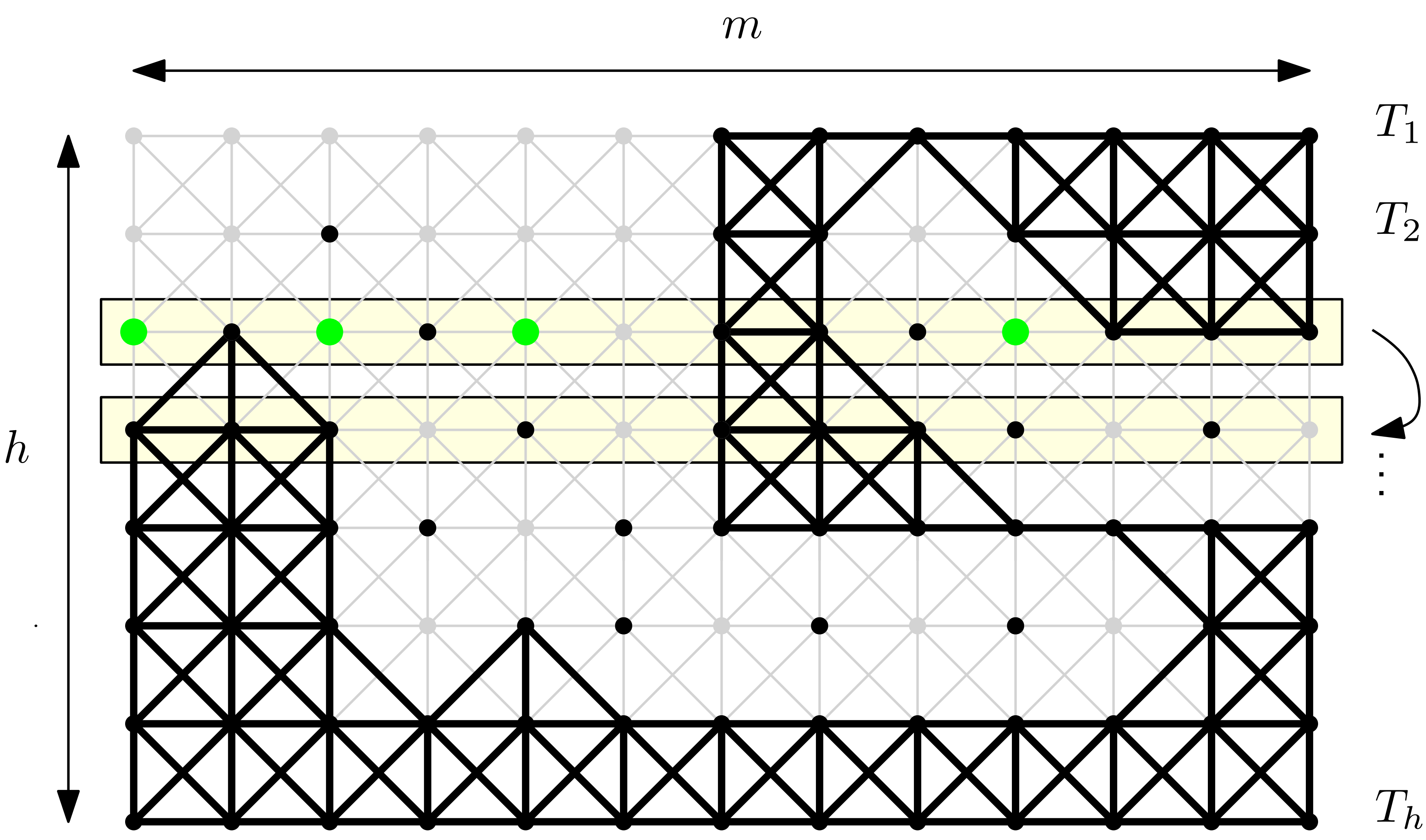
$a = 1$



$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree

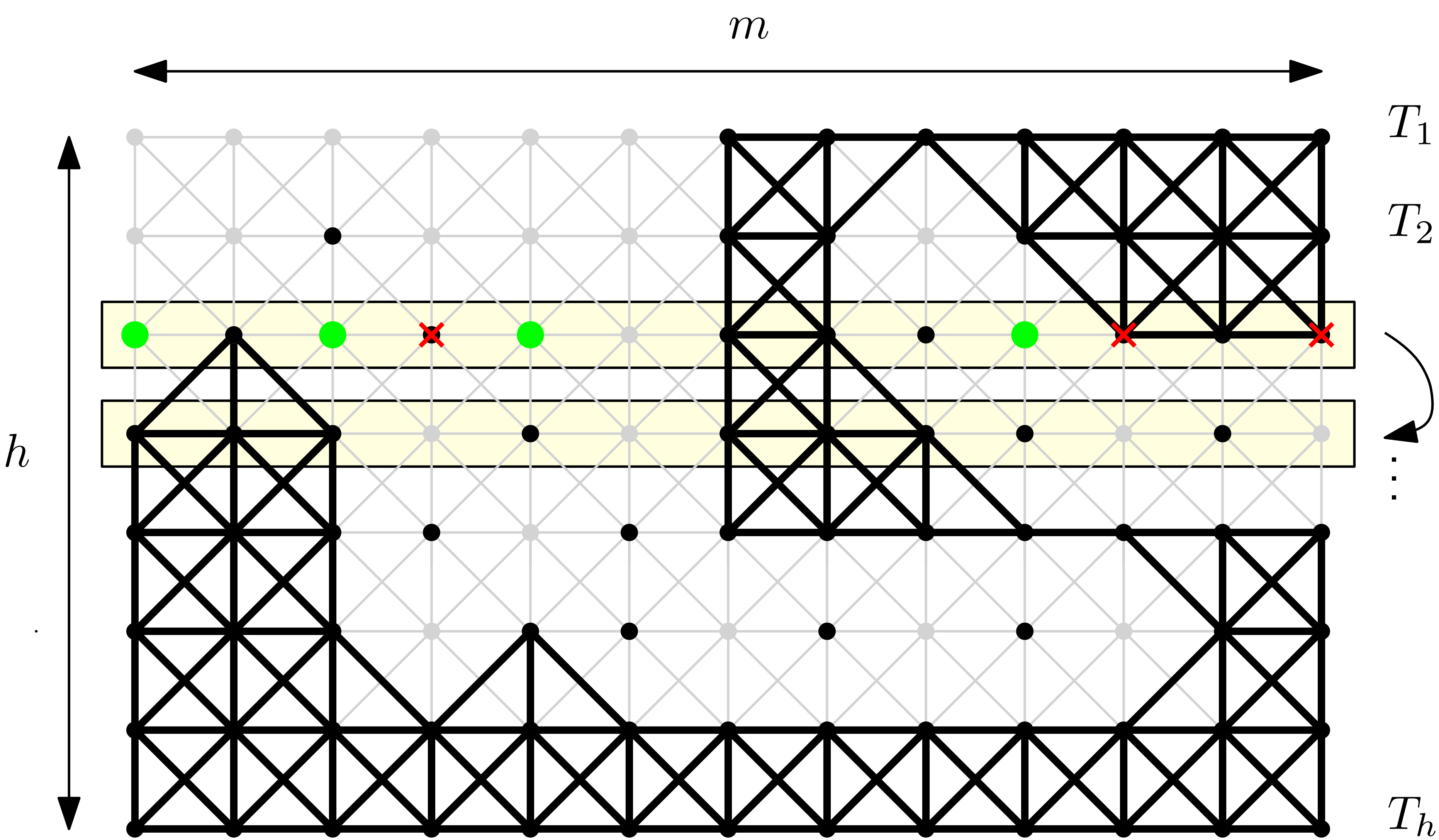


$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree



$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree

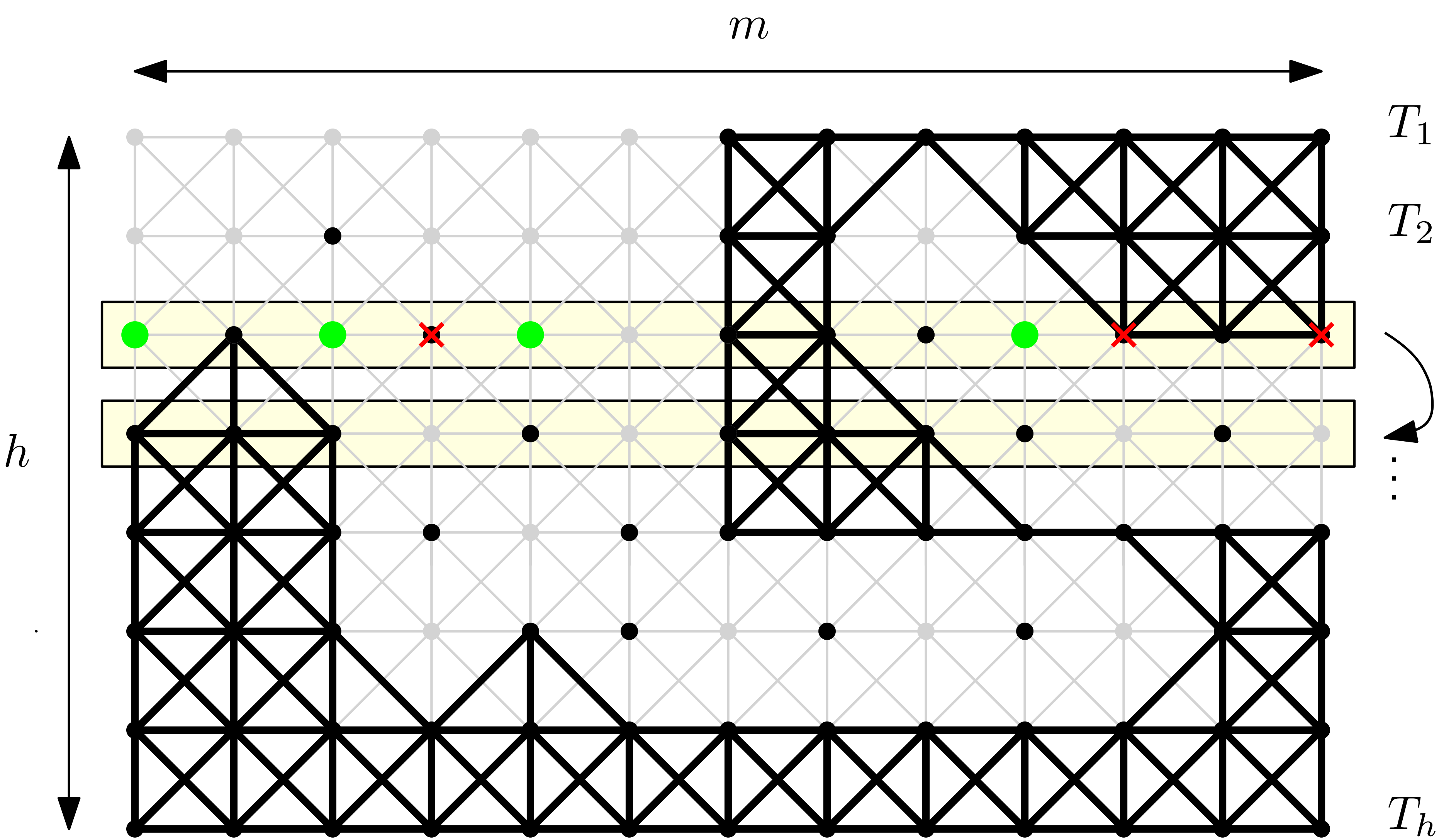
▷ insertions



$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree

▷ insertions

▷ deletions



$(T_1, T_2, \ldots, T_h)$ – a trace of a single dynamic binary search tree

▷ insertions

▷ deletions

▷ rebalancing

- insertions

  $h(T') \leqslant h(T_i) + 1$

  no impact on signatures of elements that are in both T_i and T_{i+1}

▷ insertions

$h(T') \leqslant h(T_i) + 1$

no impact on signatures of elements that are in both T_i and T_{i+1}

▷ deletions

with a standard bst algorithm

signatures of elements in T'' are prefixes of their signatures in T'

$h(T'') \leqslant h(T')$

$\frac{1}{4}|T'| \leqslant |T''| \leqslant |T'| \longrightarrow \log|T'| \leqslant \log|T''| + 2$

▷ insertions

$h(T') \leqslant h(T_i) + 1$

no impact on signatures of elements that are in both T_i and T_{i+1}

▷ deletions

with a standard bst algorithm

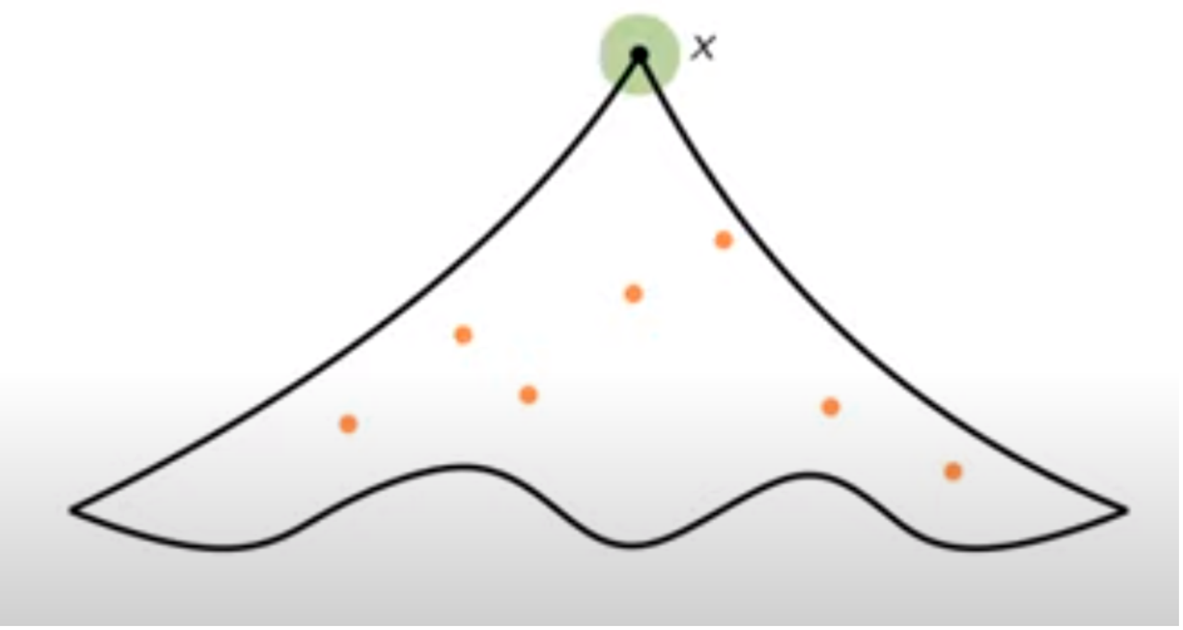
signatures of elements in T'' are prefixes of their signatures in T'

$h(T'') \leqslant h(T')$

$\frac{1}{4}|T'| \leqslant |T''| \leqslant |T'| \longrightarrow \log|T'| \leqslant \log|T''| + 2$

▷ rebalancing

balance(x, k)



▷ insertions

$h(T') \leqslant h(T_i) + 1$

no impact on signatures of elements that are in both T_i and T_{i+1}

▷ deletions

with a standard bst algorithm

signatures of elements in T'' are prefixes of their signatures in T'

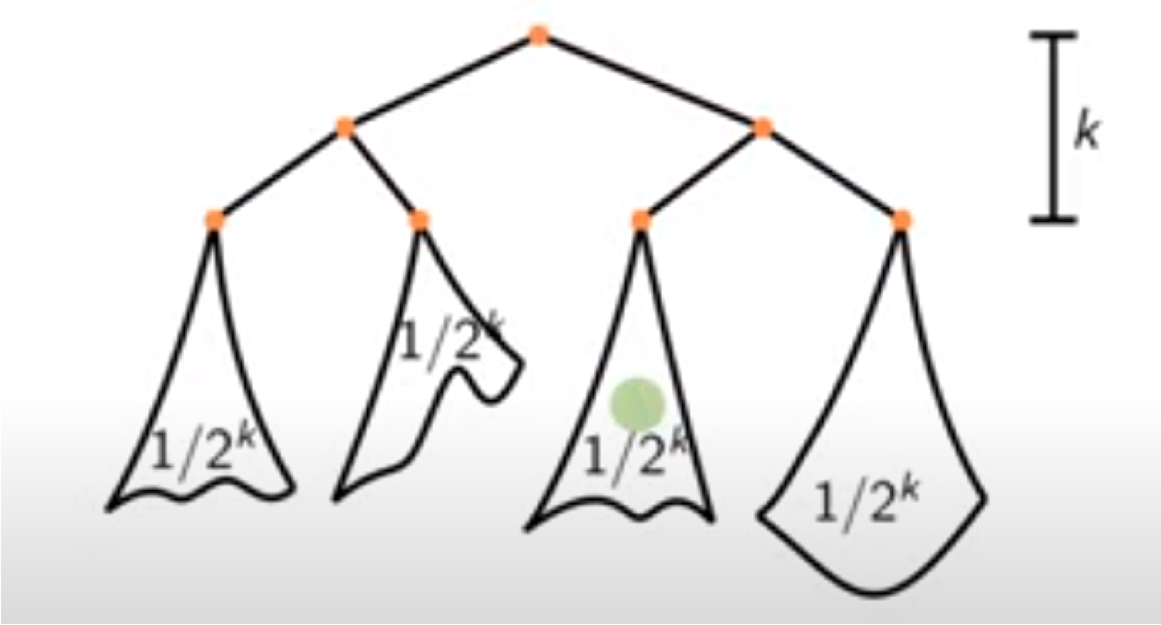
$h(T'') \leqslant h(T')$

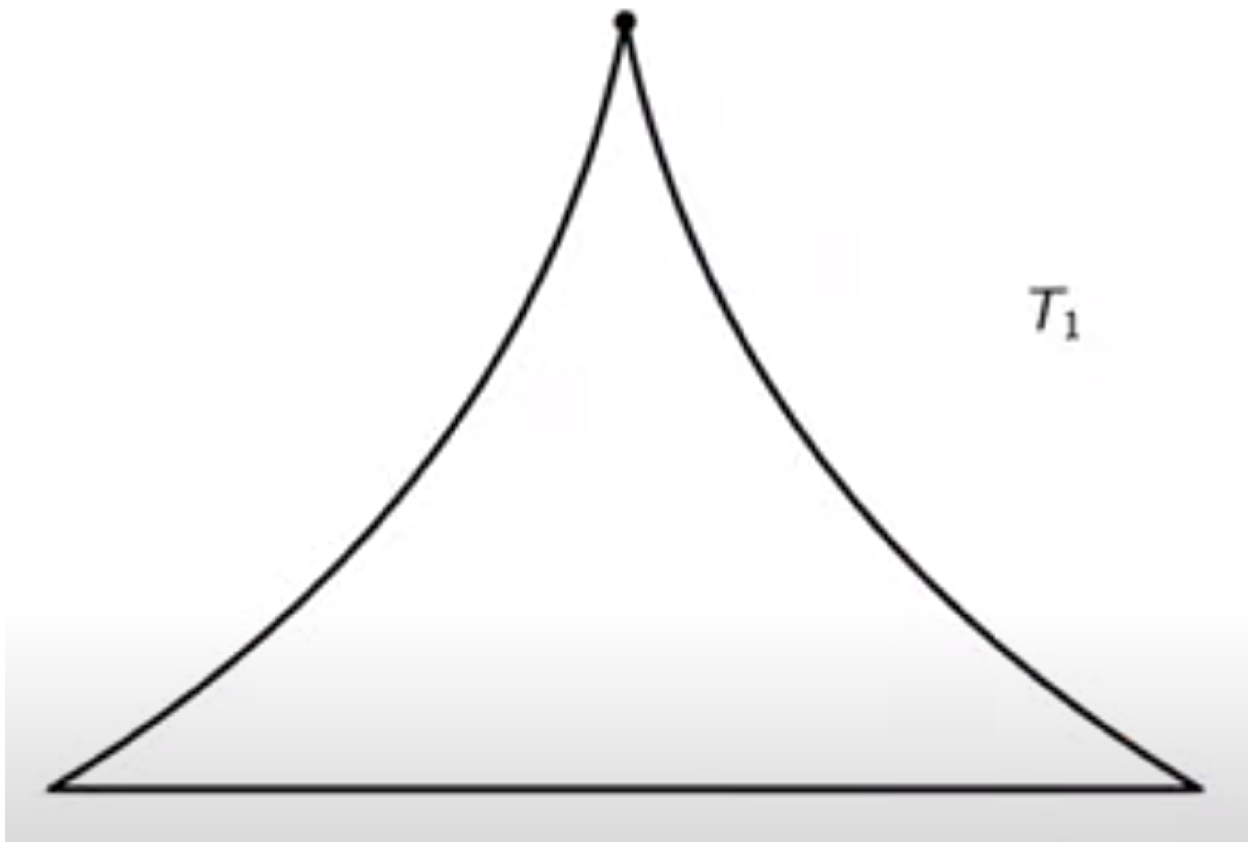
$\frac{1}{4}|T'| \leqslant |T''| \leqslant |T'| \longrightarrow \log|T'| \leqslant \log|T''| + 2$

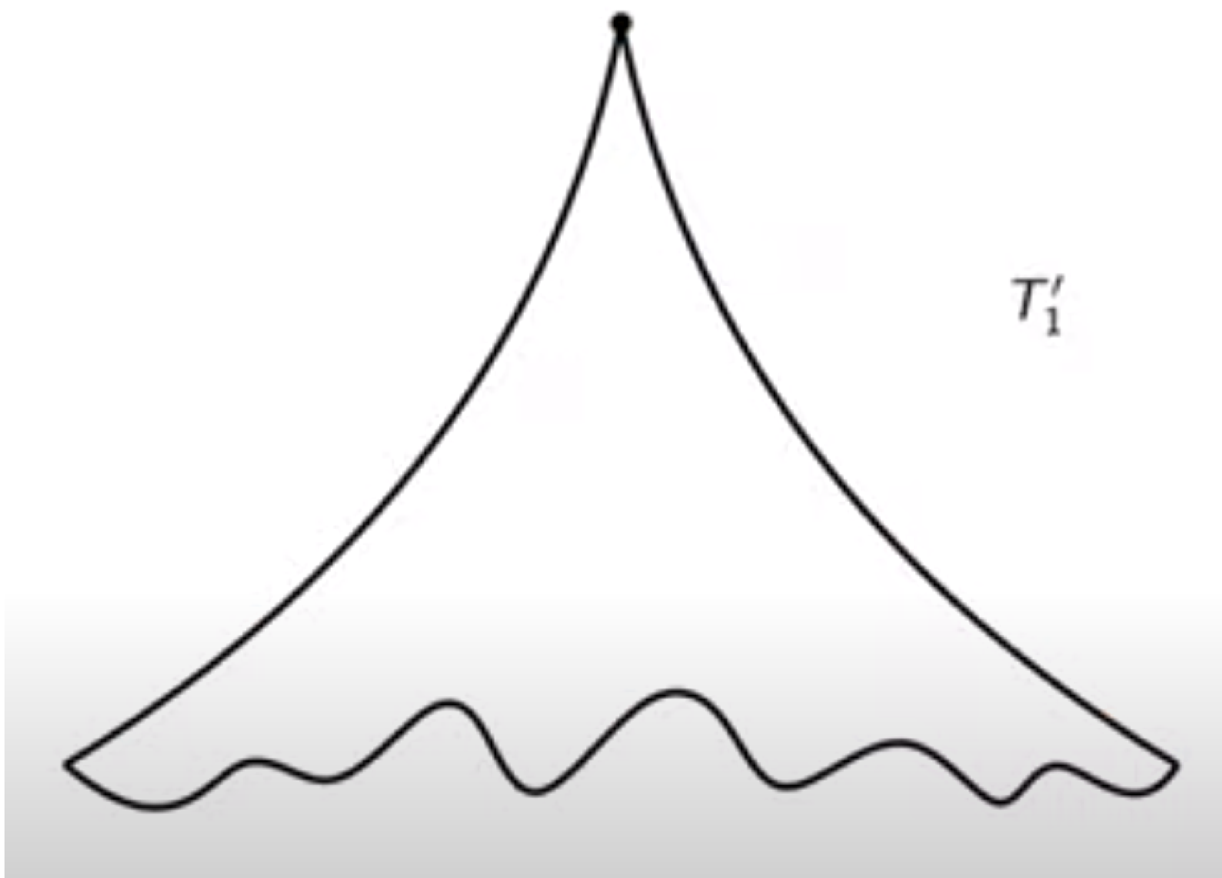
▷ rebalancing

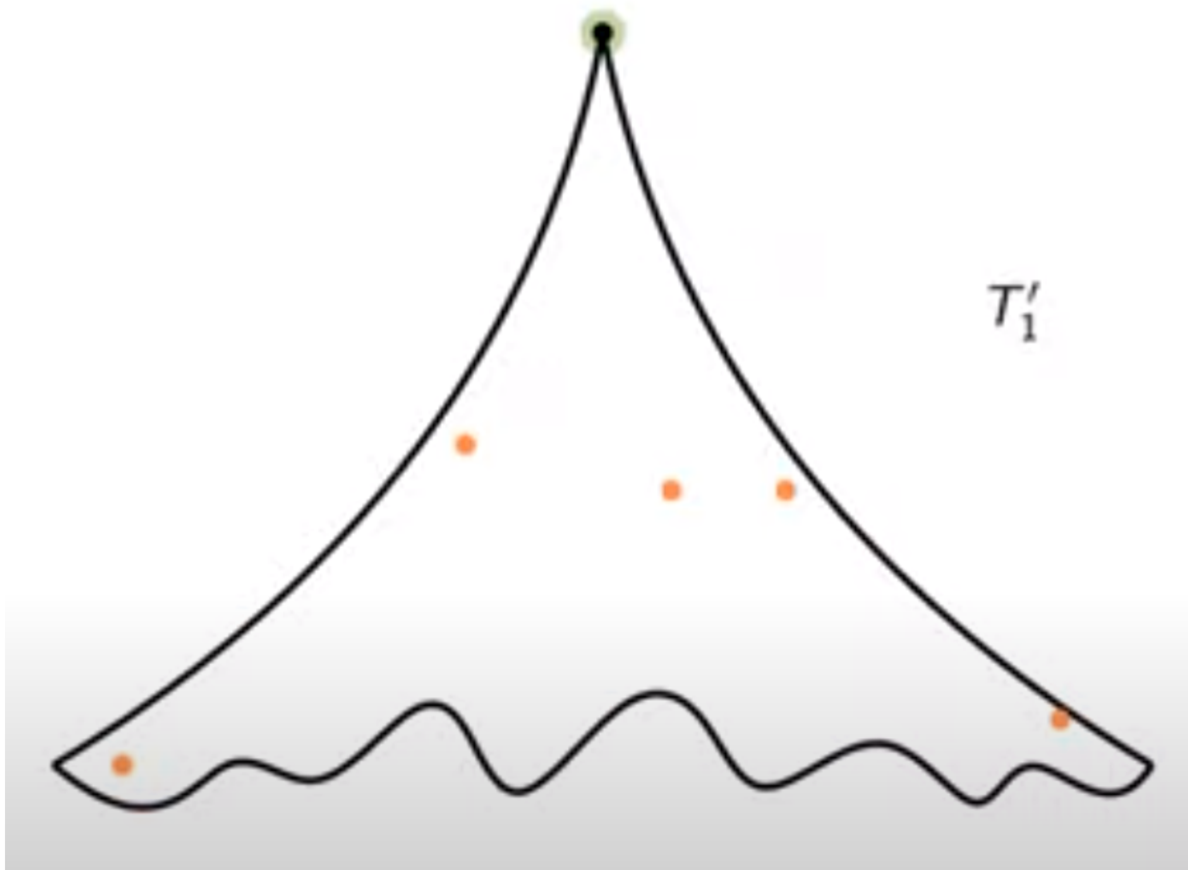
balance(x, k)

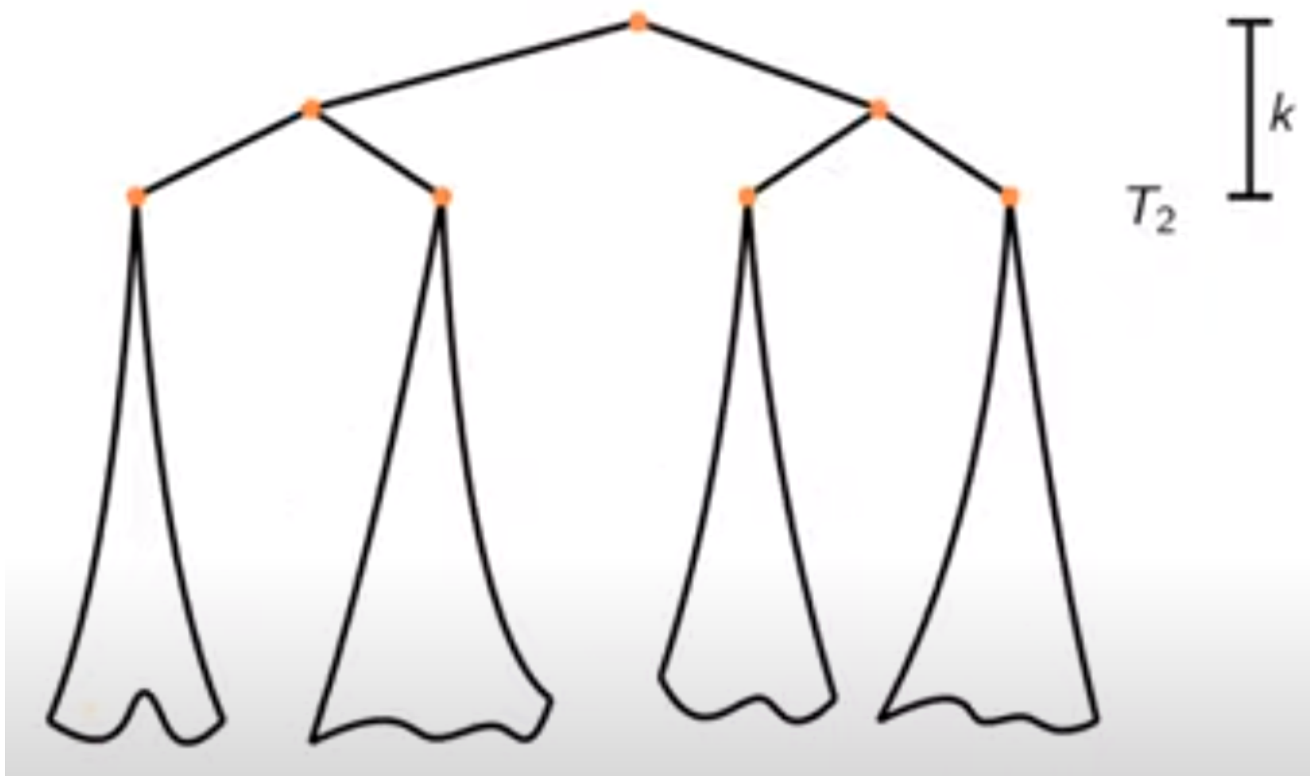
effect on signature can be encoded in $\mathcal{O}(k \log\log n)$ bits



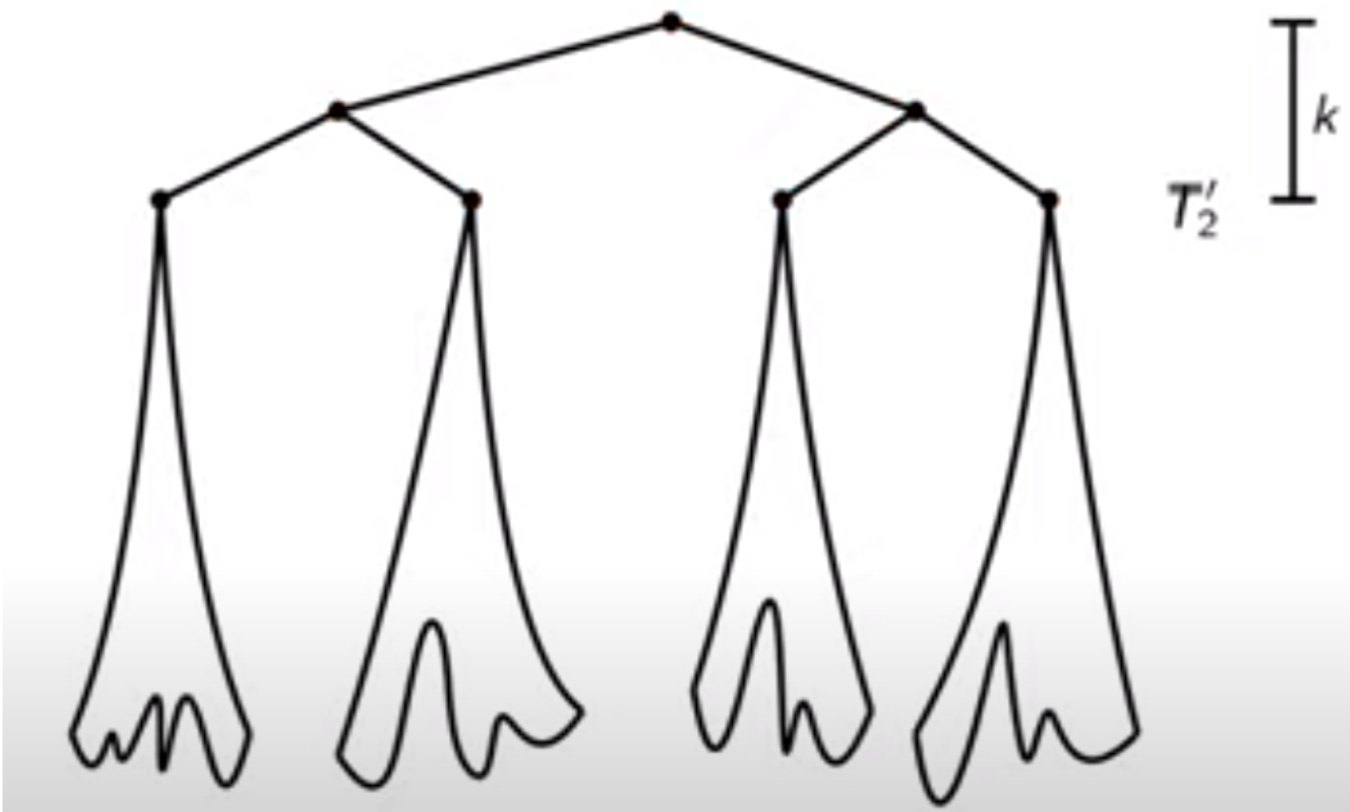

T_1

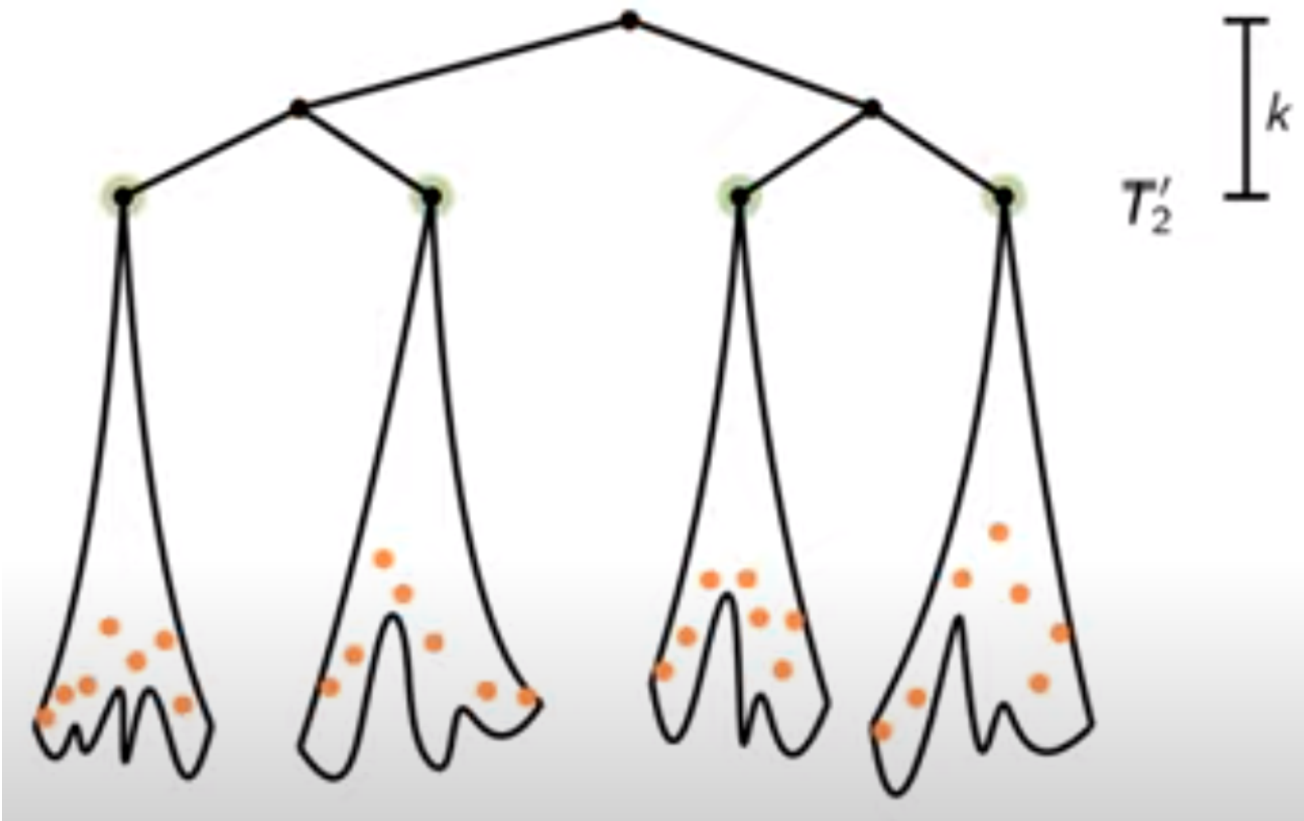

T_1'


T_1'

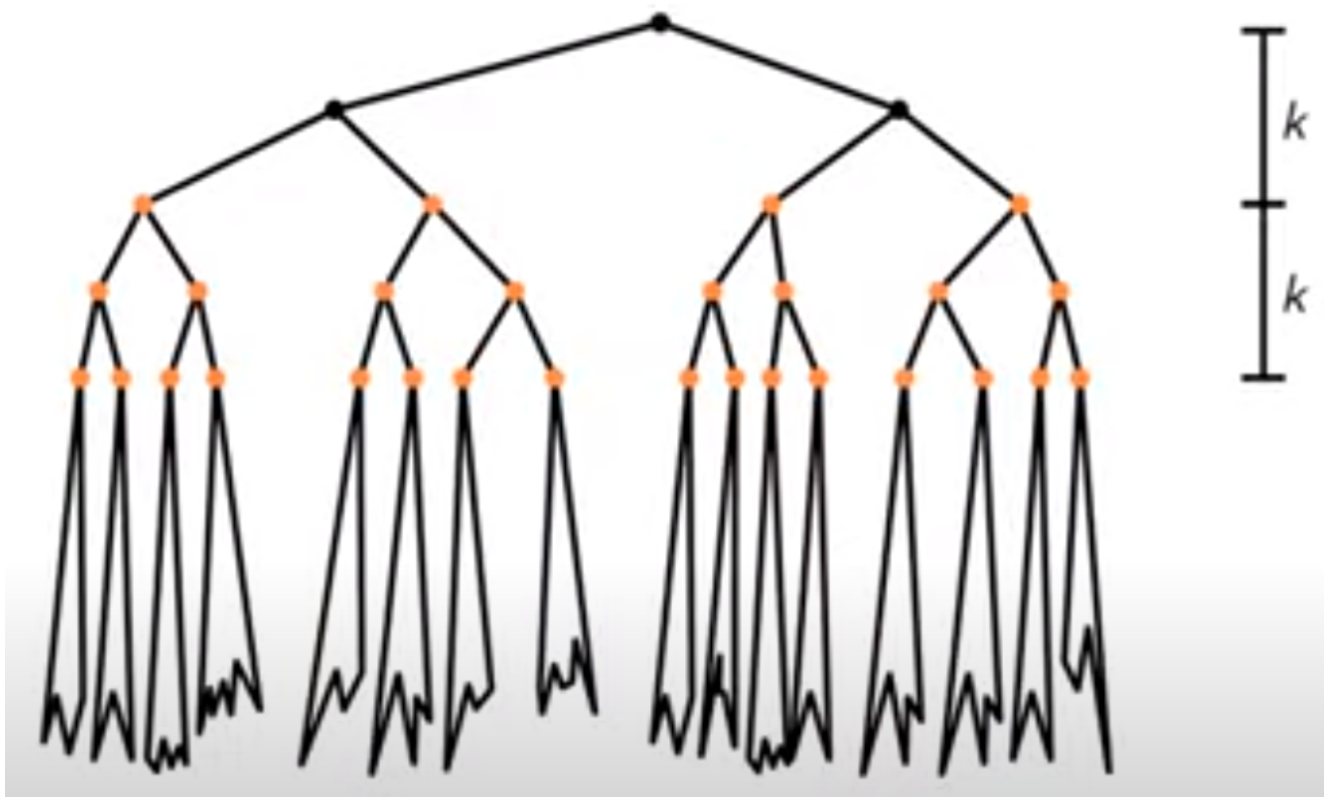


k
T2

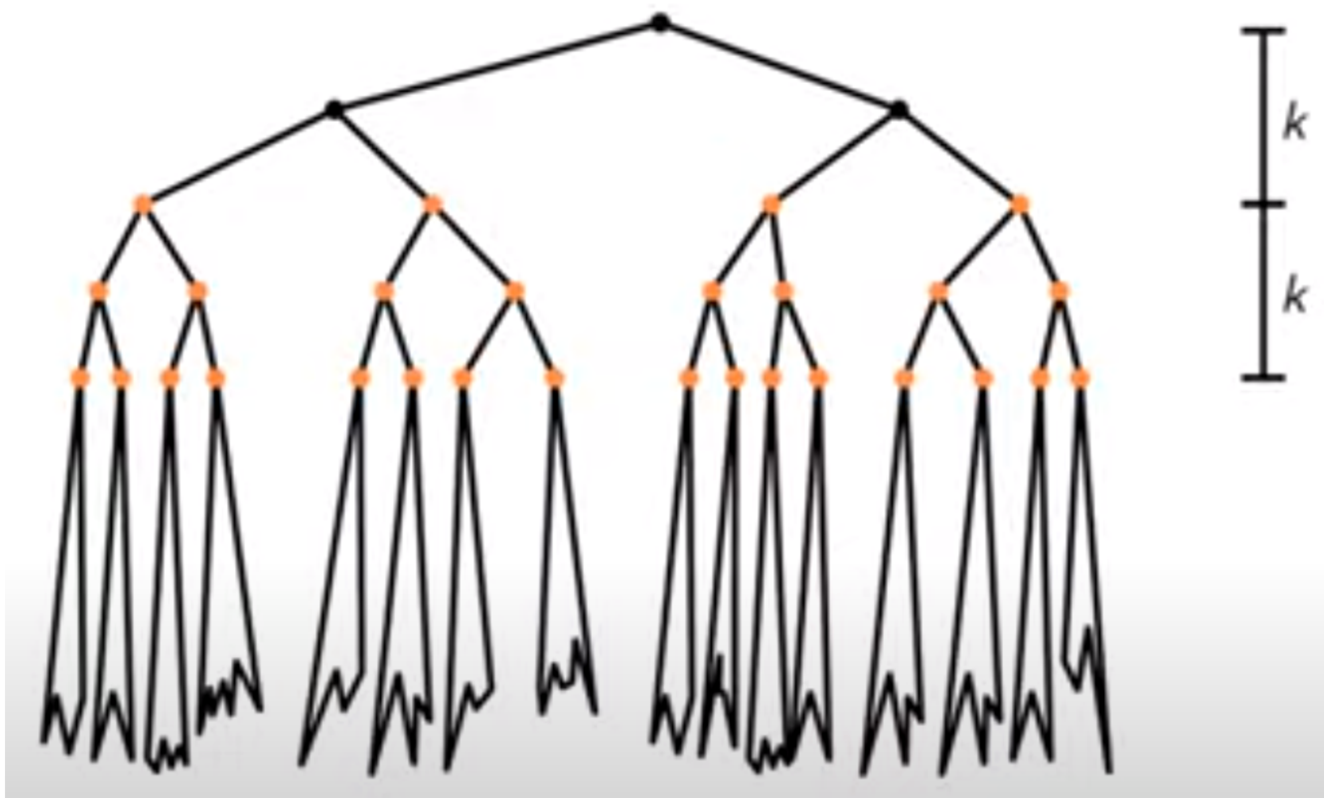

k
T_2'



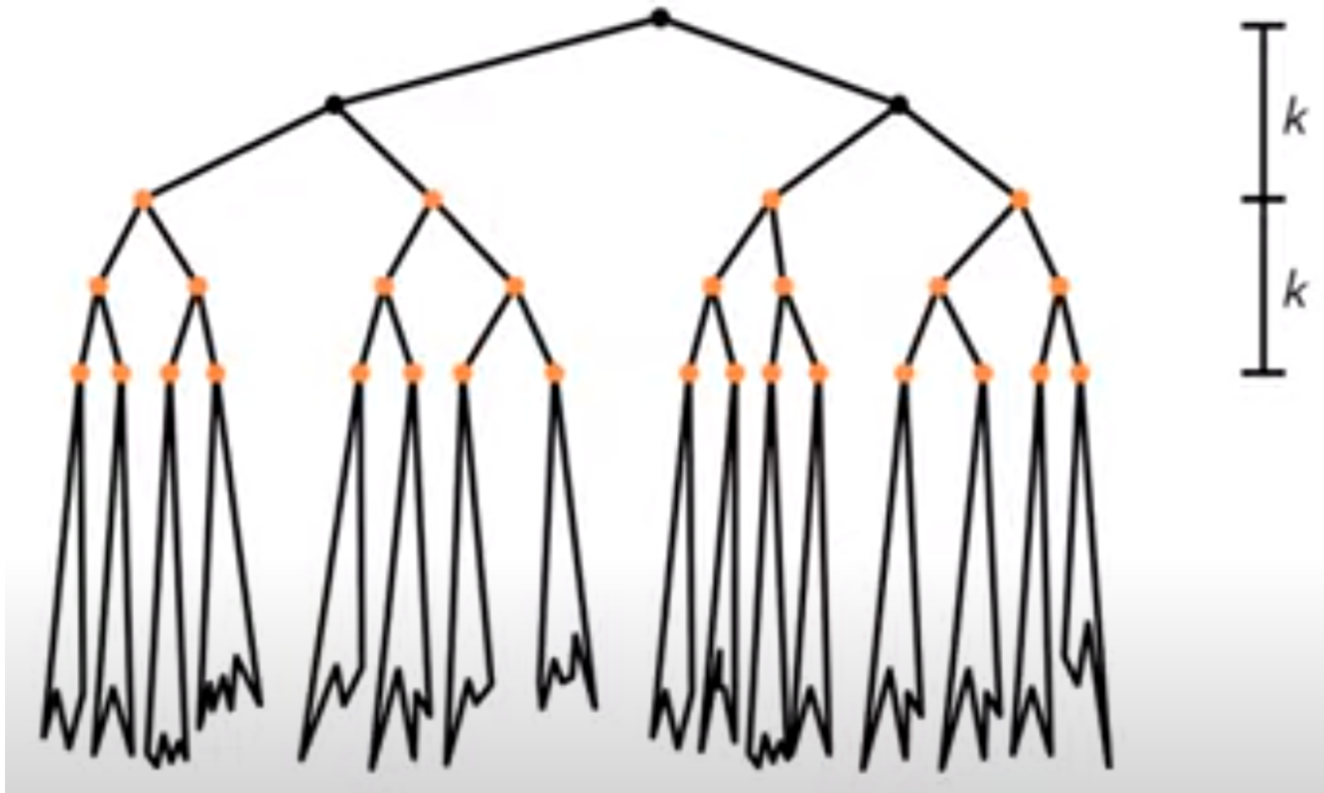
k
T_2'



k
k



$$h(T_i) \leqslant \log |T_i| + \mathcal{O}(\tfrac{1}{k} \log |T_i|)$$

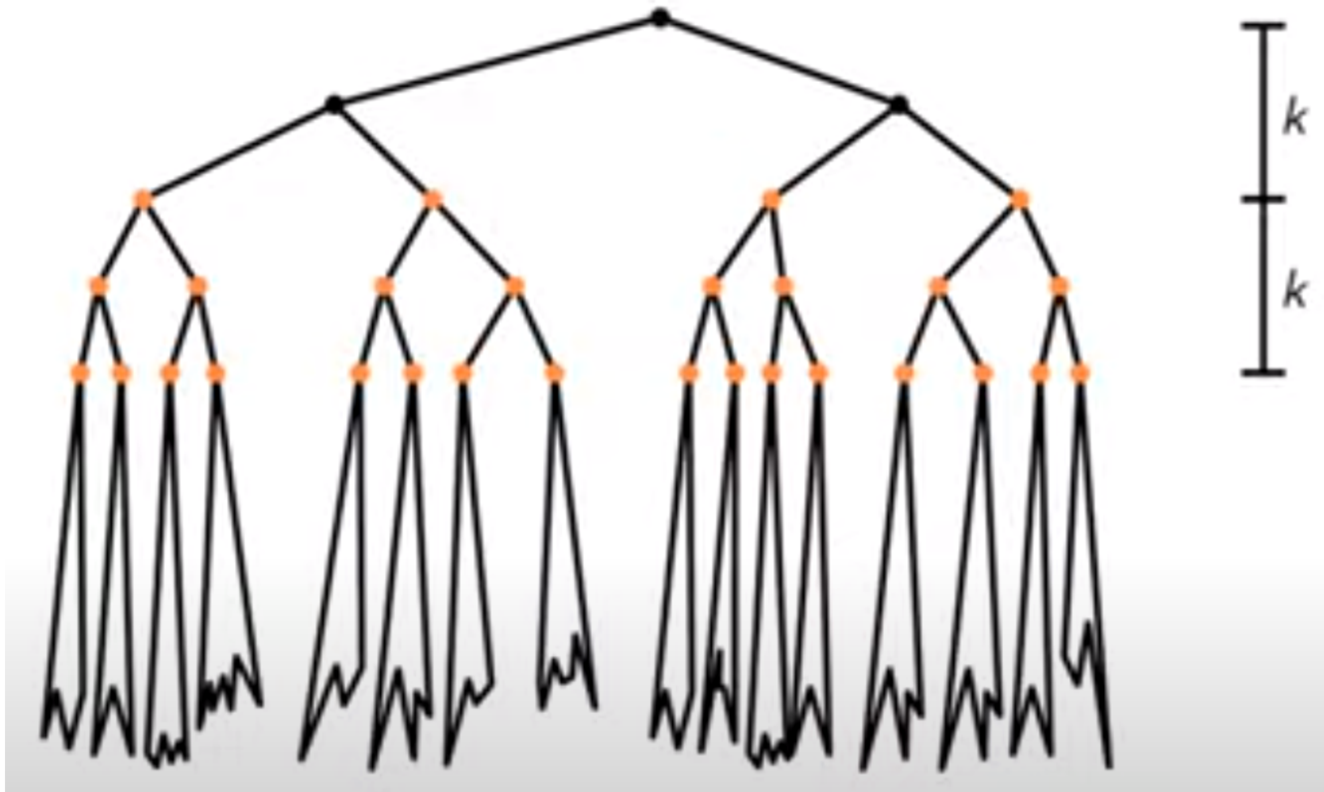


so we have a trade-off:

transition code of length $\mathcal{O}(k \log \log n)$

vs

signatures of length $\log |T_i| + \mathcal{O}(\frac{1}{k} \log |T_i|)$



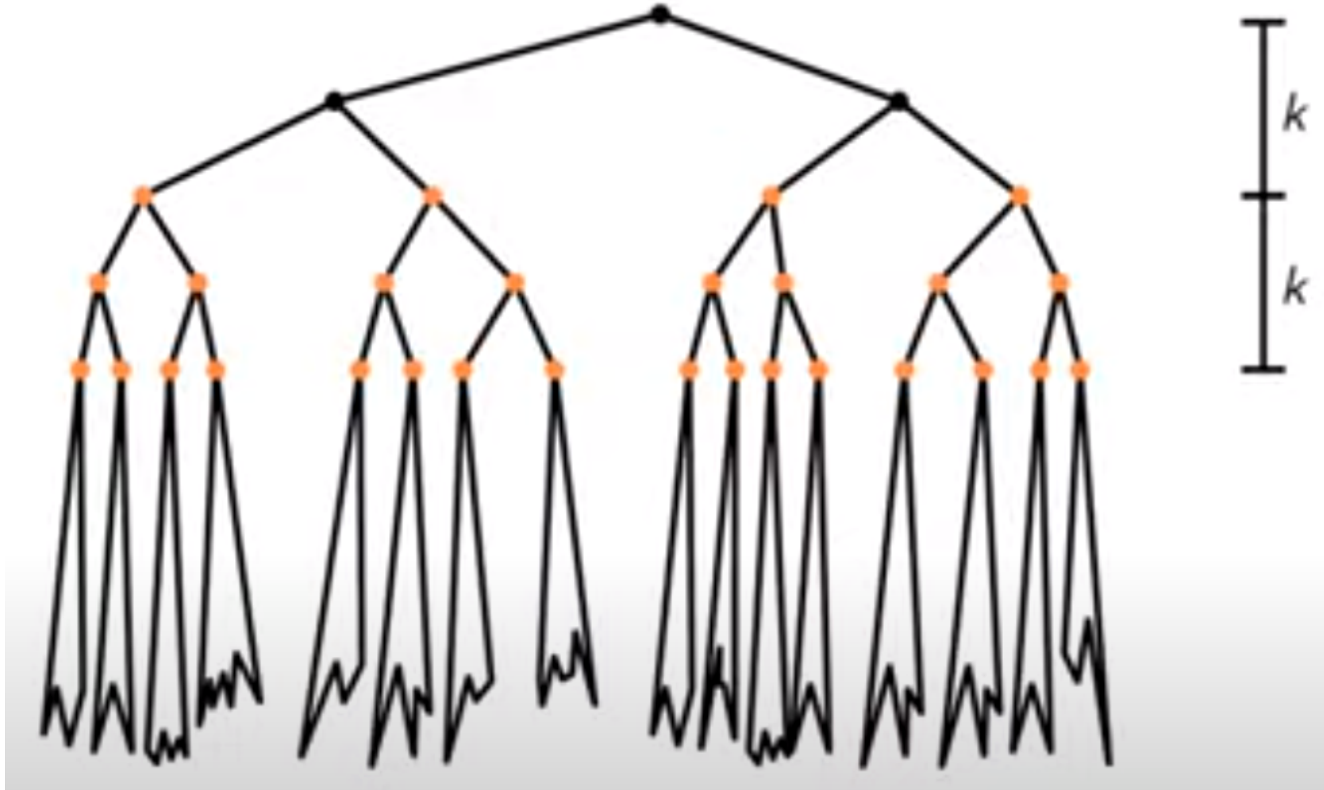
so we have a trade-off:

transition code of length $\mathcal{O}(k \log\log n)$

vs

signatures of length $\log|T_i| + \mathcal{O}(\frac{1}{k}\log|T_i|)$

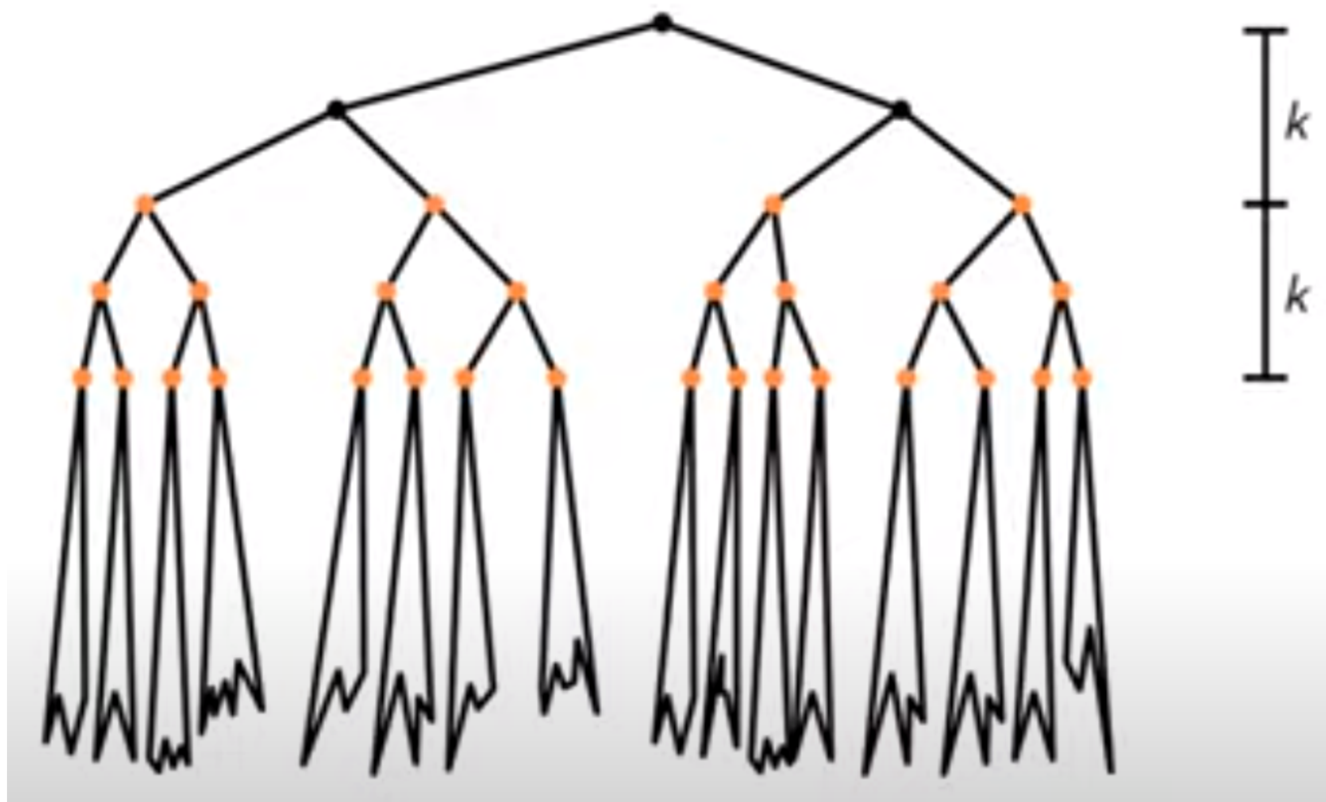
optimized choice $k = \sqrt{\frac{\log n}{\log\log n}}$



resulting labels of length

$$\log n - \log |T_i| + \mathcal{O}(\log\log n)$$
$$\log |T_i| + \mathcal{O}(\sqrt{\log n \log\log n})$$

optimized choice $k = \sqrt{\frac{\log n}{\log\log n}}$



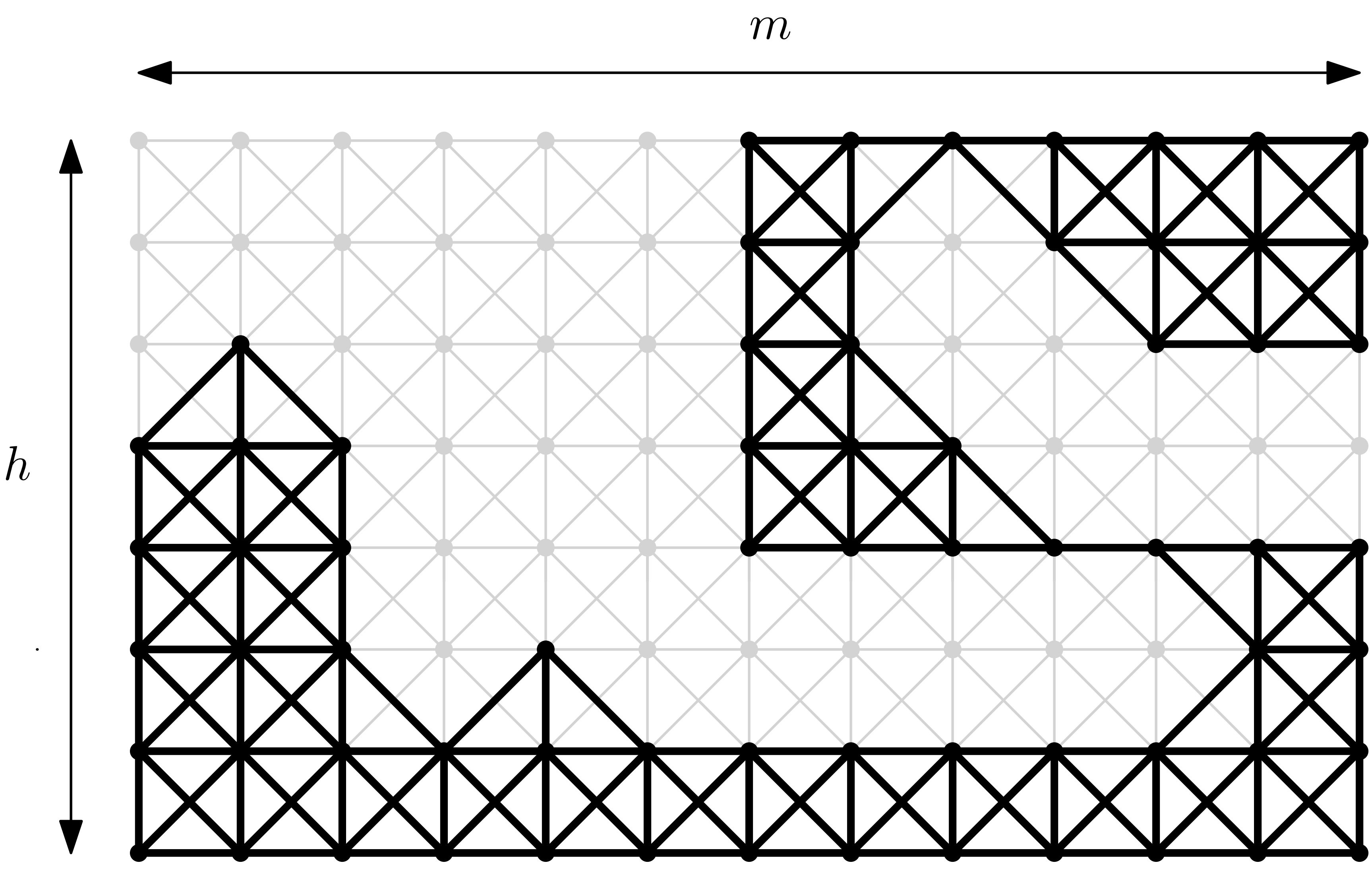
resulting labels of length

$$\log n - \log |T_i| + \mathcal{O}(\log \log n)$$
$$\log |T_i| + \mathcal{O}(\sqrt{\log n \log \log n})$$

optimized choice $k = \sqrt{\frac{\log n}{\log \log n}}$

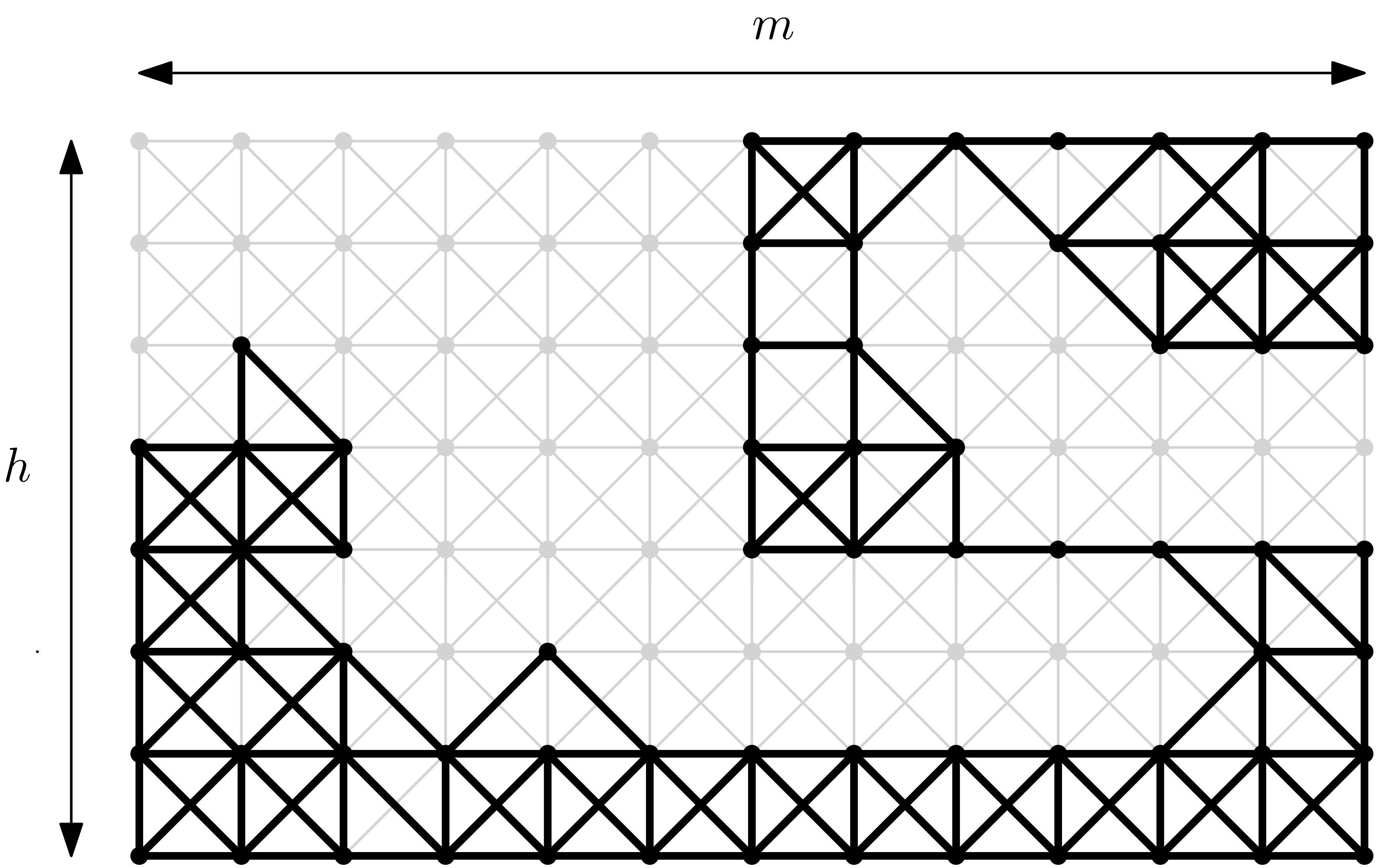
missing pieces?

~~induced~~ subgraphs of $P \boxtimes P$



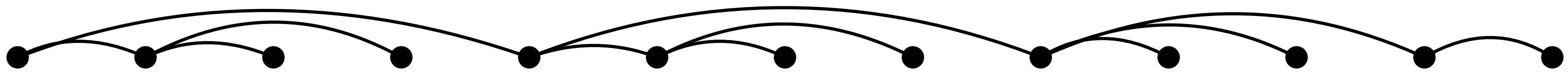
G is an n-vertex subgraph of this

~~induced~~ subgraphs of $P \boxtimes P$

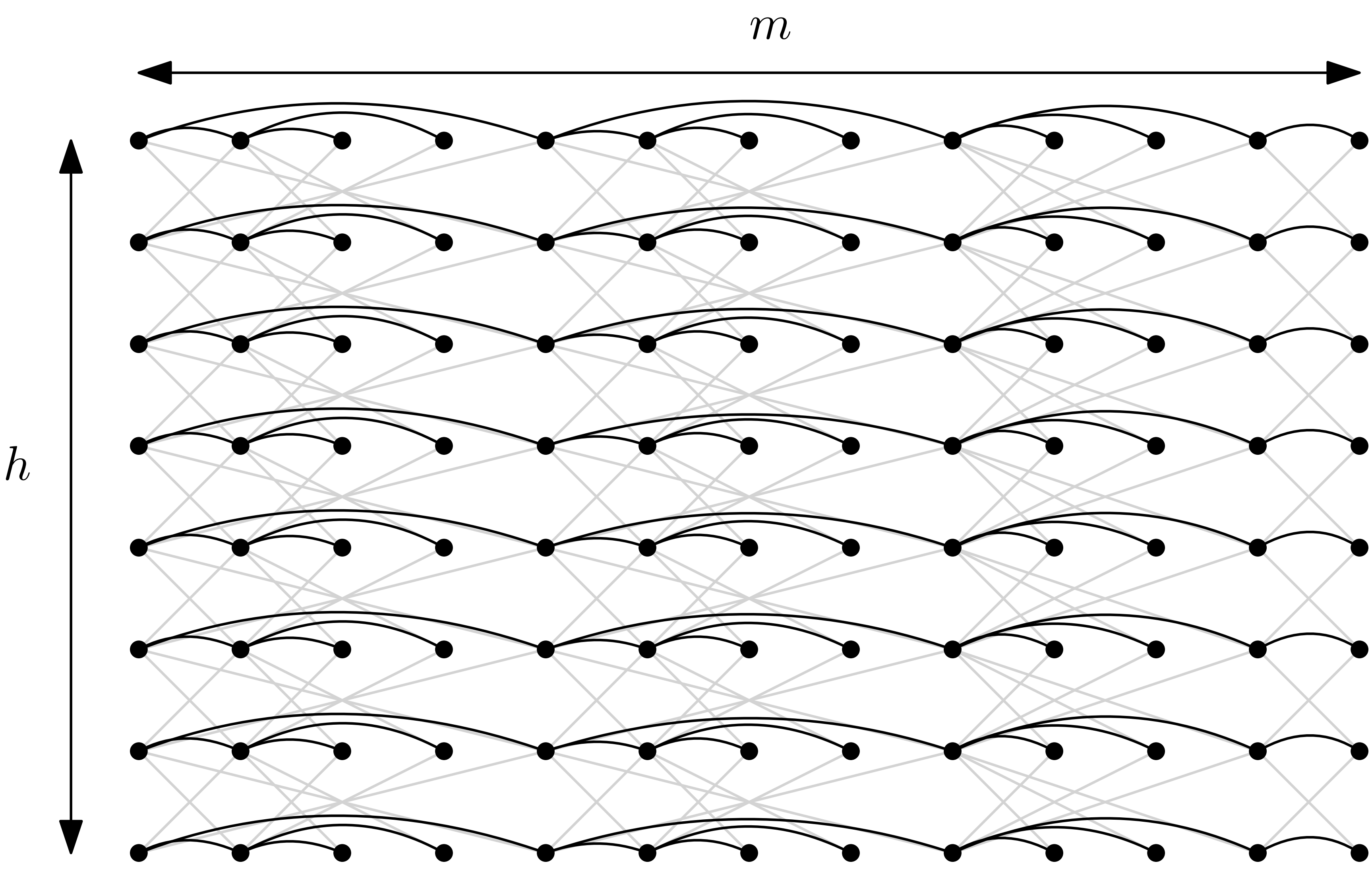


G is an n-vertex subgraph of this

subgraphs of $H \boxtimes P$



subgraphs of $H \boxtimes P$



G is an n-vertex subgraph of this

Observation [Kannan, Naor, Rudich 1988]

A class of graphs $\mathcal{C}$ has an $f(n)$-bit adjacency labelling iff

for each $n \geqslant 1$, there exists a graph U_n such that

- $|V(U_n)| = 2^{f(n)}$;
- G is an induced subgraph of U_n, for each n-vertex G in $\mathcal{C}$.

universal graphs

Observation [Kannan, Naor, Rudich 1988]

A class of graphs $\mathcal{C}$ has an $f(n)$-bit adjacency labelling iff

for each $n \geqslant 1$, there exists a graph U_n such that

- ▷ $|V(U_n)| = 2^{f(n)}$;
- ▷ G is an induced subgraph of U_n, for each n-vertex G in $\mathcal{C}$.

Proof.

$$V(U_n) = \{0,1\}^{f(n)}$$
$$E(U_n) = \{uv \mid A(u,v) = 1\}$$

Observation [Kannan, Naor, Rudich 1988]

A class of graphs $\mathcal{C}$ has an $f(n)$-bit adjacency labelling iff

for each $n \geqslant 1$, there exists a graph U_n such that

- $|V(U_n)| = 2^{f(n)}$;
- G is an induced subgraph of U_n, for each n-vertex G in $\mathcal{C}$.

Proof.

$$V(U_n) = \{0,1\}^{f(n)}$$
$$E(U_n) = \{uv \mid A(u,v) = 1\}$$

Corollary

n-vertex planar graphs have a universal graph on $n^{1+o(1)}$ vertices

universal graphs

Observation [Kannan, Naor, Rudich 1988]

A class of graphs $\mathcal{C}$ has an $f(n)$-bit adjacency labelling iff

for each $n \geqslant 1$, there exists a graph U_n such that

- $|V(U_n)| = 2^{f(n)}$;
- G is an induced subgraph of U_n, for each n-vertex G in $\mathcal{C}$.

Proof.

$$V(U_n) = \{0,1\}^{f(n)}$$
$$E(U_n) = \{uv \mid A(u,v) = 1\}$$

Corollary

n-vertex planar graphs have a universal graph on $n^{1+o(1)}$ vertices

Theorem [Esperet, Joret, Morin 2020+]

n-vertex planar graphs have a universal graph on $n^{1+o(1)}$ vertices and $n^{1+o(1)}$ edges

open problems

▷ what is the asymptotics of the lower order term?

$$\log n + \mathcal{O}(\sqrt{\log n \log\log n})$$
$$+\Omega(1)$$

open problems

- ▷ what is the asymptotics of the lower order term?

$$\log n + \mathcal{O}(\sqrt{\log n \log\log n})$$
$$+\Omega(1)$$

- ▷ adjacency labelling for K_t-minor free graphs?

$$2\log n + o(\log n)$$

open problems

- $\triangleright$ what is the asymptotics of the lower order term?

$$\log n + \mathcal{O}(\sqrt{\log n \log\log n})$$
$$+\Omega(1)$$

- $\triangleright$ adjacency labelling for K_t-minor free graphs?

$$2\log n + o(\log n)$$

Thank you.