Techniques for Grammar-Based Compression

Philip Bille

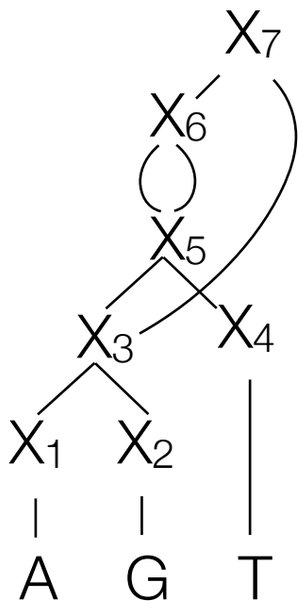
Outline

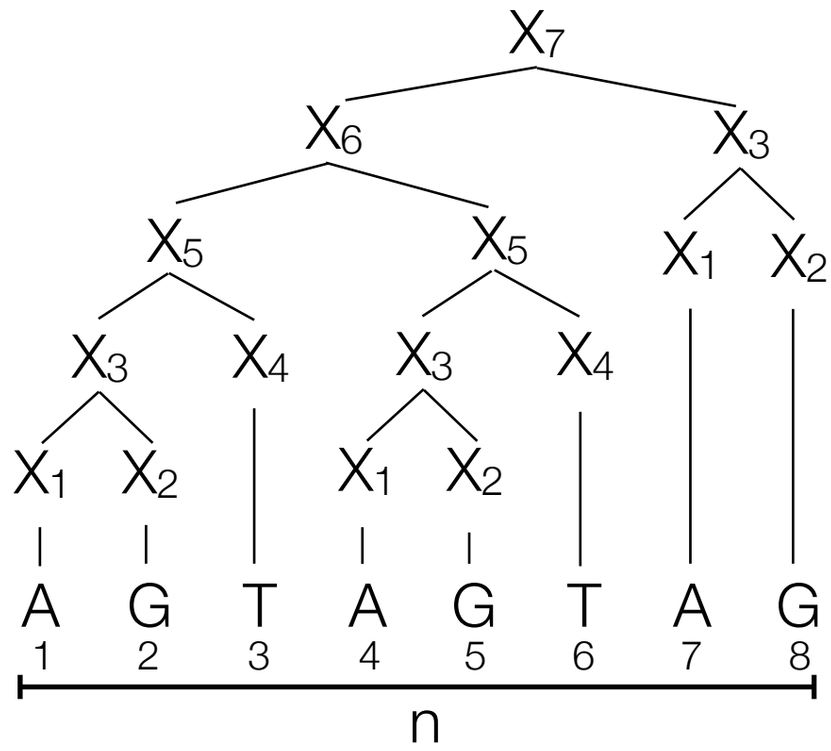
- Grammars
- Random access
- Finger search
- Bookmarking
- Applications

Grammar Compression

AGTAGTAG n = 8

$X_7 \rightarrow X_6X_3$
$X_6 \rightarrow X_5X_5$
$X_5 \rightarrow X_3X_4$
$X_4 \rightarrow \mathbf{T}$
$X_3 \rightarrow X_1X_2$
$X_2 \rightarrow \mathbf{G}$
$X_1 \rightarrow \mathbf{A}$

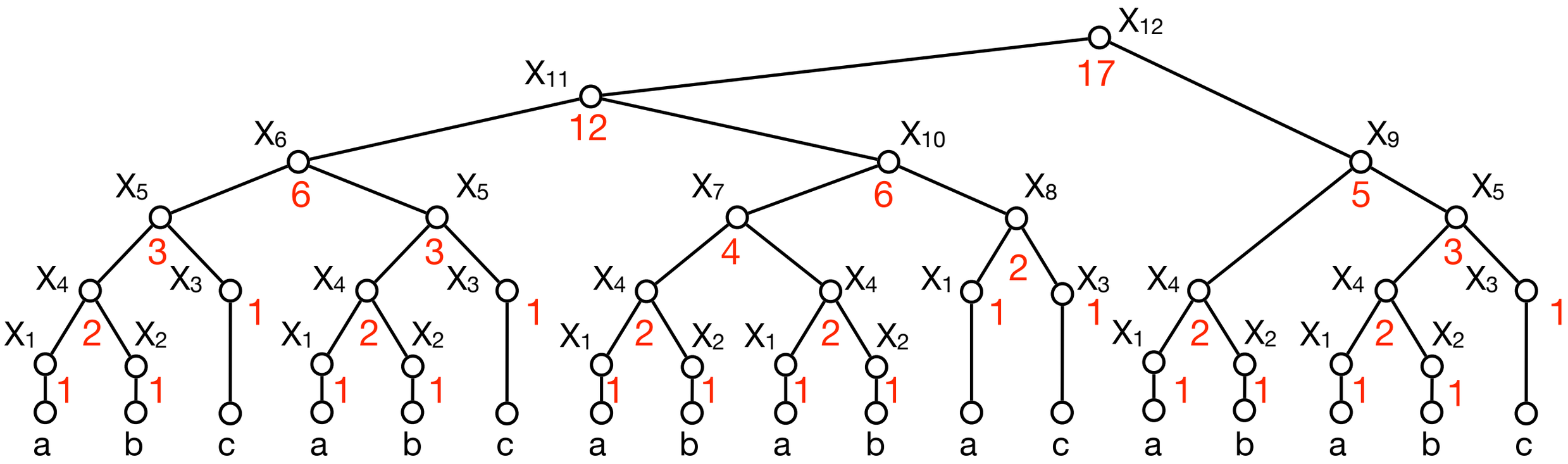




- **Random access problem:** Compactly represent grammar of size g to support
  - access(i): return S[i]
  - decompress(i,ℓ): return S[i, i + ℓ]

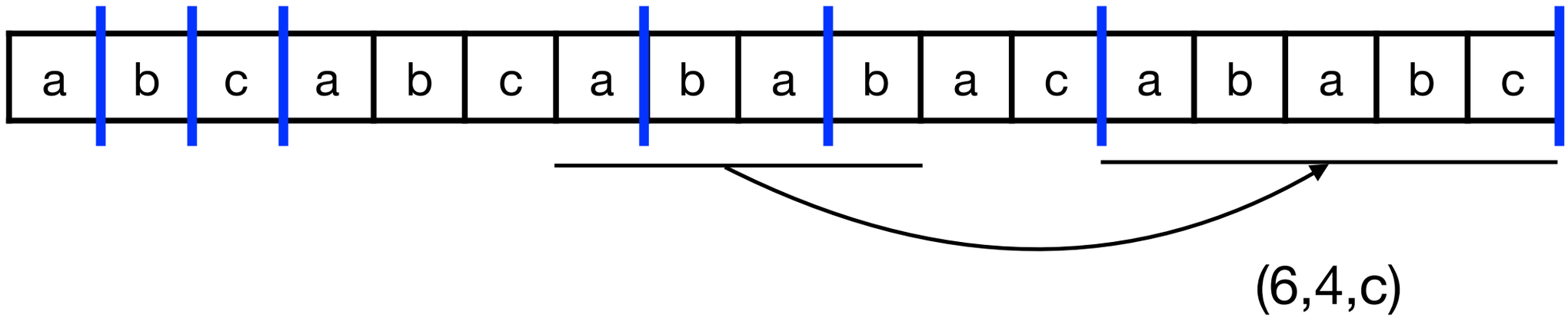
Random Access

Traversal



- O(g) space and O(h) time.

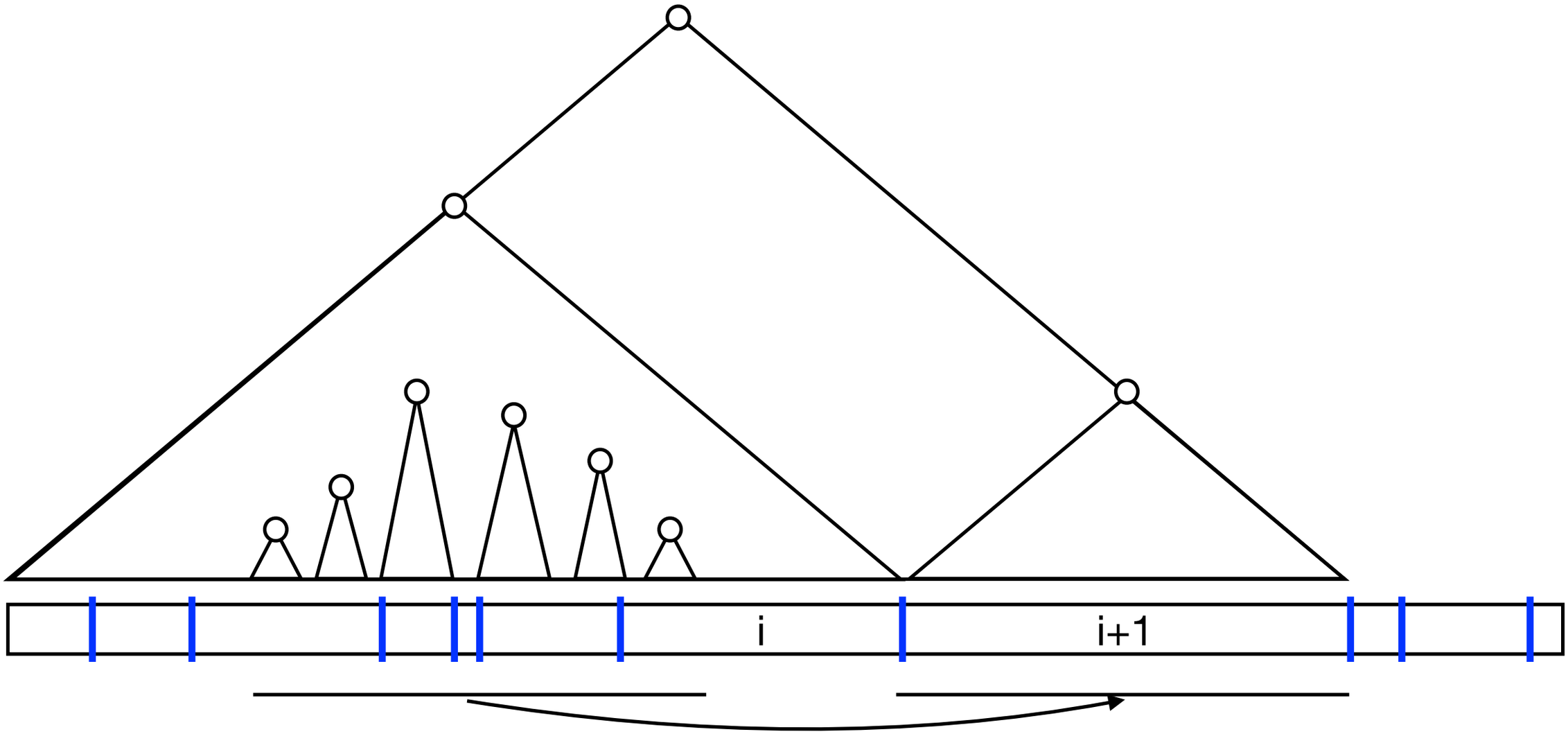
Lempel-Ziv



Balanced Grammars

[Rytter TCS 2003]

[Charikar, Lehman, Liu, Panigrahy, Prabhakaran, Sahai, Shelat Inf. The. 2005]



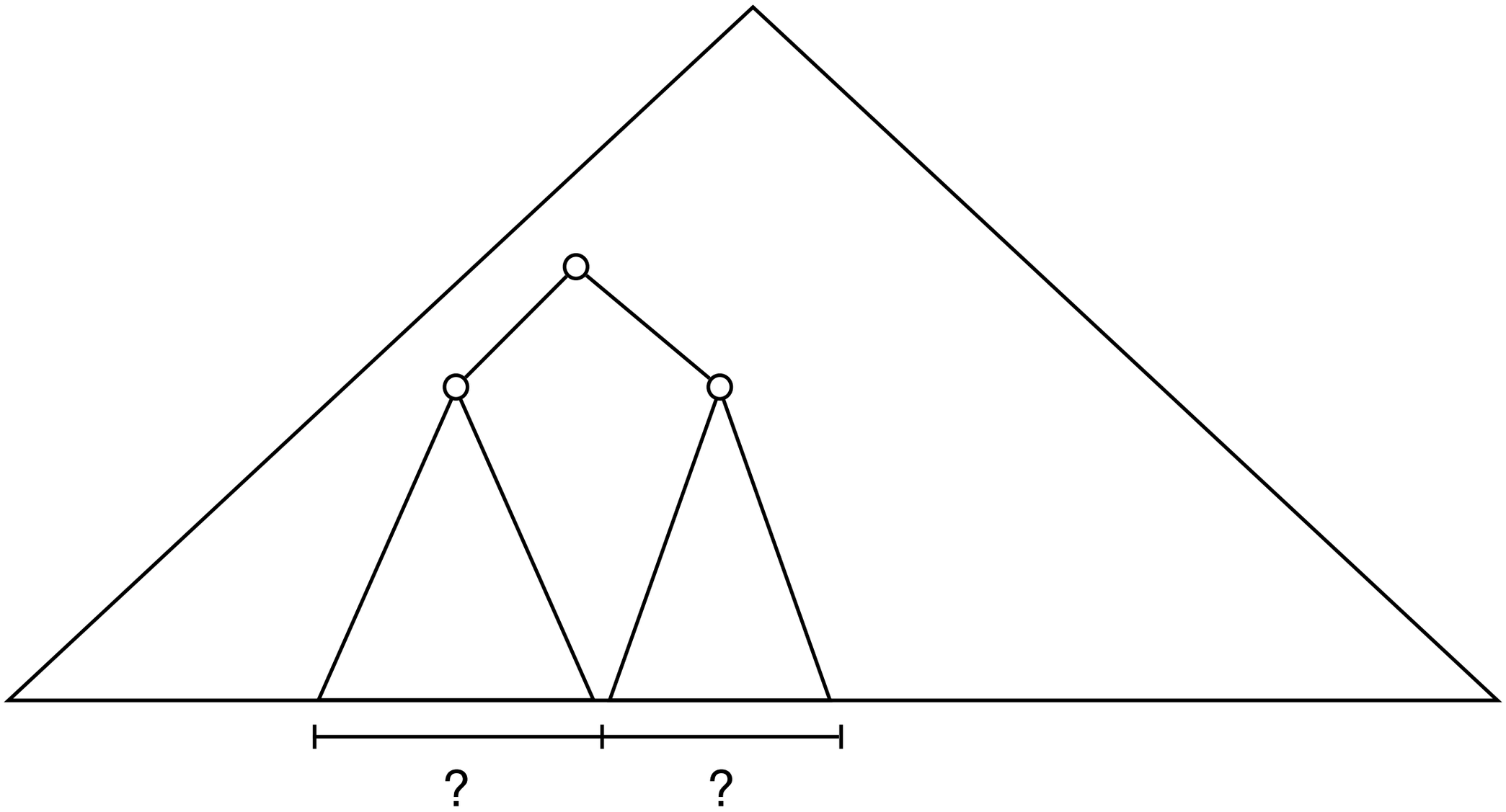
- Concat(A, B) adds O(|height(A) - height(B)|) new non-terminals
- ⇒ O(log n) new non-terminals per phrase

⇒ $g_{new} = O(z \log n) = O(g \log n)$

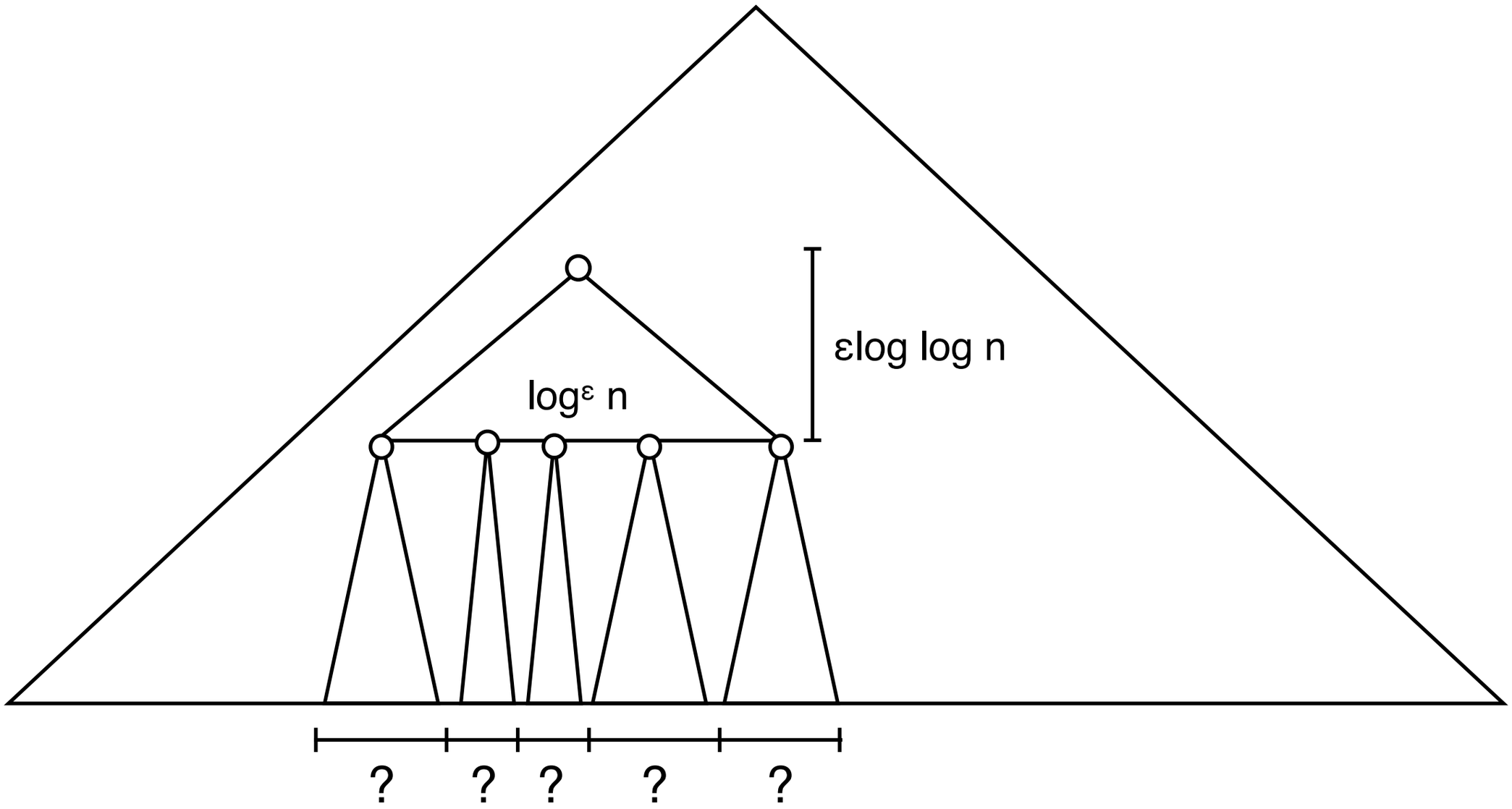
- O(g log n) space and O(log n) time.

Fusion Tree Speedup

[Belazzougui, Landau, Cording, Puglisi, Tabei ESA 2015]



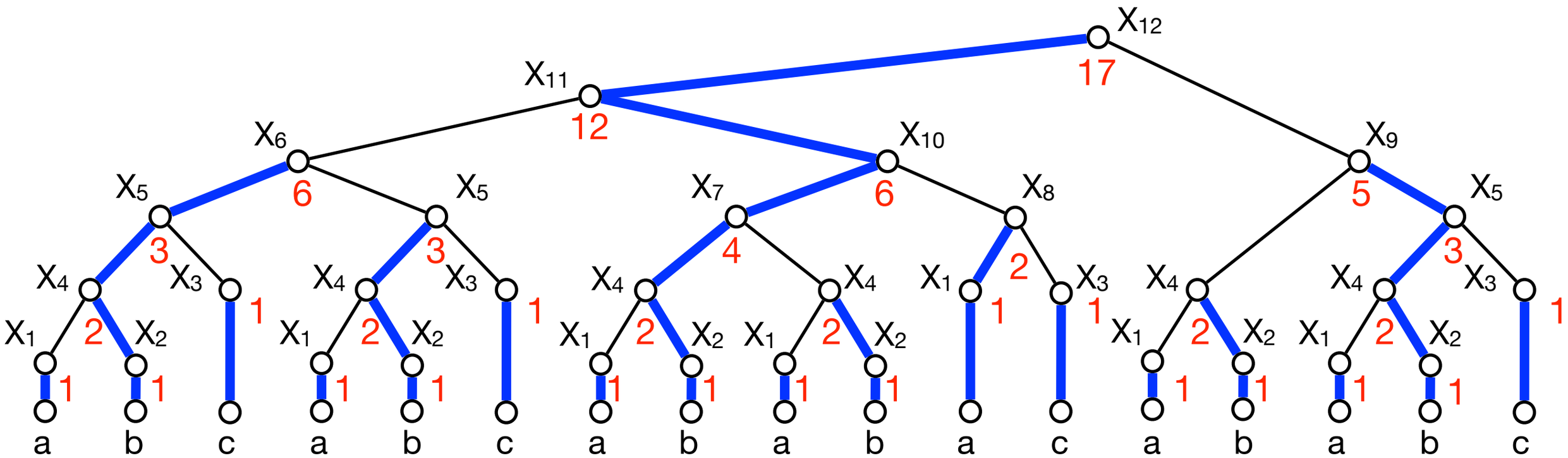
Fusion Tree Speedup



- O(g $\log^{1+\varepsilon} n$) space and O(log n / log log n) time.

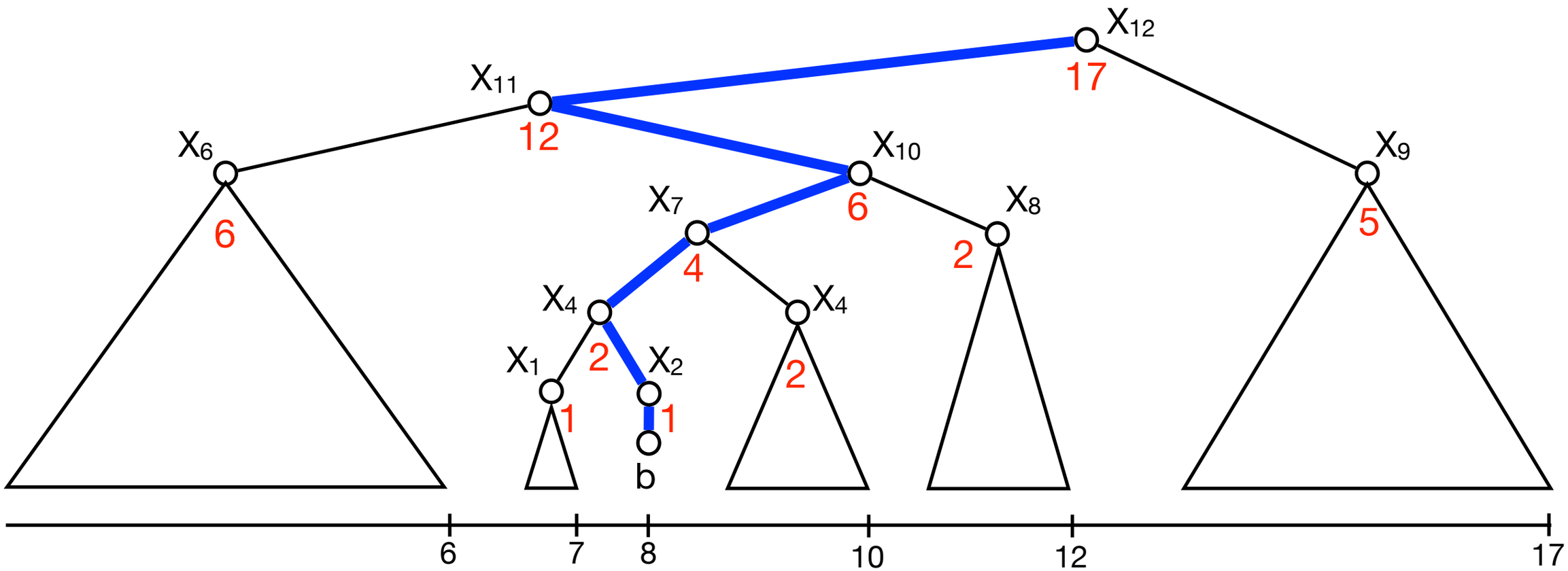
Heavy Paths

[B,Landau, Raman, Sadakane, Satti, Weimann, SICOMP 2015]



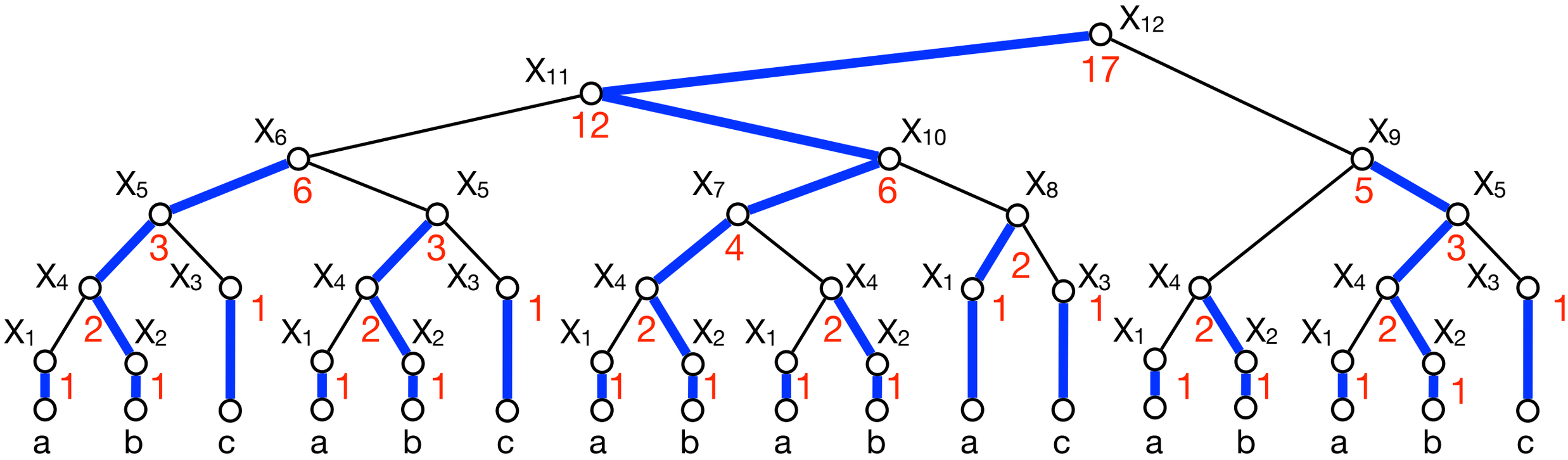
- Number of heavy paths on any root-to-leaf path is O(log n)

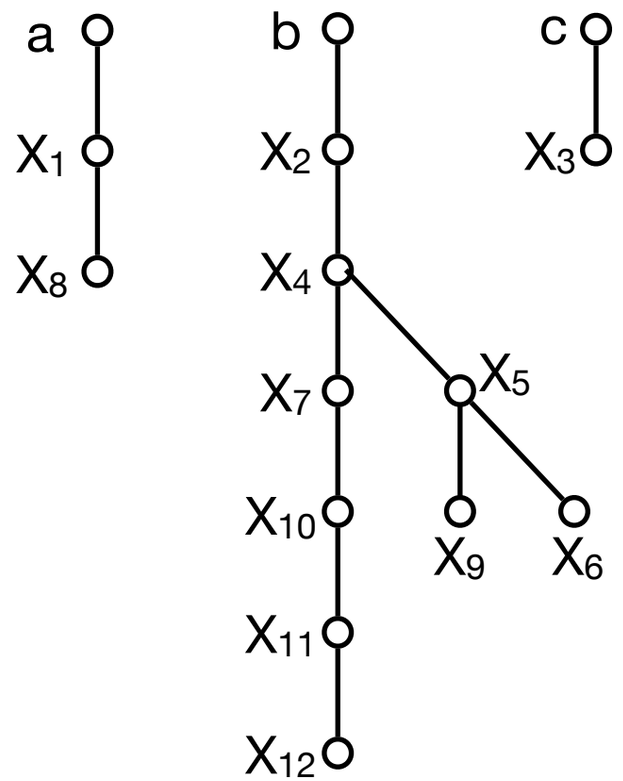
Heavy Paths



- O(log n log log n) time and O(g^2) space.

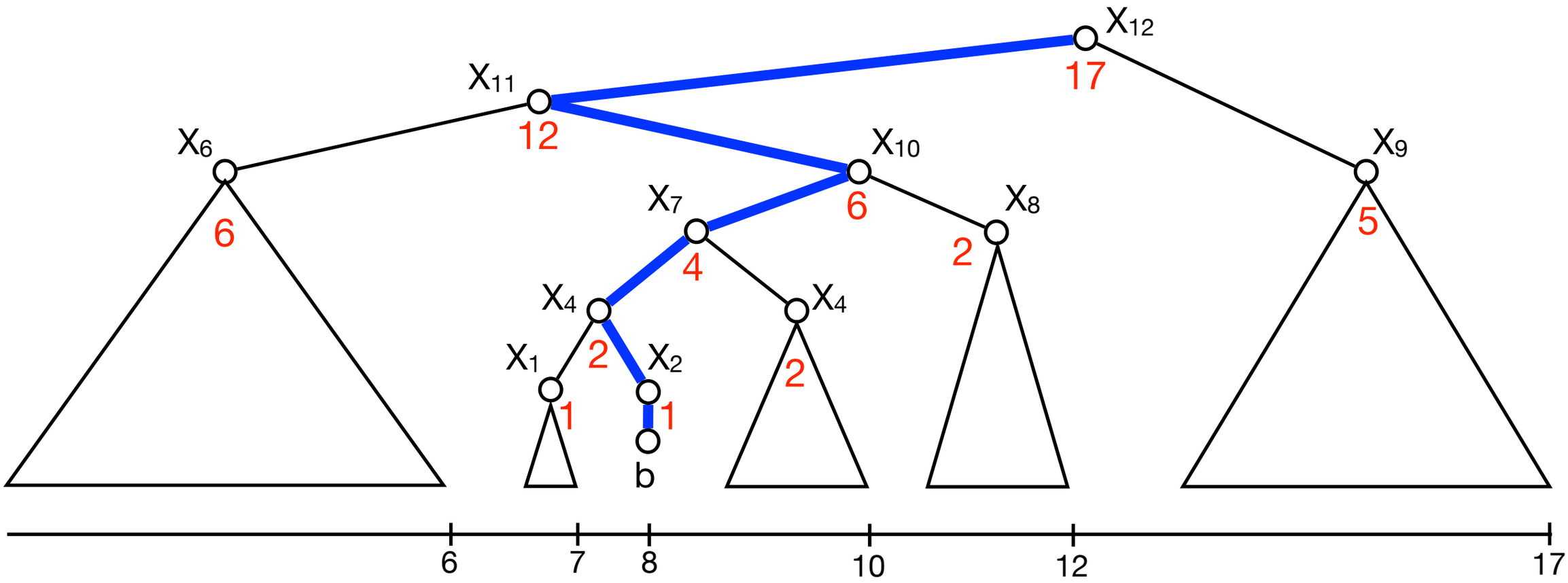
Heavy Path Redundancy





- Predecessor = Weighted ancestor.
- ⇒ O(log n log log n) time and O(g) space.

Biased Search



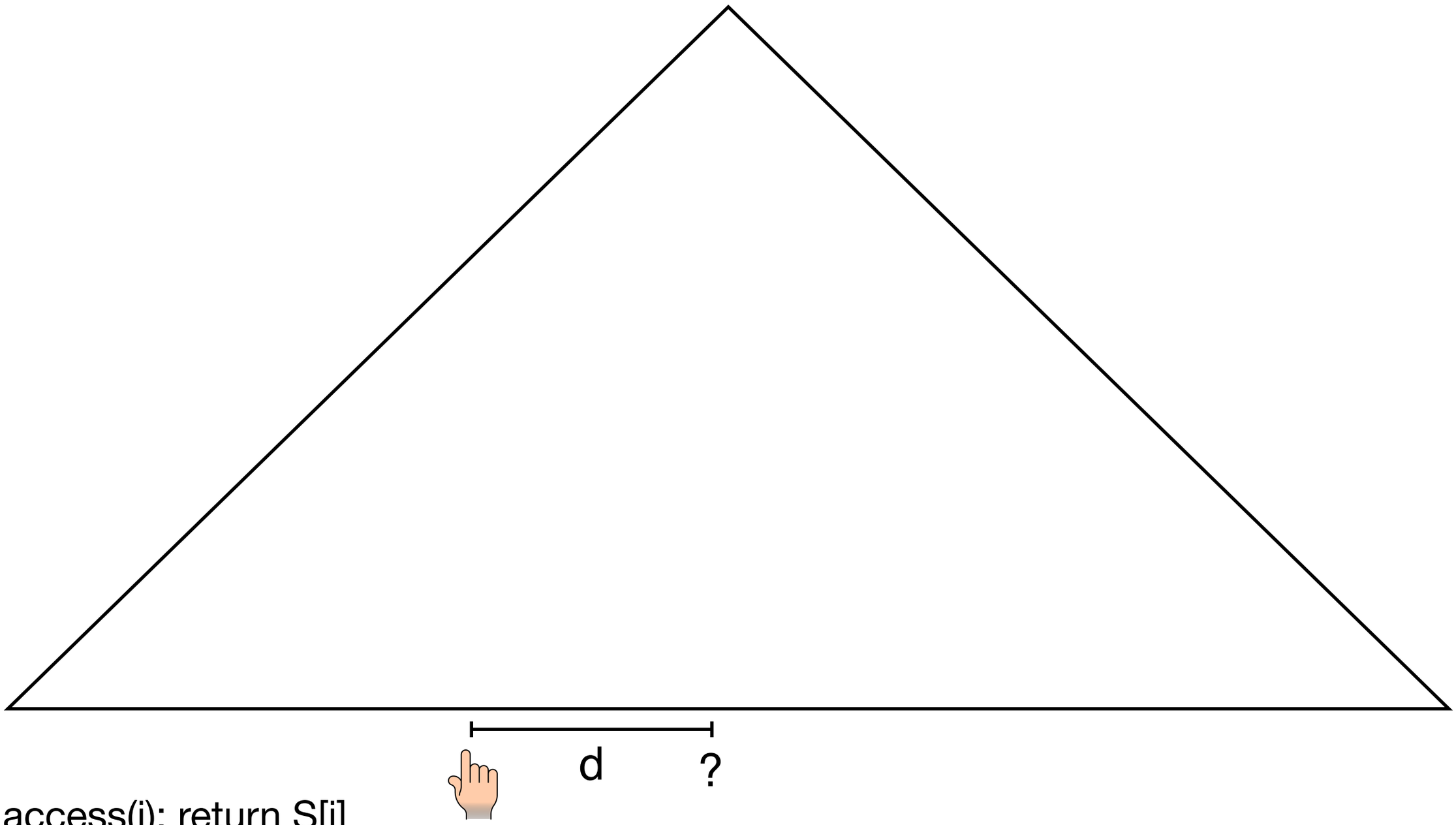
- Biased search in O(log (n/S)) time.
- log (n/S_1) + log (S_1/S_2) + log (S_2/S_3) + log (S_3/S_4) + ... + O(1) = O(log n)
- ⇒ O(log n) time and O(g) space.

Space	Time	Reference
O(g)	O(h)	
O(g log n)	O(log n)	[R2003, CELRPRSS2002]
O(g $\log^{O(1)}$ n)	Ω(log n / log log n)	[VY2013]
O(g $\log^{1+\varepsilon}$ n)	O(log n/ log log n)	[BCPT2015]
O(g)	O(log n)	[BLRSSW2011]

Finger Search

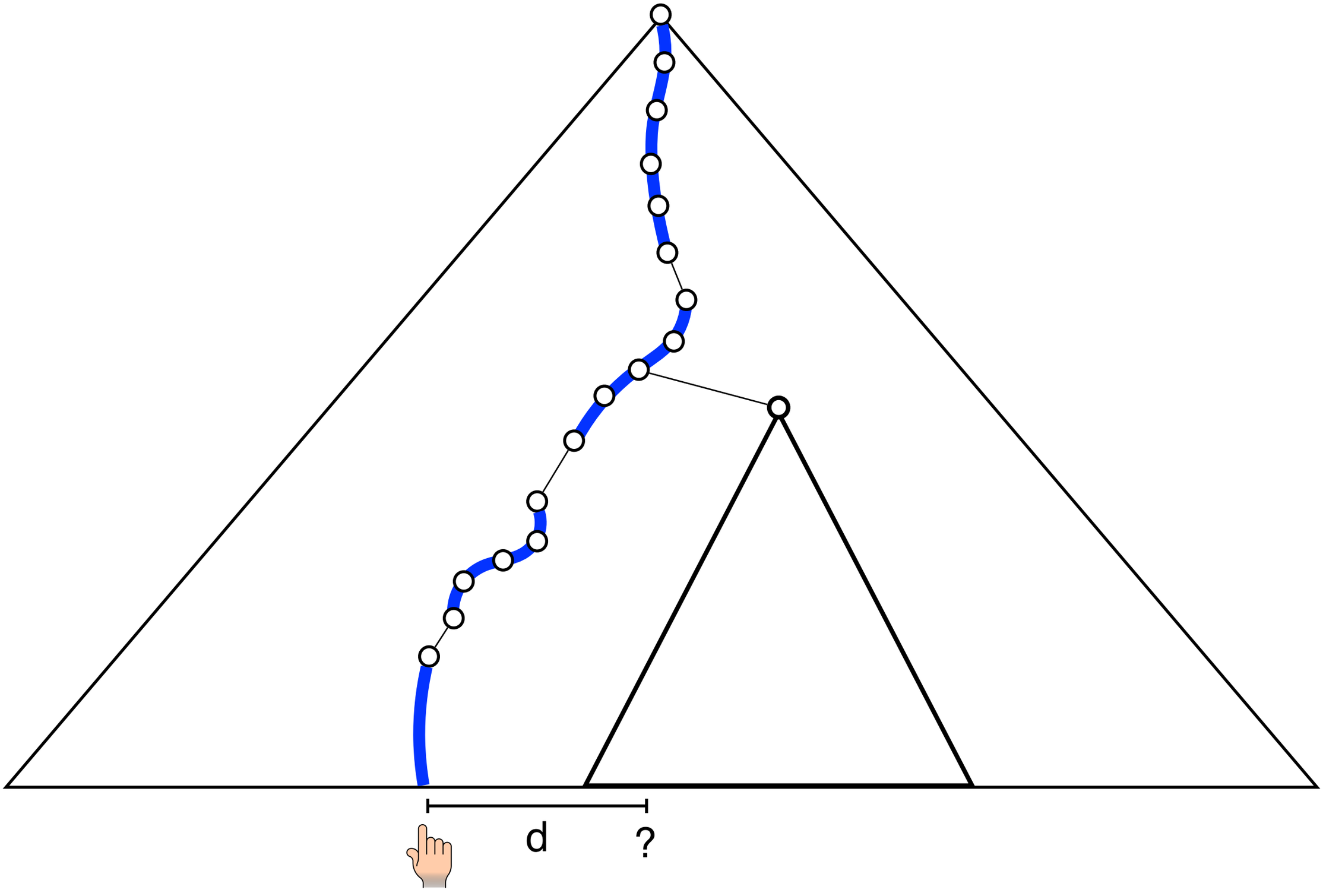
Finger Search

[B.,Christiansen, Cording, Gørtz, TOCS 2018]

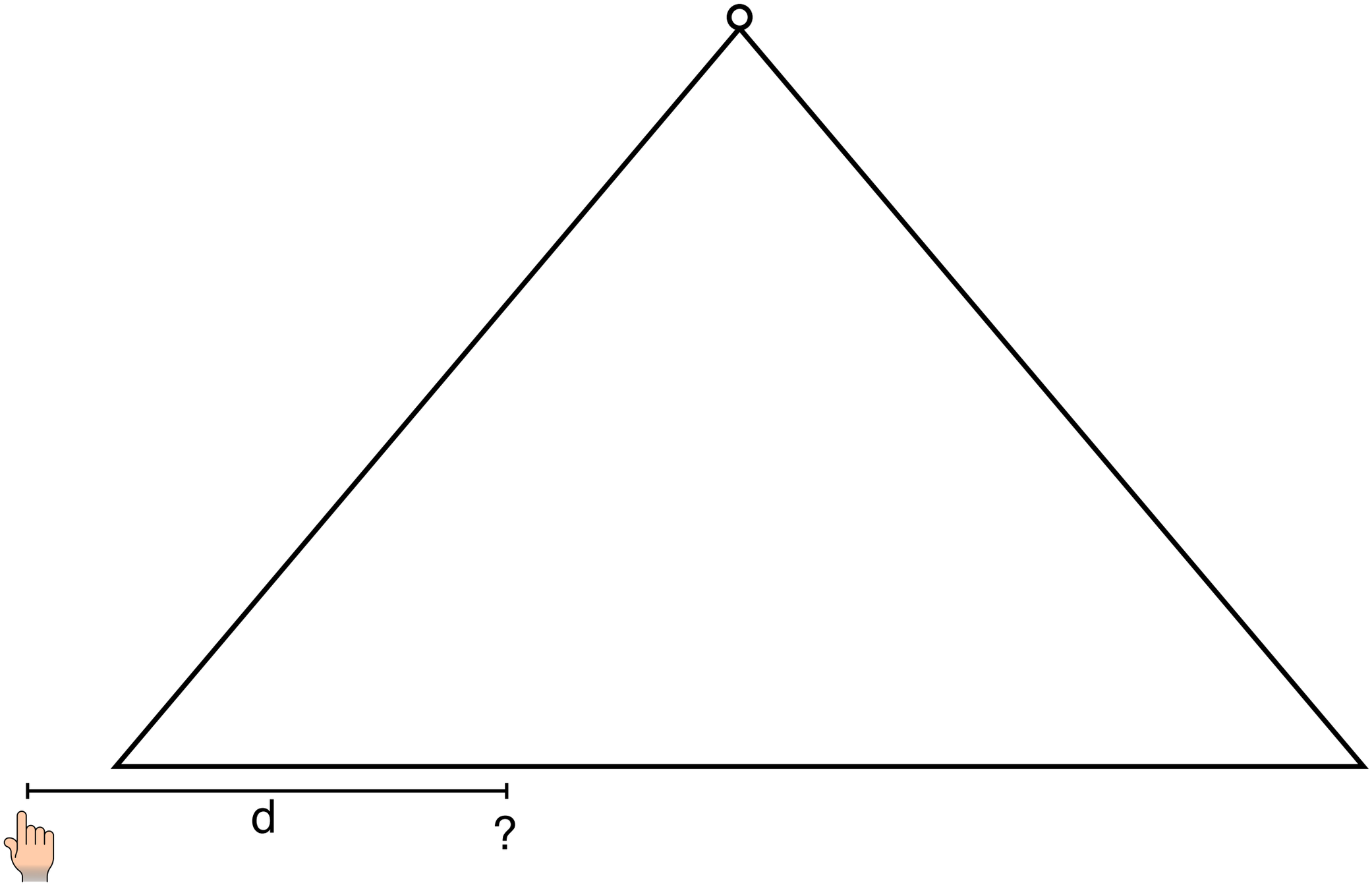


- access(i): return S[i]
- setfinger(f): place finger on f
- movefinger(f): move finger to position f.

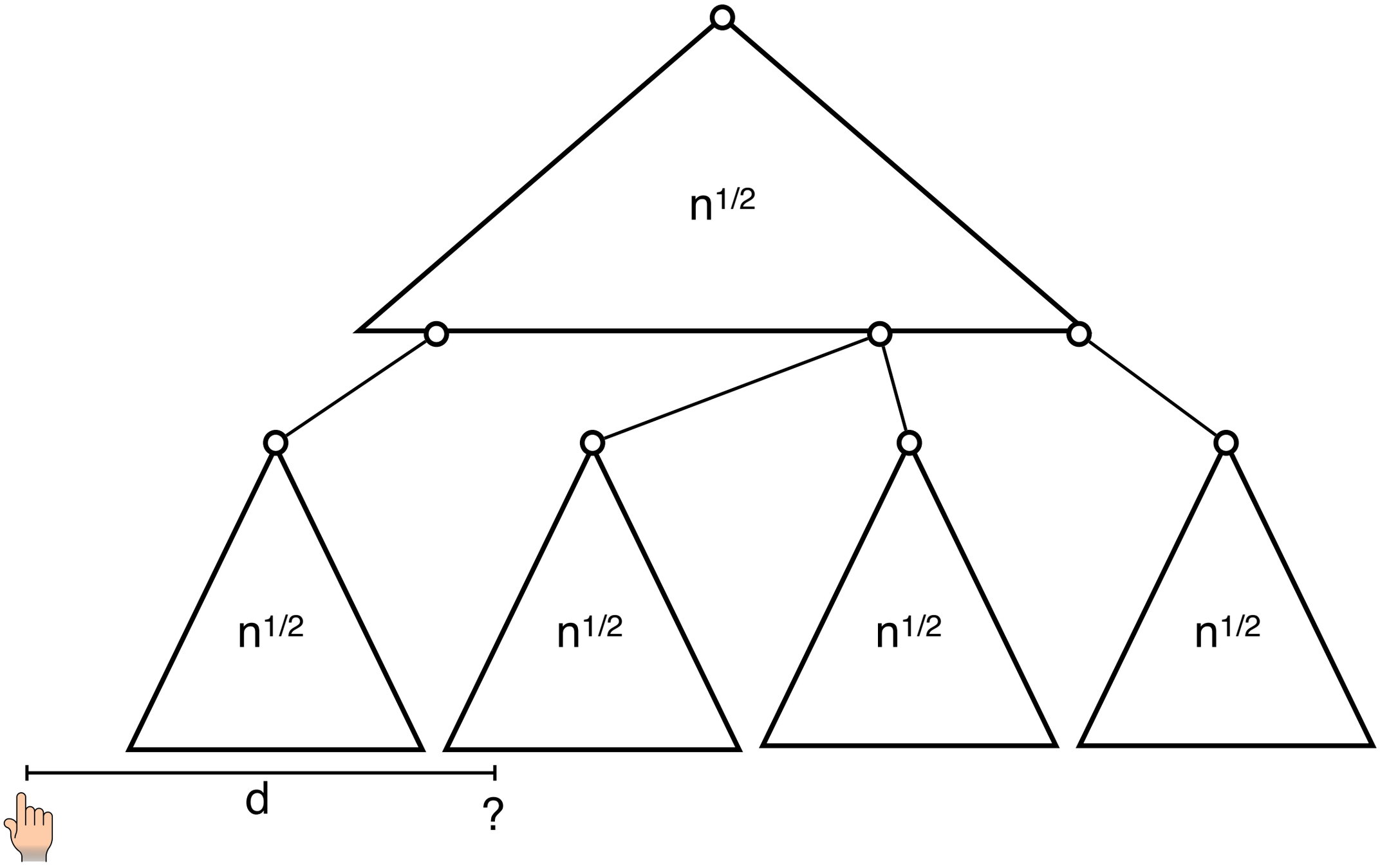
Finger Search



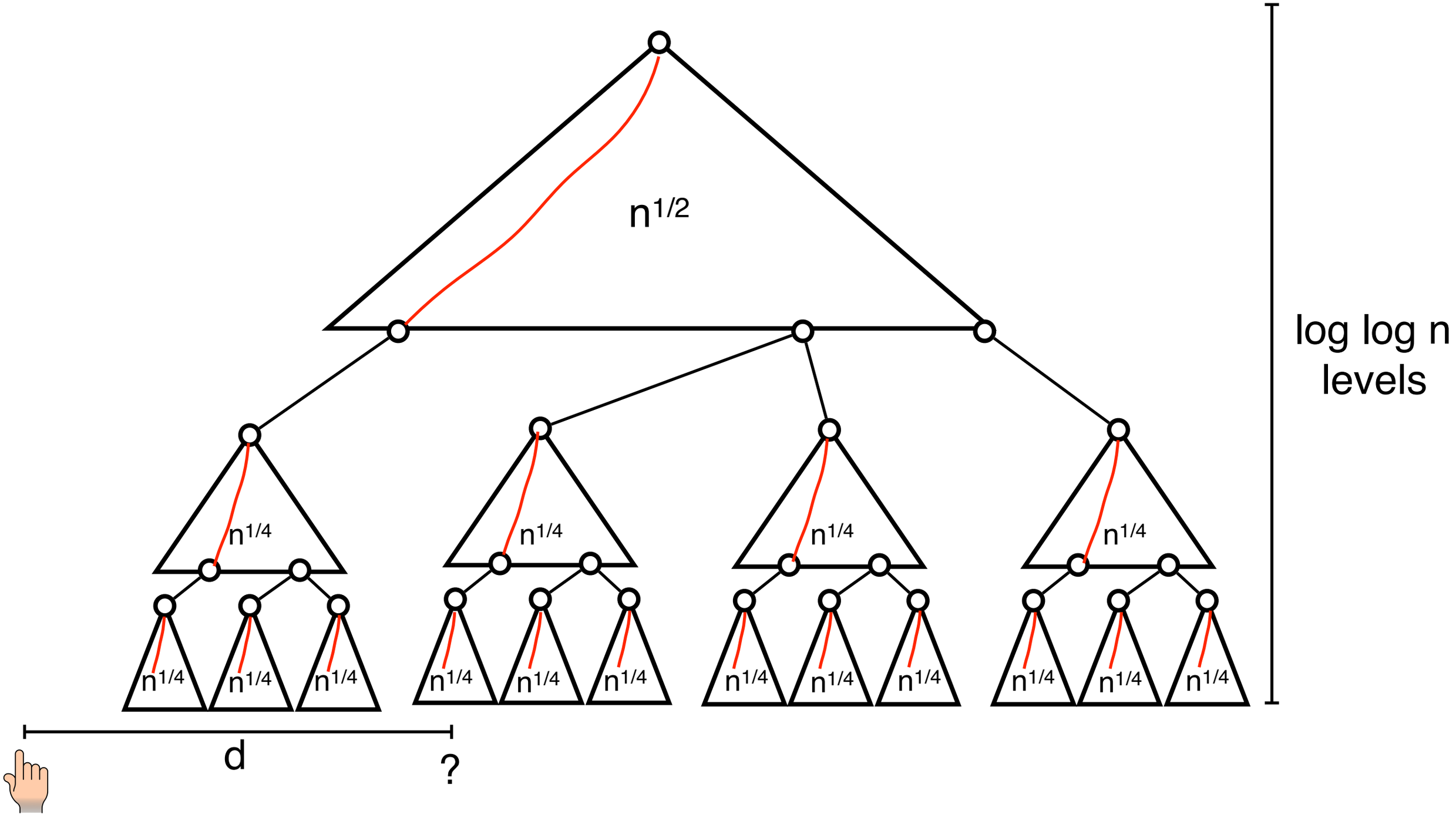
van Emde Boas



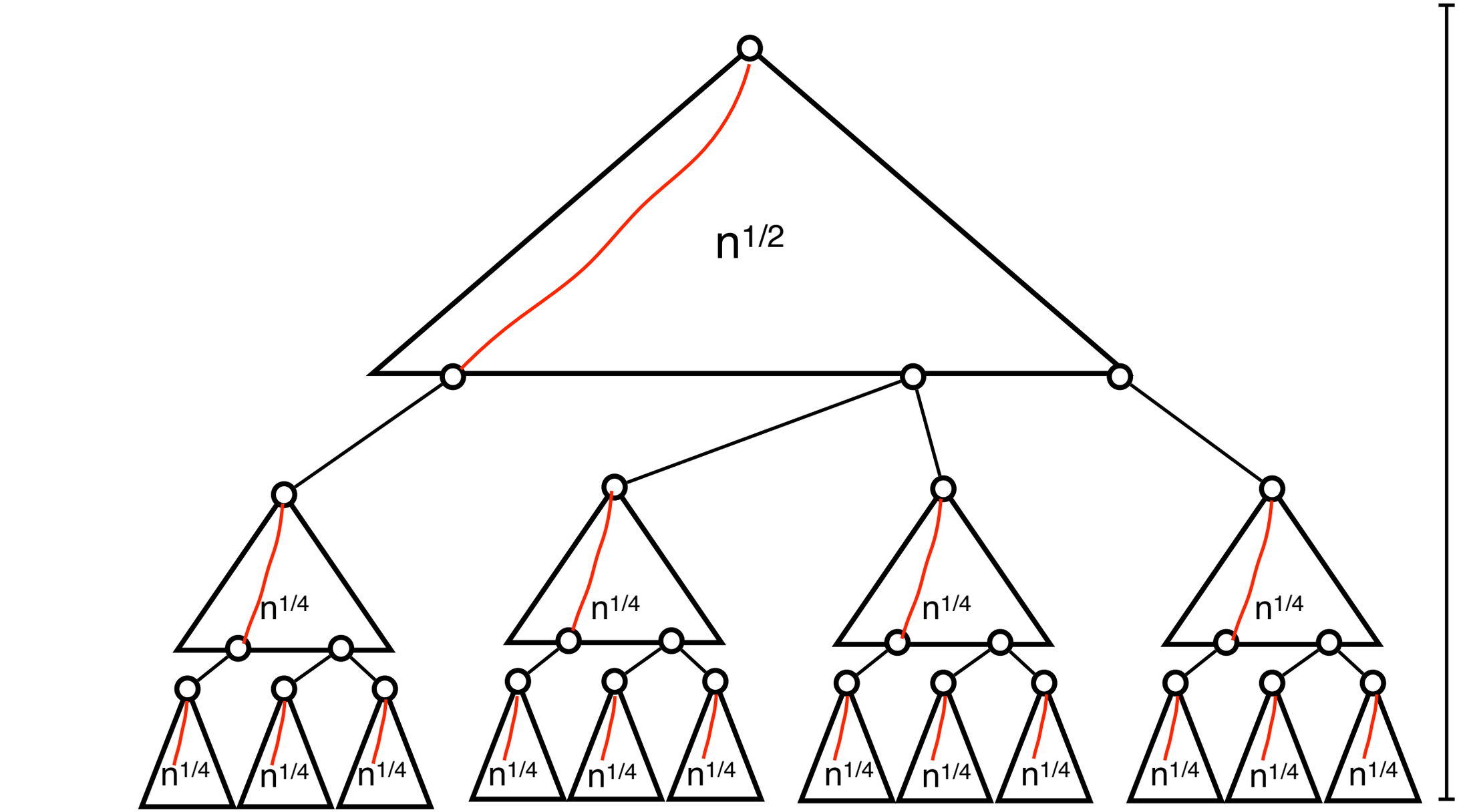
van Emde Boas



van Emde Boas



van Emde Boas



$d \geq n^{1/2} \Longrightarrow O(\log n) = O(\log d)$

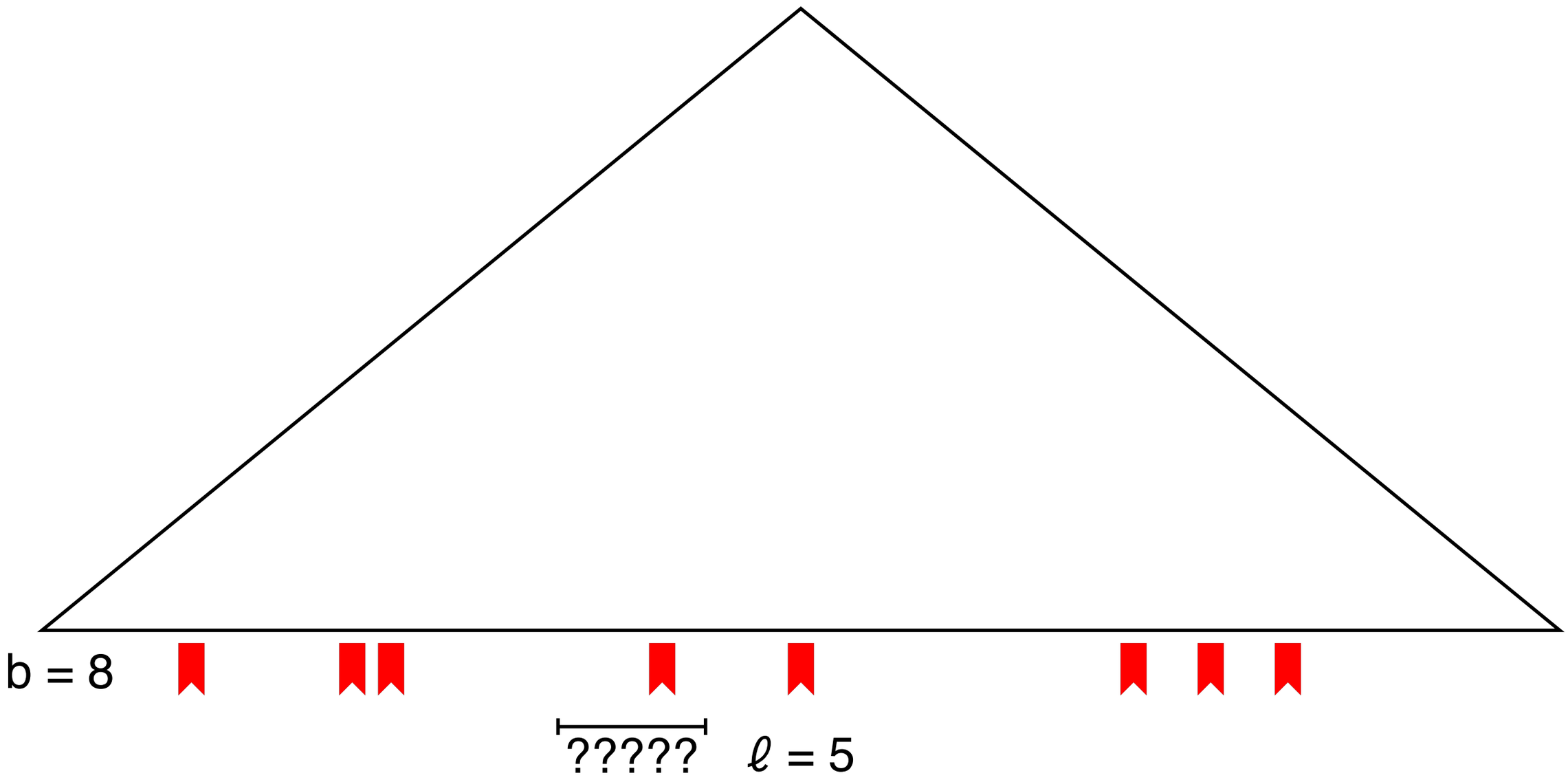
$d < n^{1/2} \Longrightarrow O((\log \log n)^2 + \log d)$

space	setfinger	access	movefinger
O(g)	O(log n)	O(log d)	x
O(g)	O(log n)	O(log d + log log n)	O(log d + log log n)

Bookmarking

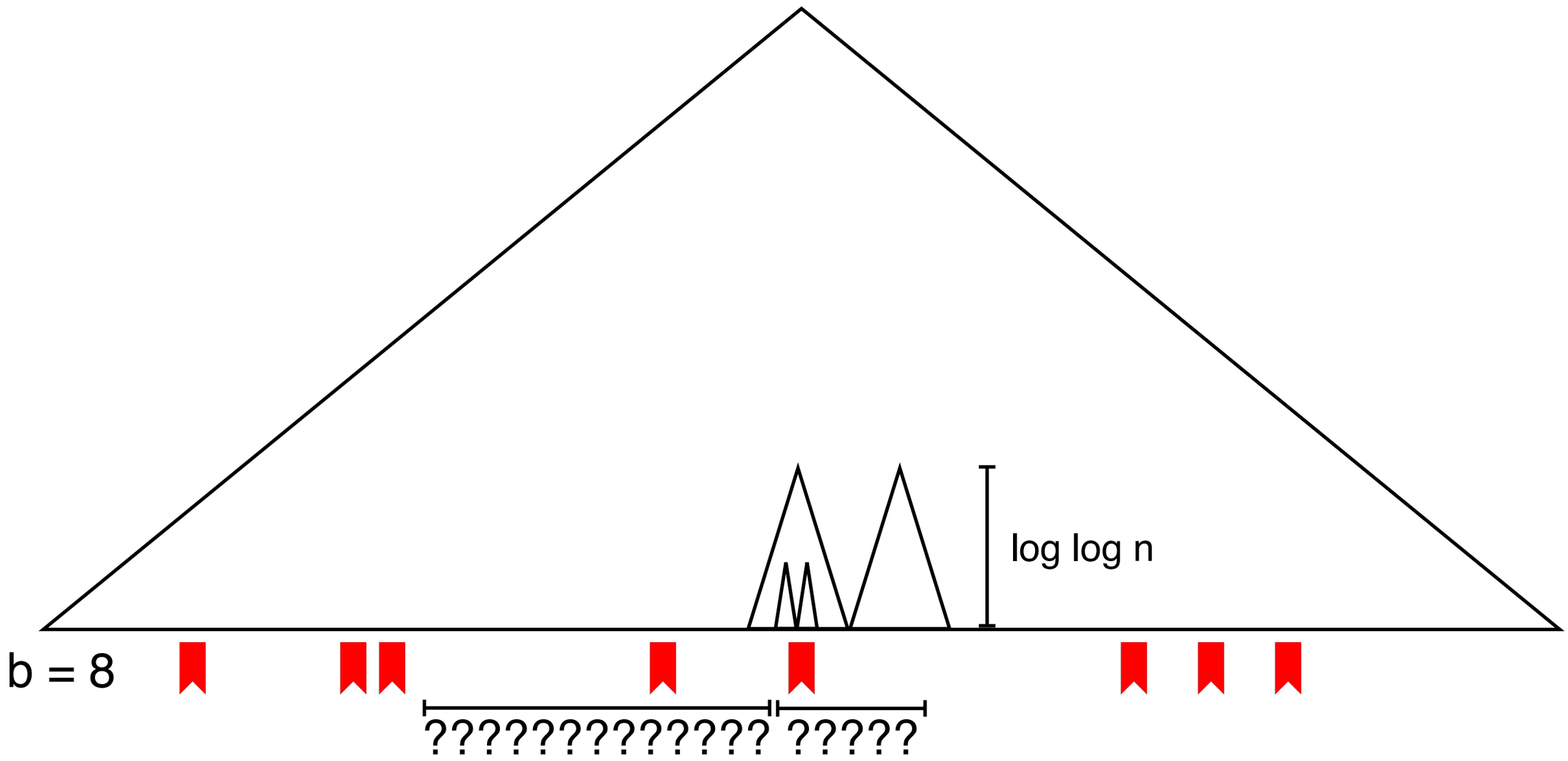
Bookmarking

[Gagie,Gawrychowski,Karkkainen,Nekrich,Puglisi, LATIN 2014]



- b bookmarks at preprocessing time.
- decompress any string of length ℓ crossing a bookmark in $O(\ell)$ time.

Recursive Speedup



$\ell \geq \log n \implies O(\log n + \ell) = O(\ell)$

$\ell < \log n \implies O(\log \log n + \ell)$

$\implies$ $O(\log \log n + \ell)$ time, $O(g \log n + b)$ space

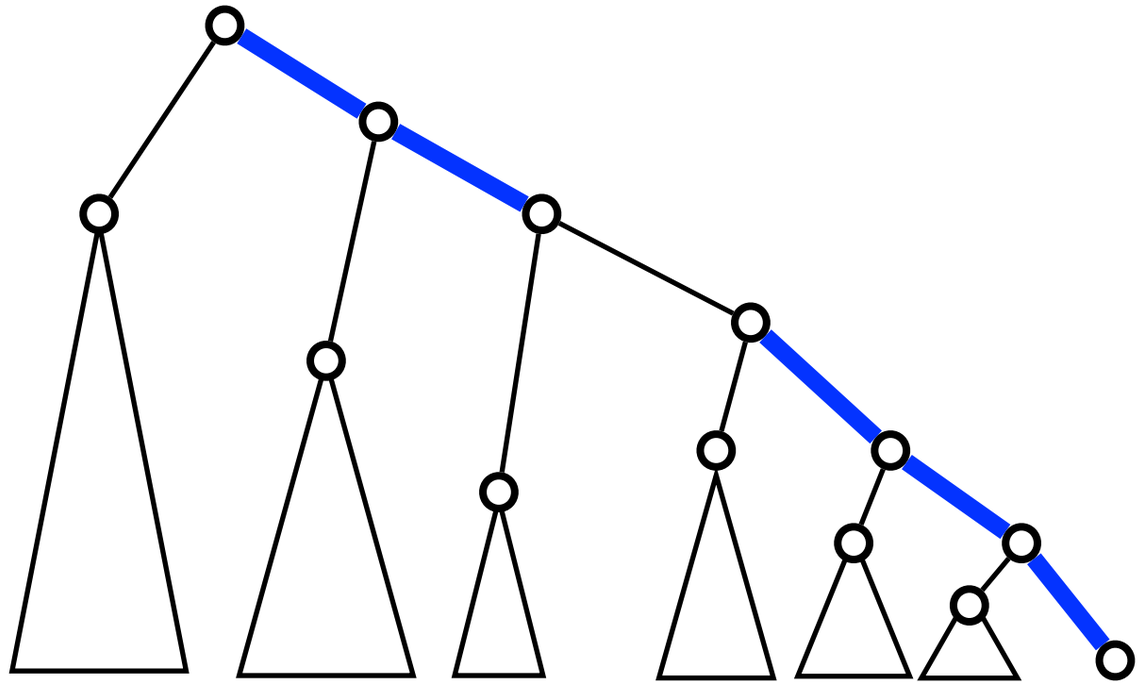
$\implies$ $O(\ell)$ time, $O(g \log n + b \log^* n)$ space

Space	Time	Reference
O(g log n + blog* n)	O(ℓ)	[GGKNP2014]
O(g + blog* n)	O(ℓ)	[CGW2016]

Applications

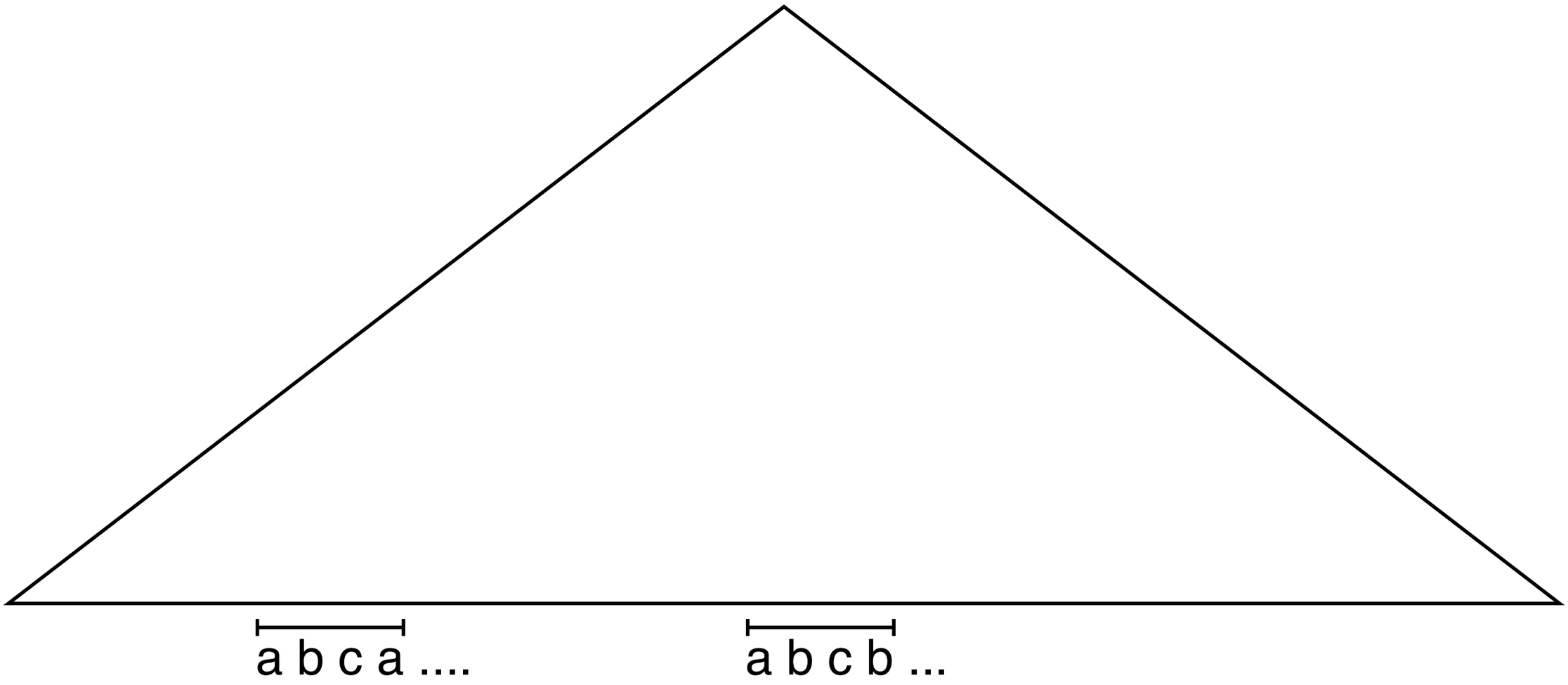
Traversal Framework

[B., Cording, Gørtz, Sach, Vildhøj, Vind JCSS 2017]
[Belazzougui, Landau, Cording, Puglisi, Tabei ESA 2015]



- Traversal on heavy paths for computation.
  - Karp-Rabin fingerprints
  - Rank/select

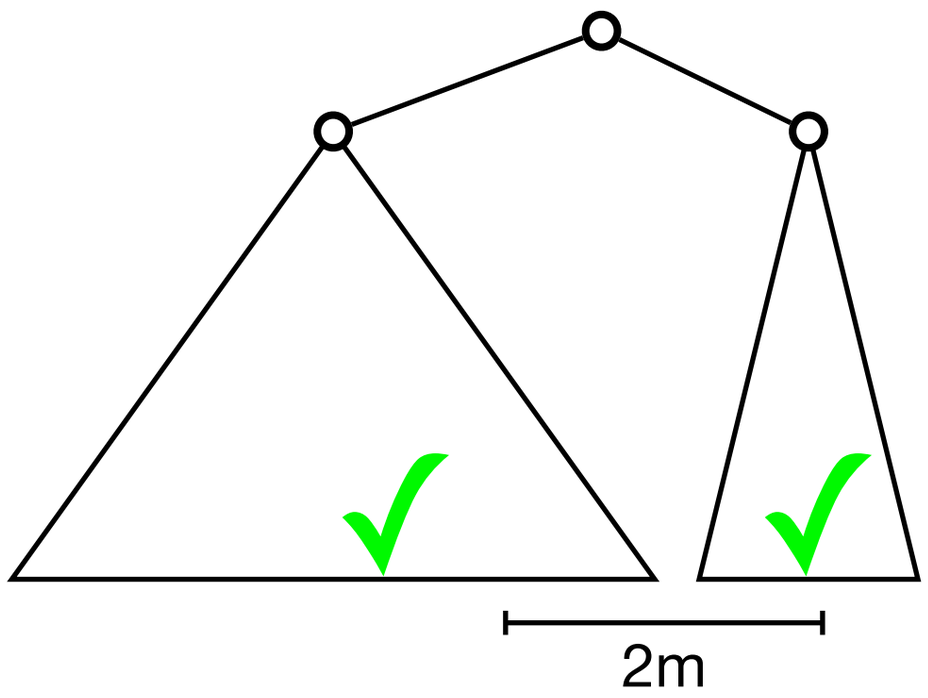
Longest Common Extension



- LCE(i,j): compute longest common extension of S[i,n] and S[j,n]
- ⇒ O(g) space and O(log n + $\log^2 \ell$) time.

Pattern Matching

[B,Landau, Raman, Sadakane, Satti, Weimann, SICOMP 2015]



- Does pattern P of length m appear (perhaps with k errors)?
- O(g (log n + m + Blackbox(m))) time.