Finger Search in Grammar-Compressed Strings

Philip Bille
Anders Roy Christiansen
Patrick Hagge Cording
Inge Li Gørtz

Plan

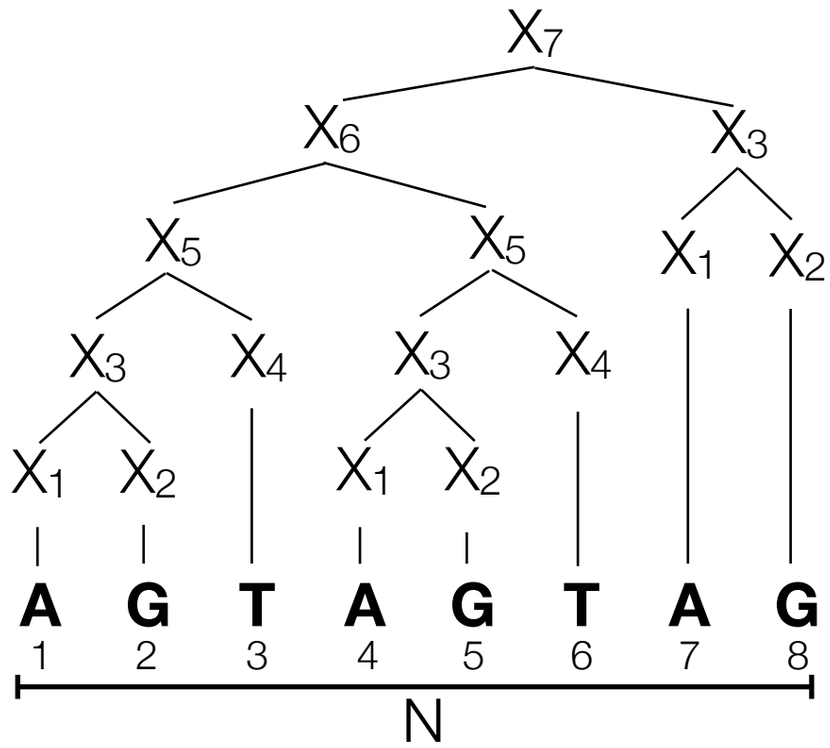
- Grammar compression
- Random access
- Bookmarking
- Finger search
  - Longest common extensions
  - Fringe access
  - Static finger search
  - Dynamic finger search

Grammar Compression

AGTAGTAG N = 8

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

n = 7



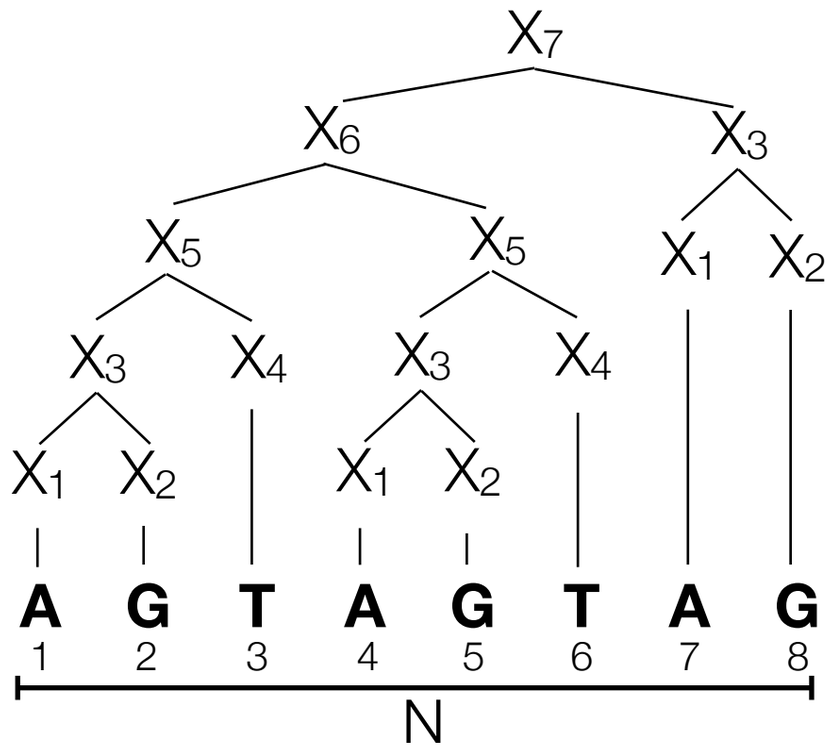
- Grammar compression.
  - Compress string of length N into a straight-line program of size n.
  - Captures many schemes with no or little blowup: Lempel-Ziv family, Sequitur, Run-Length Encoding, Re-Pair, ...,

Random Access

AGTAGTAG N = 8

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

n = 7



- Access(i): what is the ith character in S?
- Decompress(i,j): what is the substring S[i,j]?

Random Access

Space	Time	Reference
O(n)	O(h)	
O(n)	O(log N)	[BLRSSW2011]
$O(n\tau \log_\tau (N/n))$ $O(n \log^\varepsilon N)$	$O(\log_\tau N)$ O(log N/ log log N)	[BCPT2015] $\tau = \log^\varepsilon N$
$O(n\log^{O(1)} N)$	$\Omega(\log^{1-\varepsilon} N)$	[VY2013]

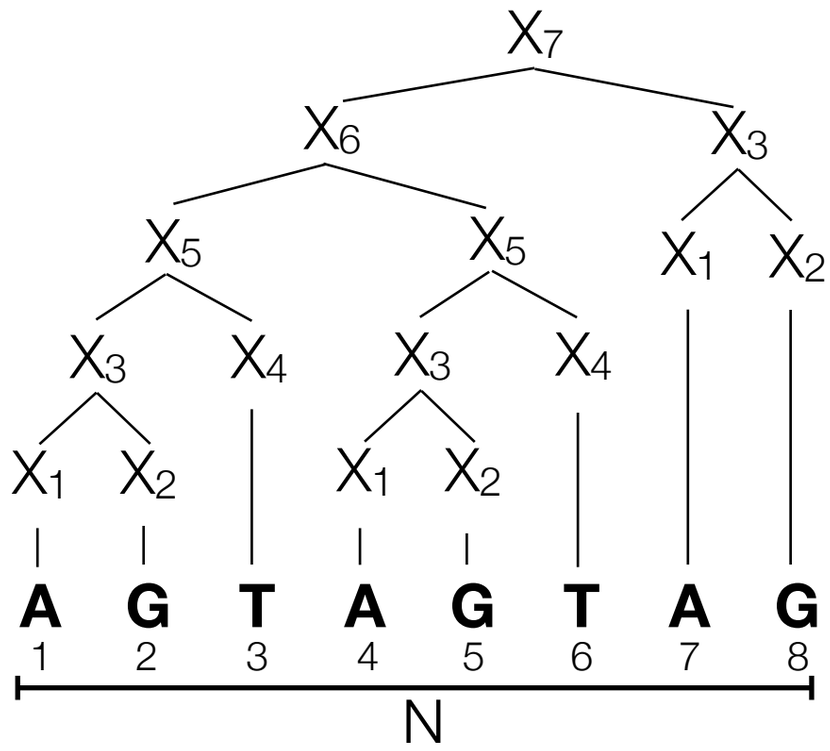
- Decompress in t_{access} + O(D) time.

Bookmarking

AGTAGTAG N = 8

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

n = 7



- b bookmarks at preprocessing time.
- Decompress string of length D from any bookmark in O(D) time.

Bookmarking

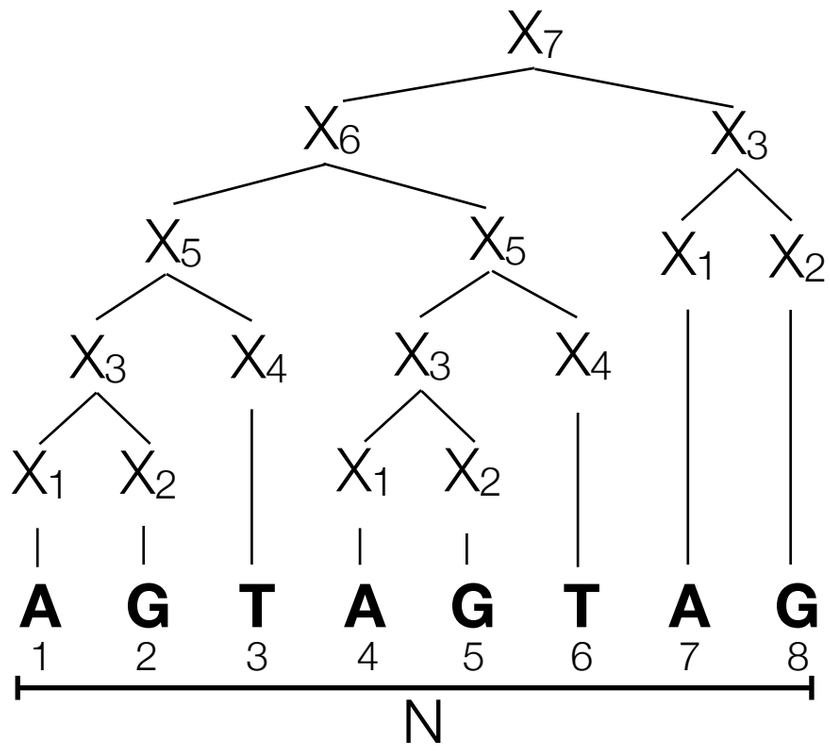
Space	Time	Reference
O(nlog(N/n))	O(D)	[GGKNP2014]
O((n+b) max{1,log*n - log*(n/b + b/n)})	O(D)	[CGW2016]

Finger Search

AGTAGTAG N = 8

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

n = 7

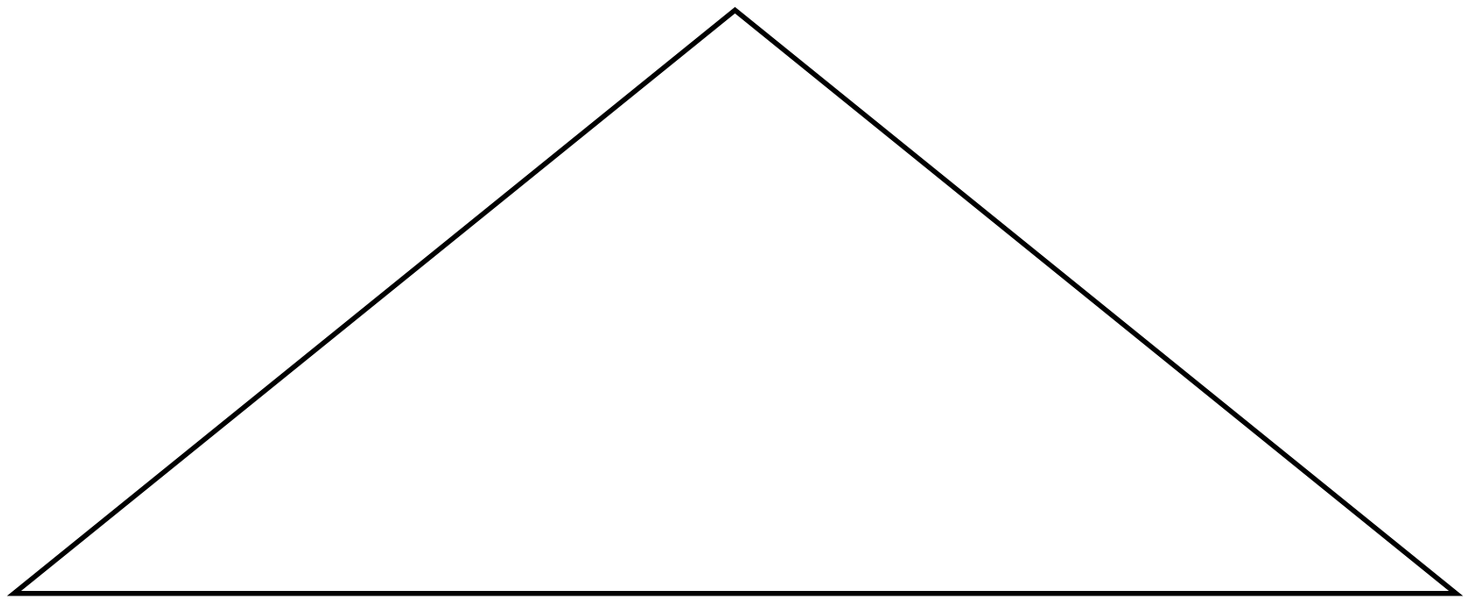


- Setfinger(f): place finger at position f.
- Movefinger(f): move finger to position f.
- Access(i): what is the ith character in S?

Finger Search

Space	setfinger	access	movefinger
O(n)	O(log N)	O(log D)	x
O(n)	O(log N)	O(log D + log log N)	O(log D + log log N)

Longest Common Extension

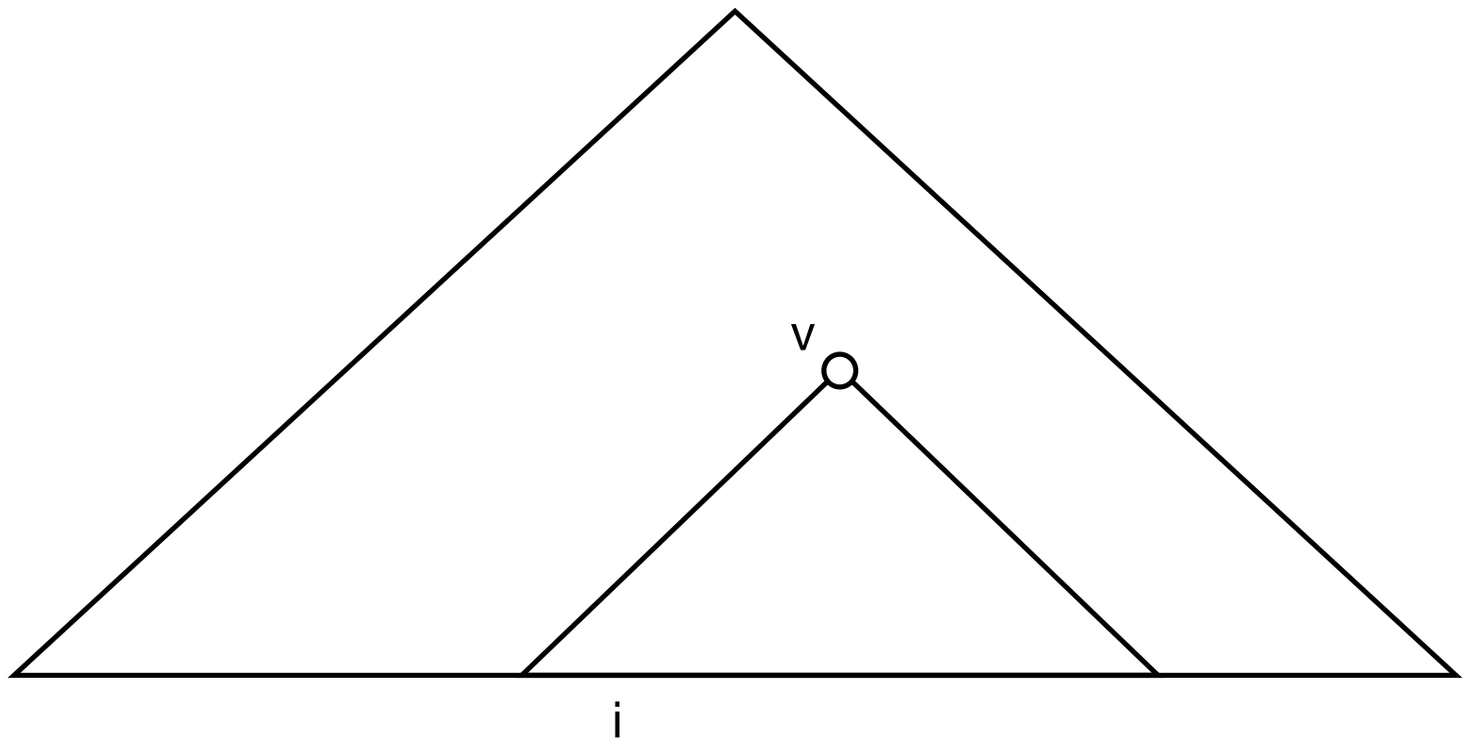


- LCE(i,j): compute longest common extension of S[i,N] and S[j,N]

Longest Common Extension

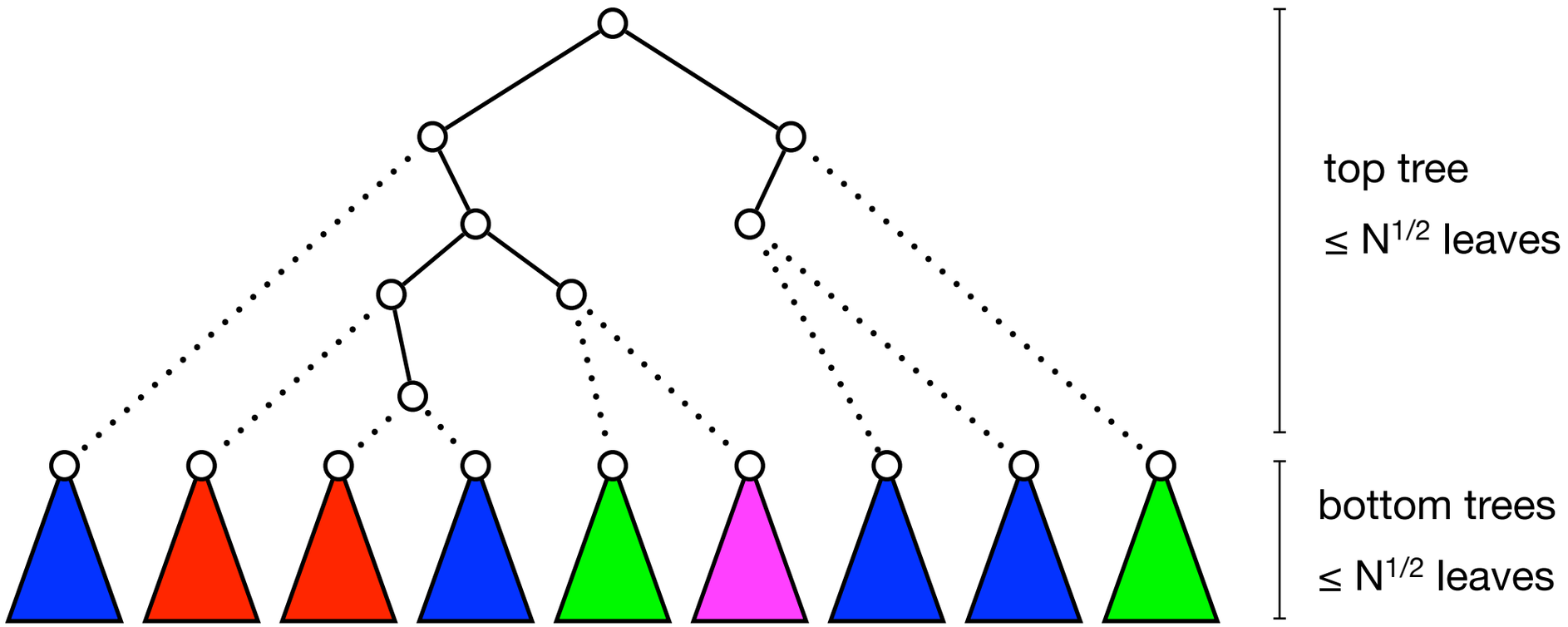
Space	LCE	reference
O(n)	O(log Nlog L)	[BCGSVV2013]
O(n log N log*N)	O(log N + log L log*N)	[NIIBT2016]
O(n)	O(log N + $\log^2$ L)	This paper

Fringe Access



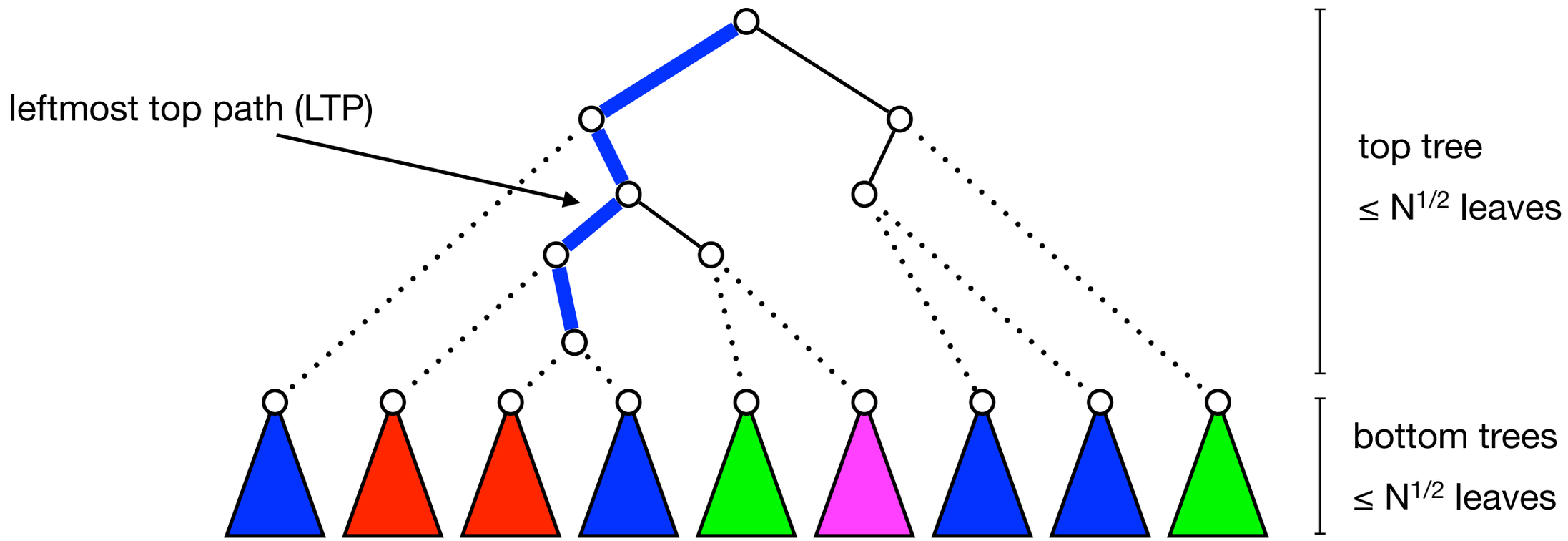
- What is the ith character in S(v)?
- Fast when i is close to left or right end of S(v).
- Goal.
  - O(log i + log log N) time and O(n) space.
  - Consider only left fringe.

van Emde Boas



- Recursive ART [AHR1998] decomposition.
- O(log log N) levels.

Fringe Access

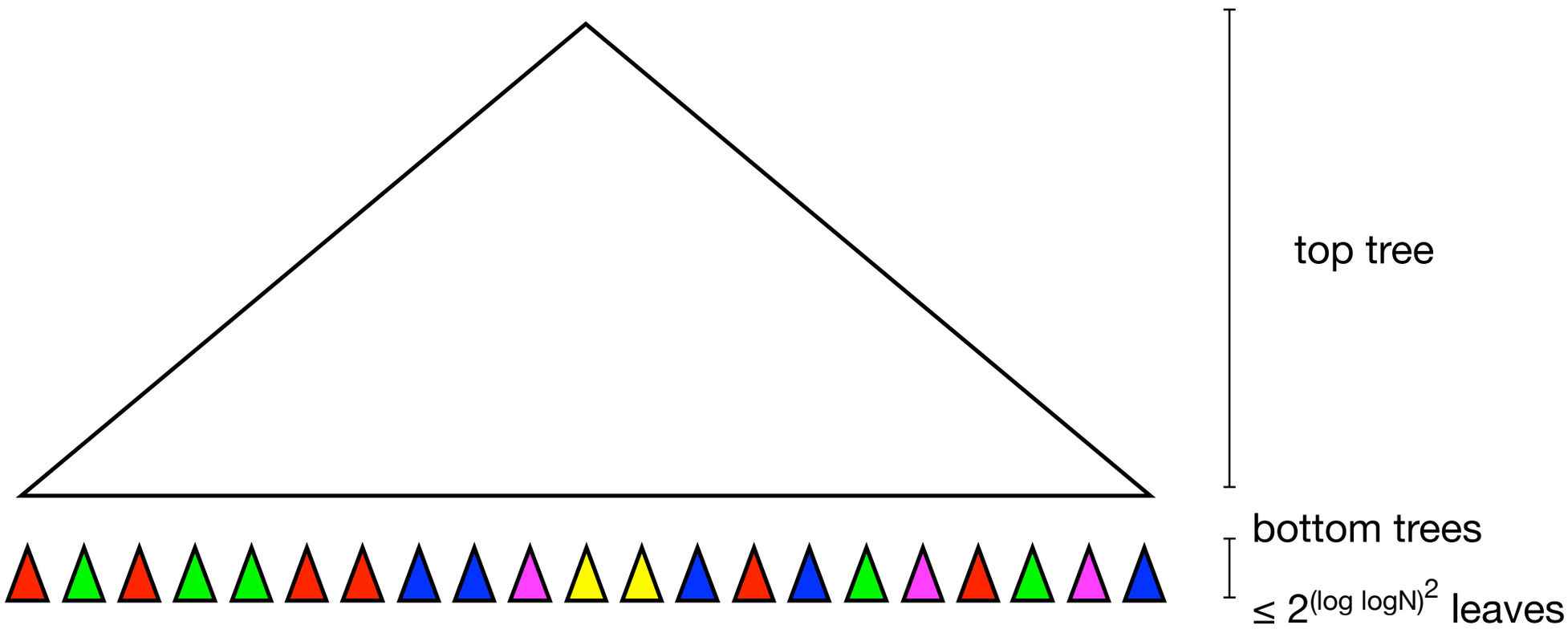


- Data structure.
  - Predecessor on tree sizes to the left of LTP — O(n) space via weighted ancestor queries.
  - Random access data structure.
- Access(i).
  - Case 1. O(1) size tree: decompress. — O(1)
  - Case 2. Left or below LTP: predecessor query + recurse. — $O((\log \log N)^2)$
  - Case 3. Right of LTP: random access. — $i > N^{1/2} \Longrightarrow O(\log N) = O(\log i)$
- Lemma. Fringe access in O(n) space and $O(\log i + (\log \log N)^2)$ time.

Fringe Access

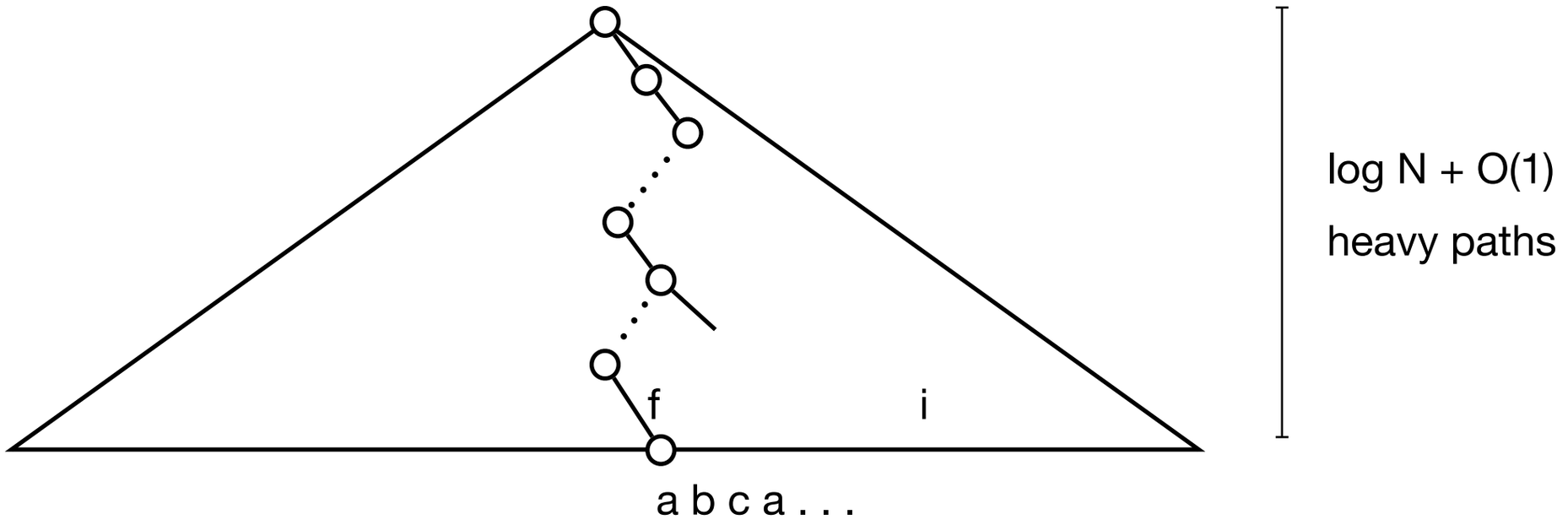
- How to speed up queries?
- Case 1. $i \geq 2^{(\log \log N)^2}$: use previous solution. $O(\log i + (\log \log N)^2) = O(\log i)$ time.
- Case 2. $i < 2^{(\log \log N)^2}$: new data structure.

Fringe Access



- Data structure for $i < 2^{(\log \log N)^2}$.
  - Special decomposition at level 1 in recursion.
  - $O(1 + \log \log 2^{(\log \log N)^2}) = O(\log \log \log N)$ levels of recursion.
- Access(i).
  - Level 1: $O(\log \log N)$.
  - Levels ≥ 2: $O(\log \log 2^{(\log \log N)^2}) = O(\log \log \log N)$ (with new WA trick)
  - $\Longrightarrow O(\log i + \log \log N + (\log \log \log N)^2) = O(\log \log N + \log i)$ time.
- Lemma. Fringe access in O(n) space and $O(\log i + \log \log N)$ time.

Static Finger Search



- Heavy path decomposition.
- Setfinger(f).
  - Find heavy path to f. O(log N)
  - Decompress string S[f, f + log N] O(log N)
- Access(i).
  - Case 1. D ≤ log N: Return char. O(1)
  - Case 1. D > log N: Search heavy paths + fringe search. O(log log N + log D) = O(log D)
- Theorem. Finger search in O(n) space, O(log N) setfinger, and O(log D) access.

Dynamic Finger Search

- Theorem. Finger search in O(n) space, O(log N) setfinger, and O(log D + log log N) movefinger and access.