Random Access to Grammar Compressed Strings

Philip Bille, Gad M. Landau, Rajeev Raman, Kunihiko Sadakane, Srinivasa Rao Satti, and Oren Weimann

Random Access to Compressed Strings

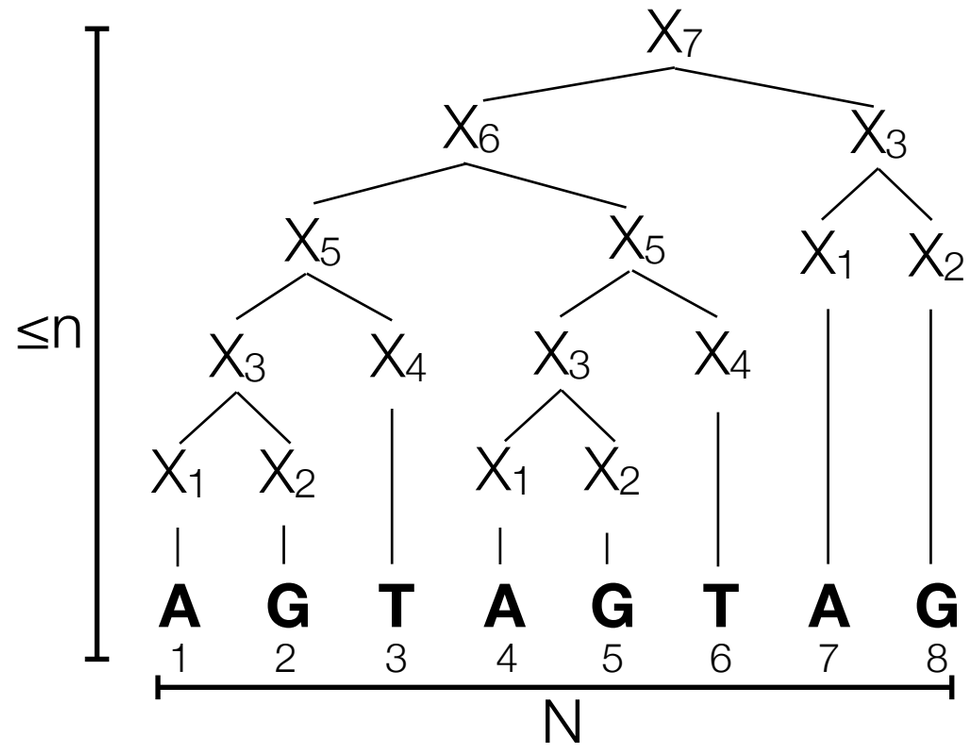
Random Access to Compressed Strings

- What is the ith character?
- What is the substring at [i,j]?
- Does pattern P appear in text? (perhaps with k errors?)

Random Access to Grammar Compressed Strings

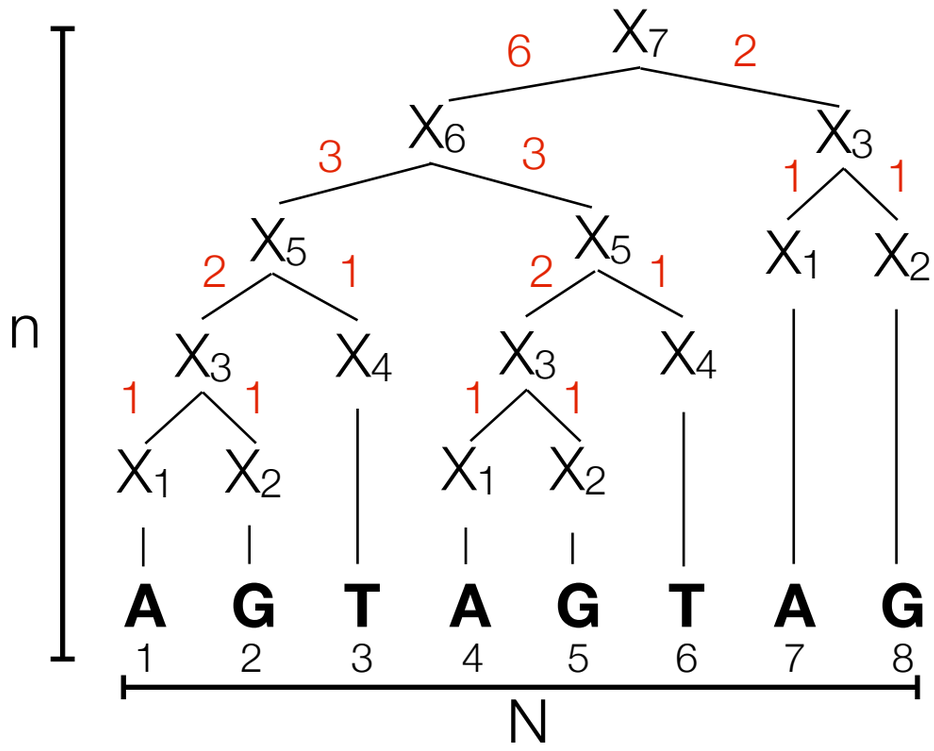
AGTAGTAG N = 8

$X_7 \rightarrow X_6X_3$
$X_6 \rightarrow X_5X_5$
$X_5 \rightarrow X_3X_4$
$X_4 \rightarrow \mathbf{T}$ n = 7
$X_3 \rightarrow X_1X_2$
$X_2 \rightarrow \mathbf{G}$
$X_1 \rightarrow \mathbf{A}$



- Grammar based compression captures many popular compression schemes with no or little blowup in space [Charikar et al. 2002, Rytter 2003].
- Lempel-Ziv family, Sequitur, Run-Length Encoding, Re-Pair, ...

Tradeoffs and Results



- What is the ith character?

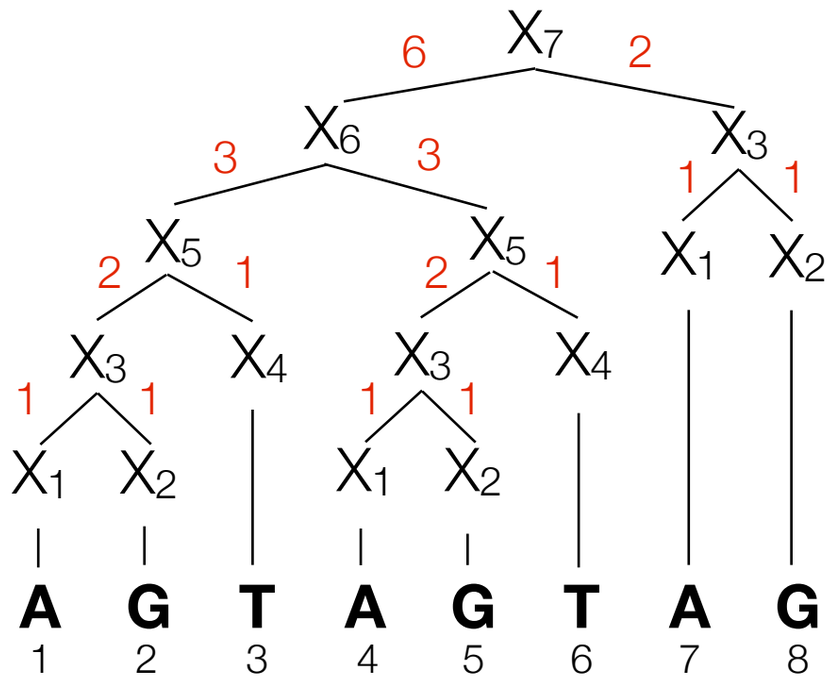
O(N) space	O(n) space	O(n) space
O(1) query	O(n) query	O(log N) query

- What is the substring at [i,j]?

O(n) space

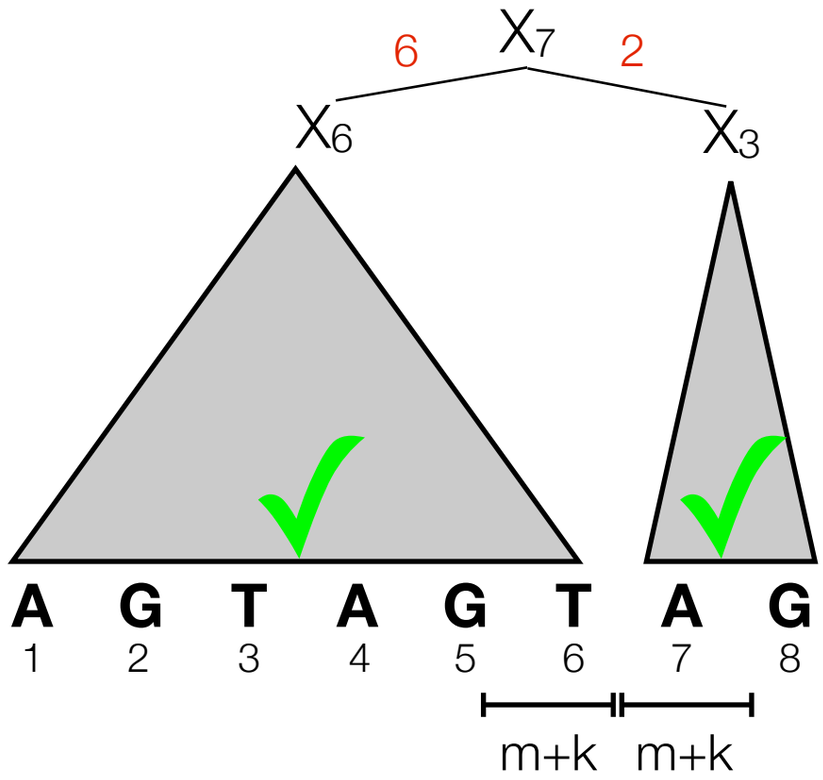
O(log N + j - i) query

Application: Black-Box Compressed String Matching



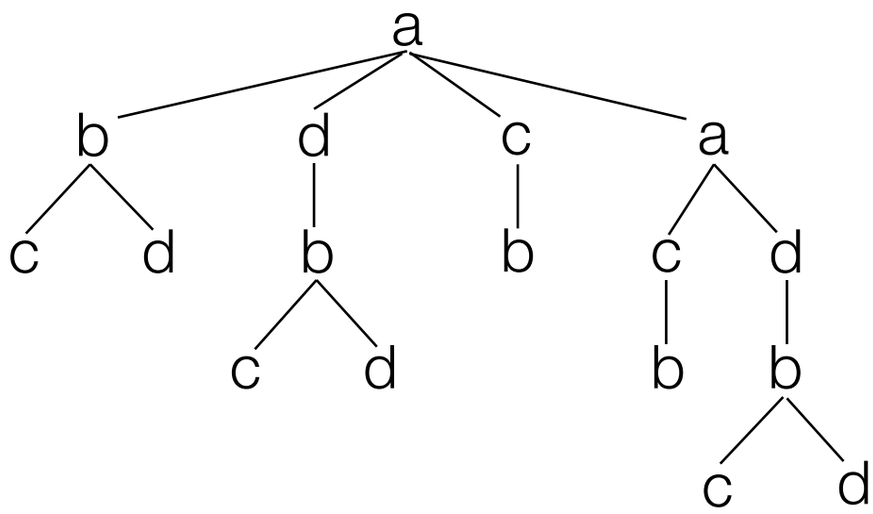
- Does “**AGGA**” appear in the text (perhaps with k errors)?

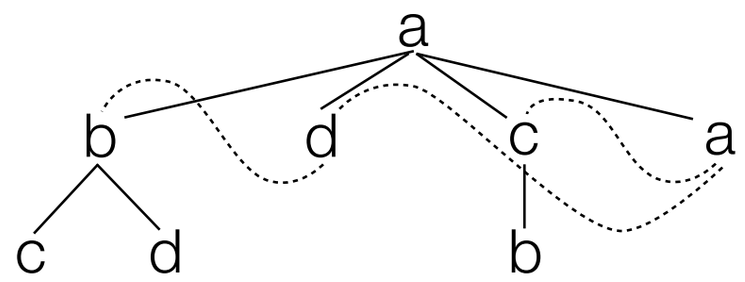
Application: Black-Box Compressed String Matching



- Does “**AGGA**” appear in the text (perhaps with k errors)?
- Total time O(n (log N + m + Blackbox(m))).

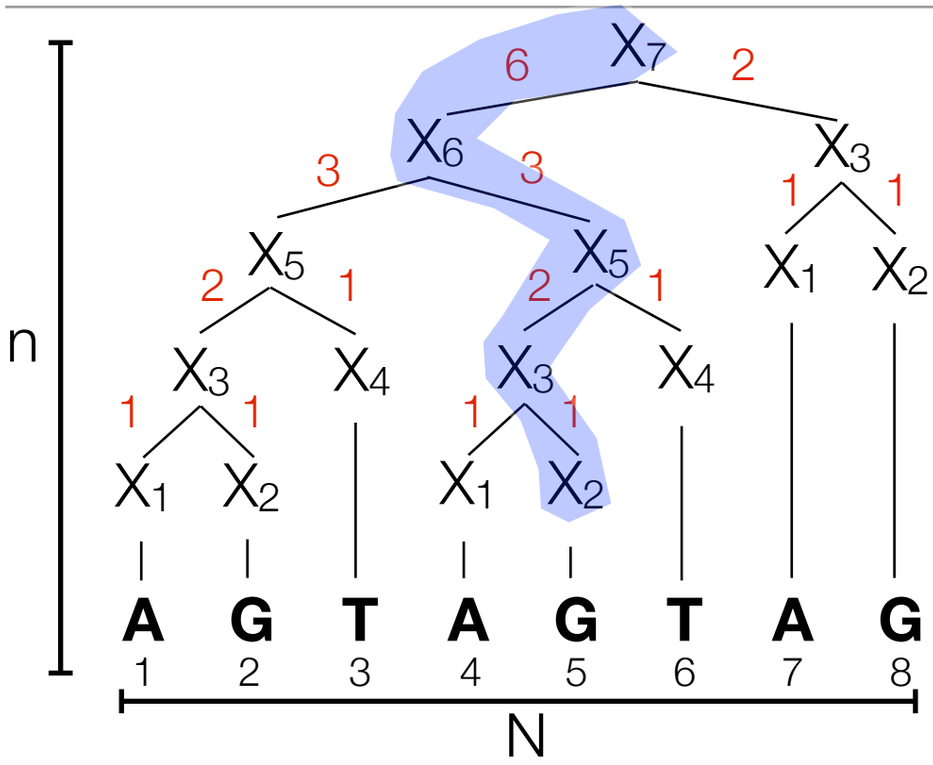
Extension: Compressed Trees



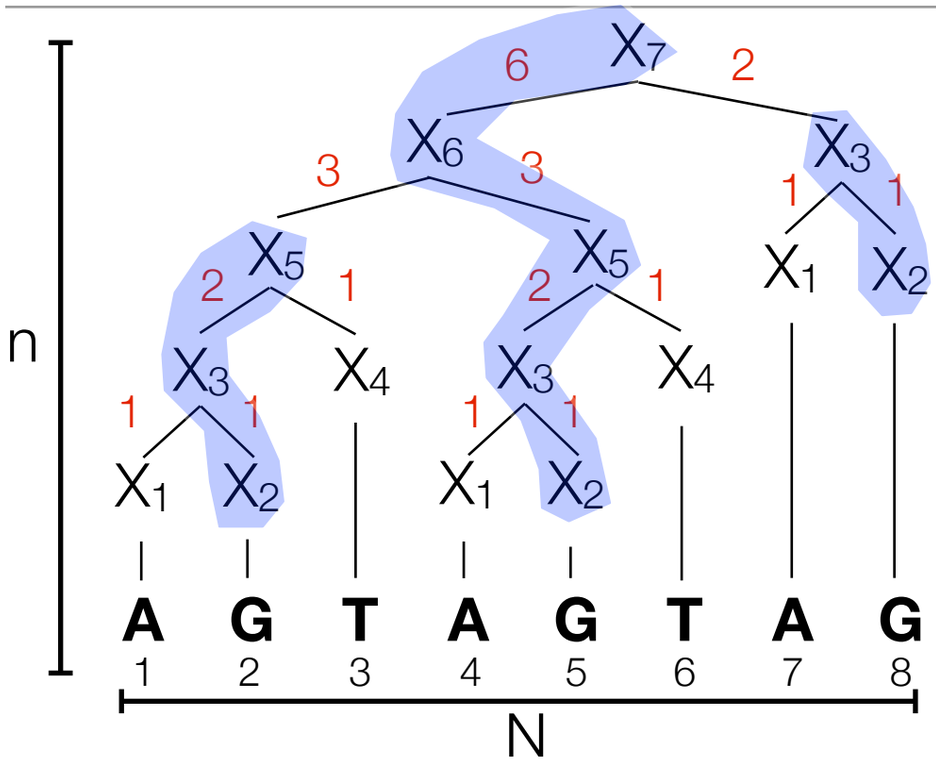


- Linear space in compressed tree.
- Fast navigation operations (select, access, parent, depth, height, subtree_size, first_child, next_sibling, level_ancestor, nca).

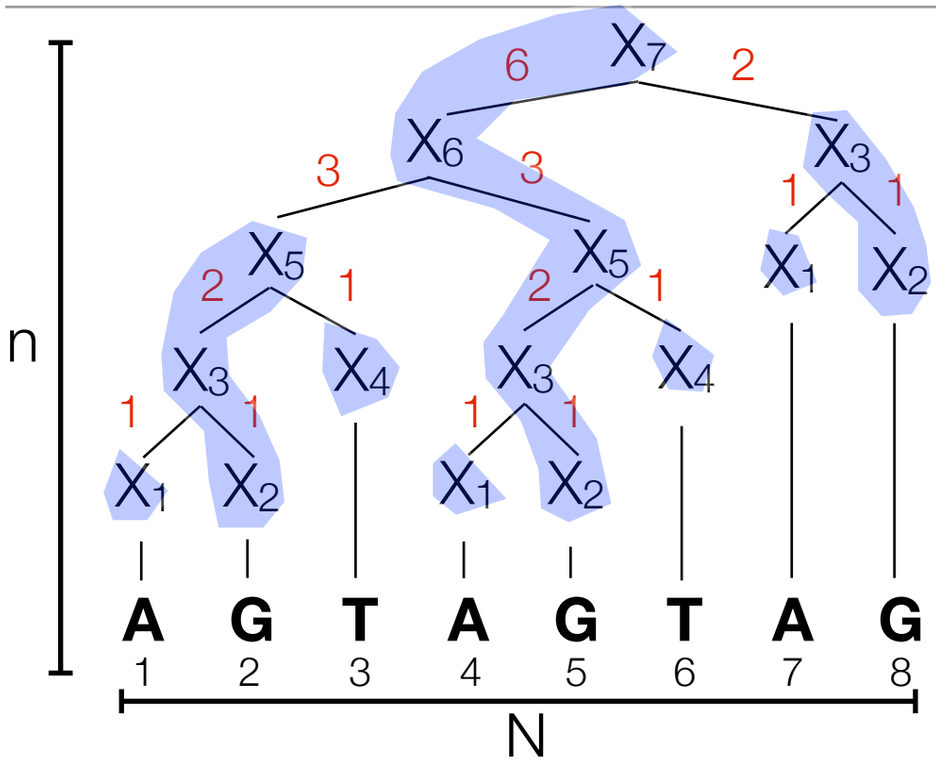
Heavy Path Decomposition



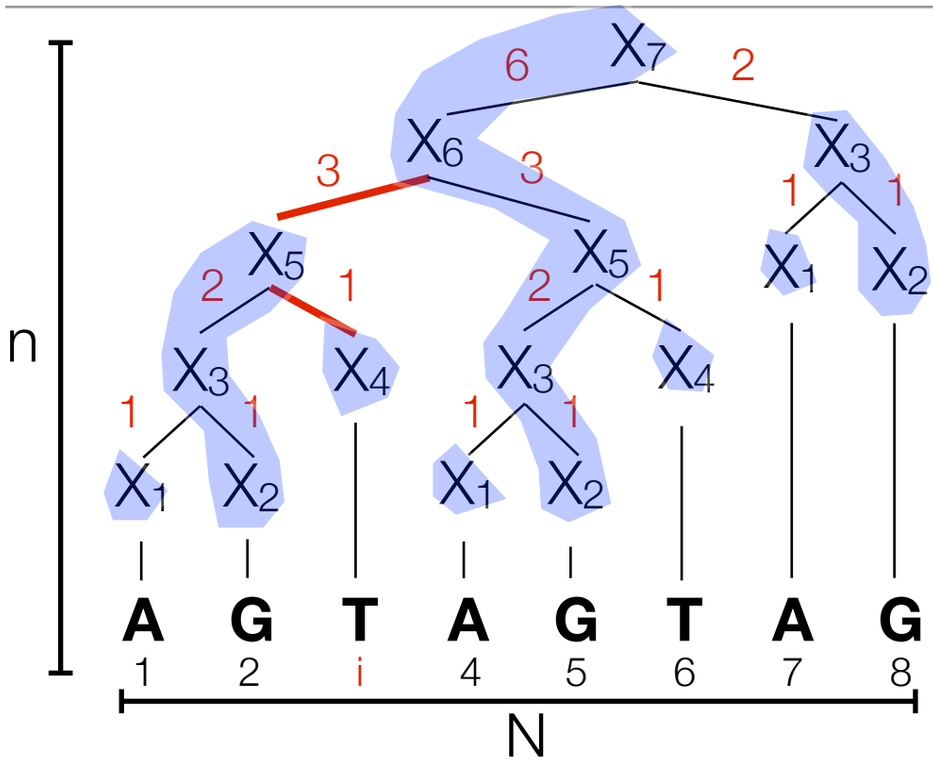
Heavy Path Decomposition



Heavy Path Decomposition



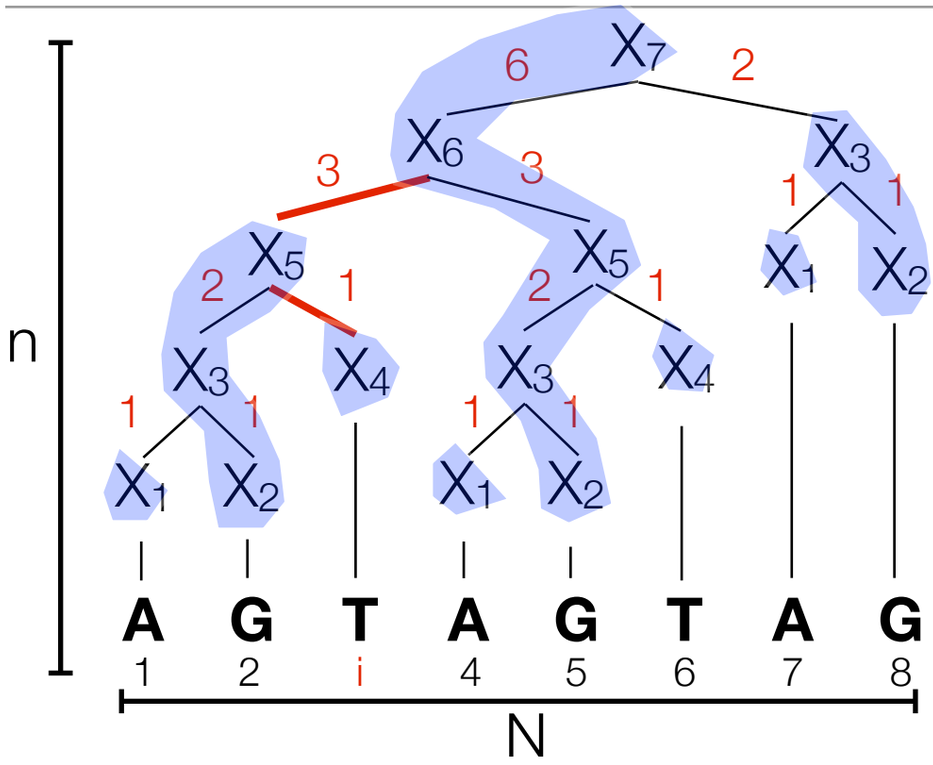
Random Access Query



- The path from root to i goes through O(log N) heavy paths
- Query: Binary search all heavy paths on the way

$$O(\log n) \cdot O(\log N)$$

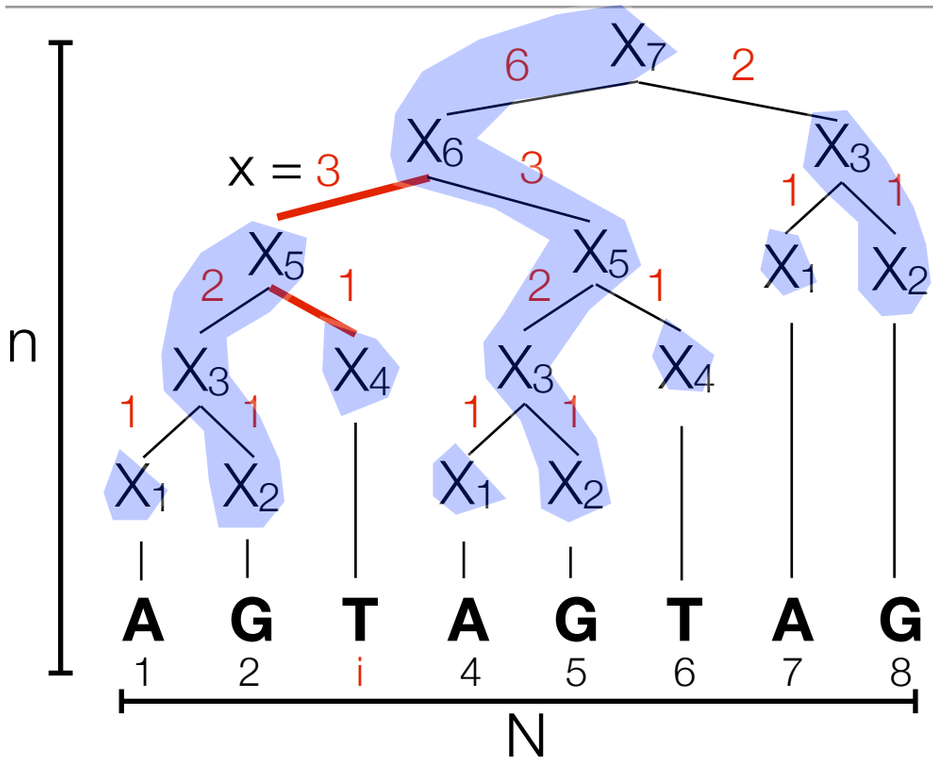
Random Access Query

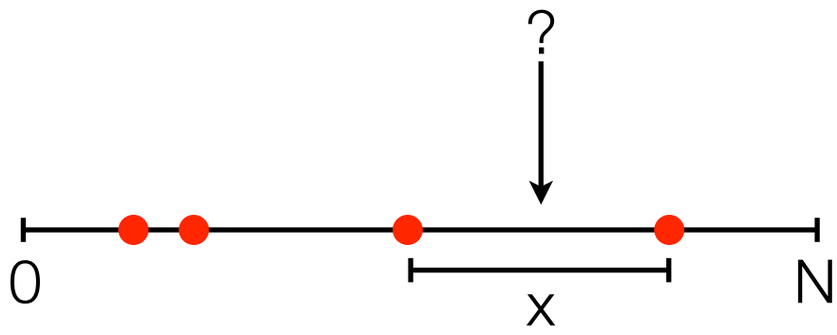


- The path from root to i goes through O(log N) heavy paths
- Query: Binary search all heavy paths on the way

~~O(log n)~~ · O(log N)

Interval Biased Search Tree





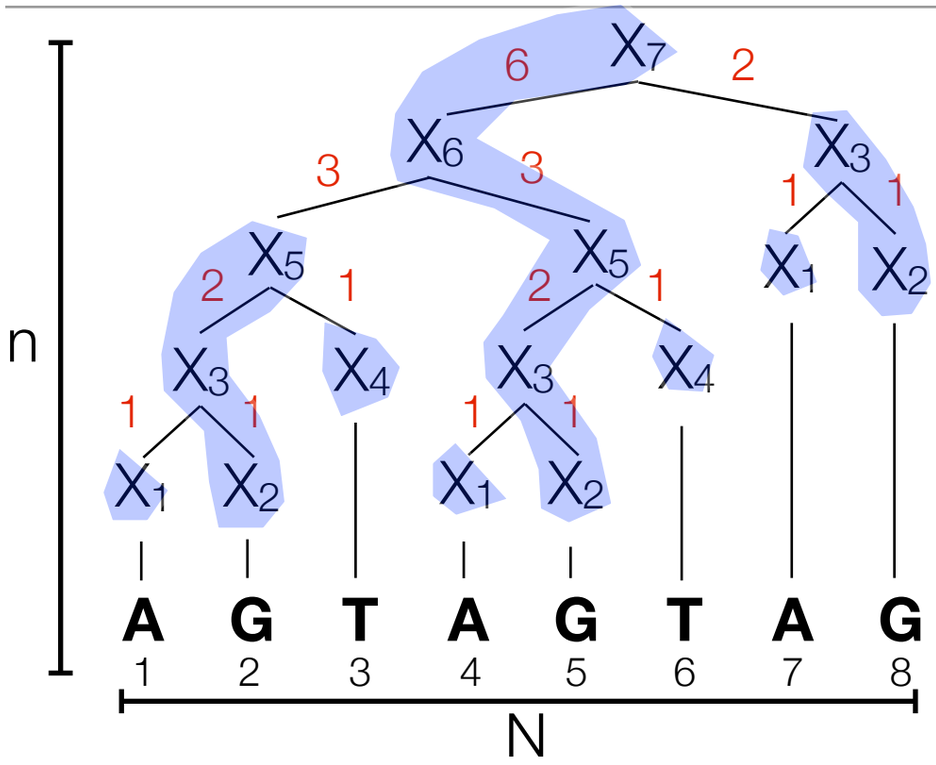
- The path from root to i goes through O(log N) heavy paths
- Query: Binary search all heavy paths on the way

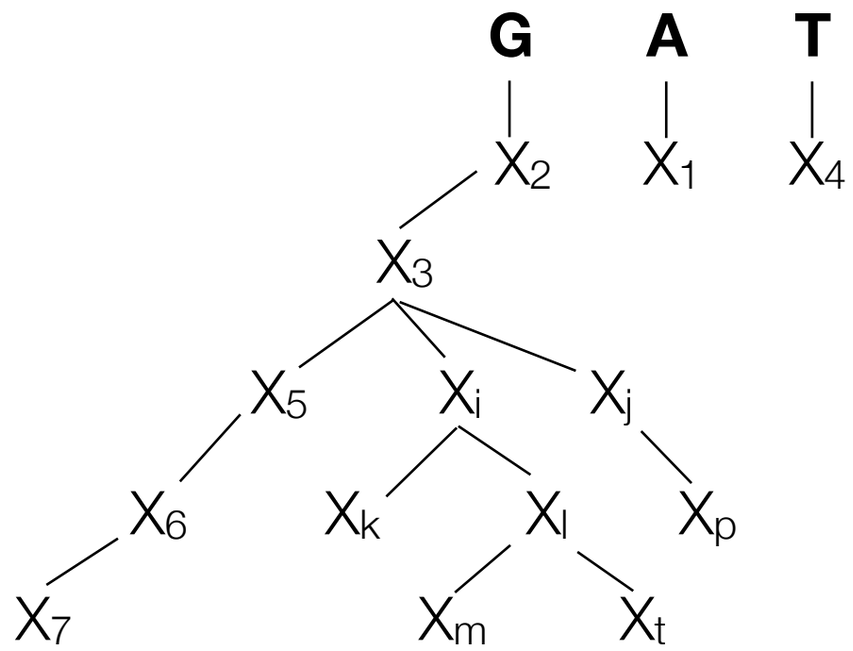
  ~~O(log n)~~ · O(log N)

  O(log N/x) Telescopes to O(log N)

- Space: Each IBSTs uses linear space => total $O(n^2)$ space for all heavy paths.

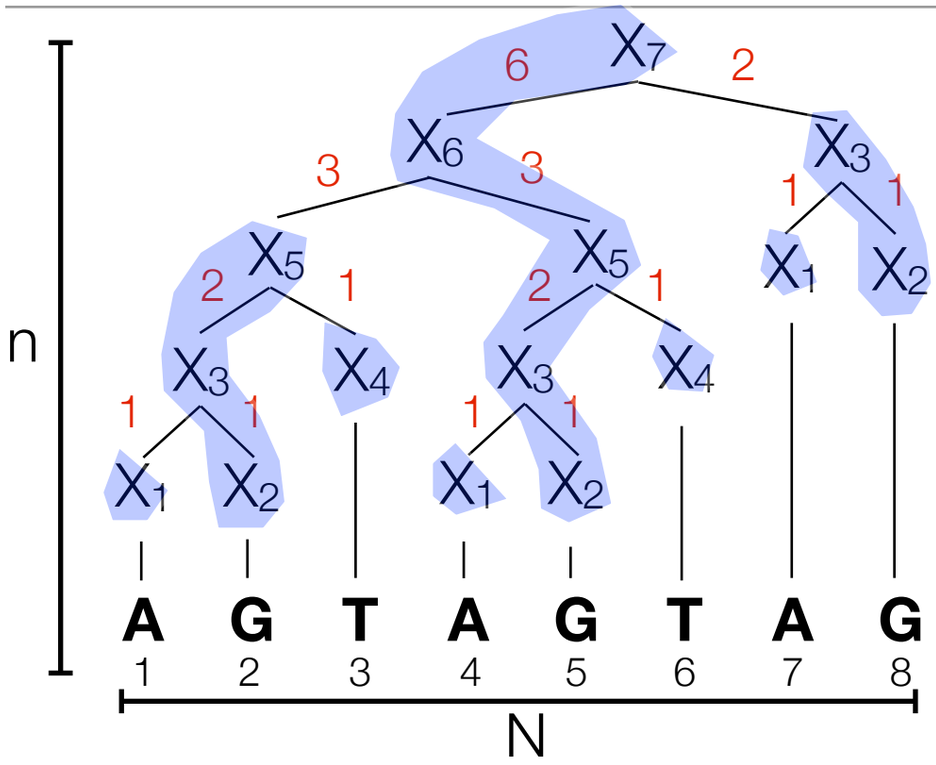
O(n) Representation of Heavy Paths

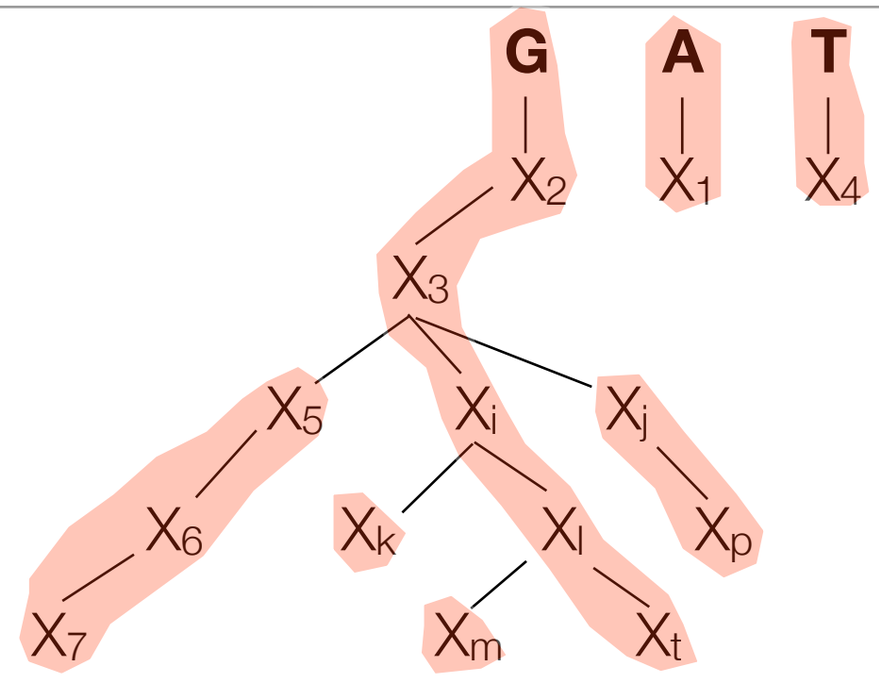




- Search for i on heavy path = lowest ancestor of distance i.

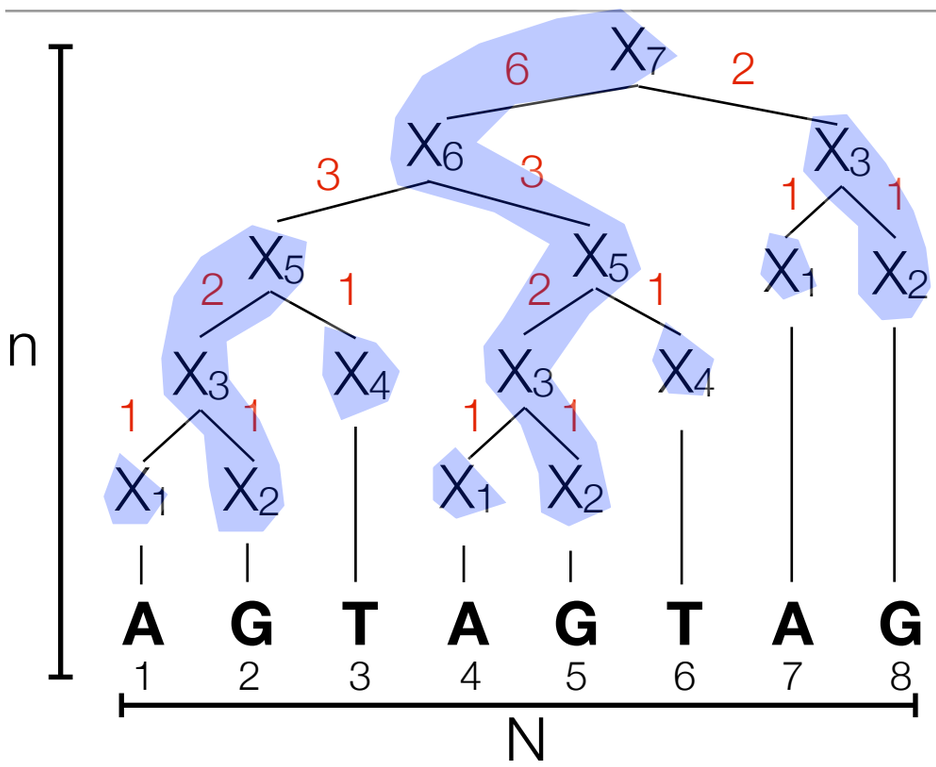
O(n) Representation of Heavy Paths

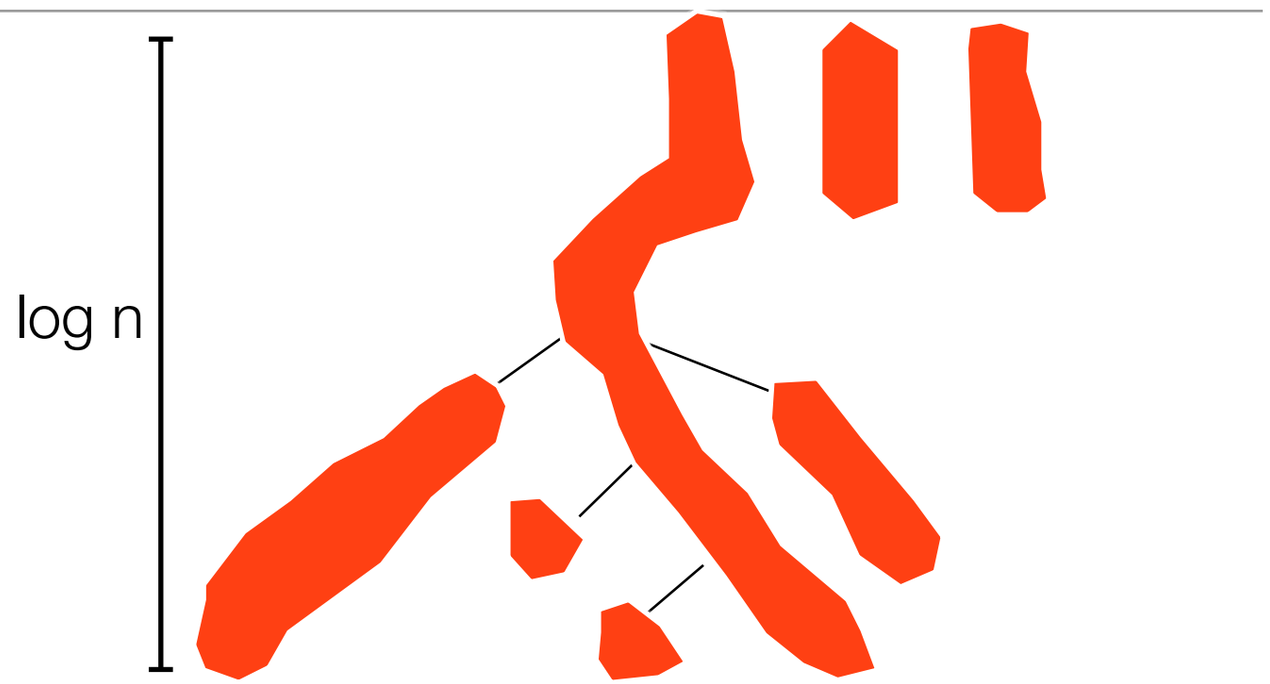




- Search for i on heavy path = lowest ancestor of distance i.
- A heavy path decomposition of heavy path representation.
- In-path: O(log N/x) time, total O(n) space.

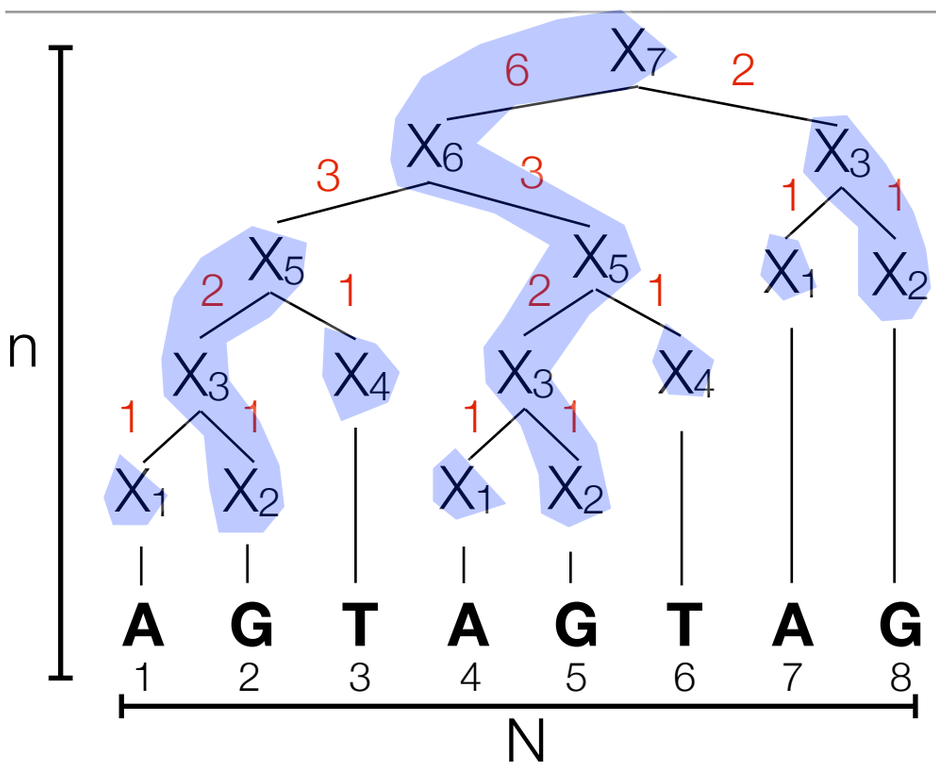
O(n) Representation of Heavy Paths

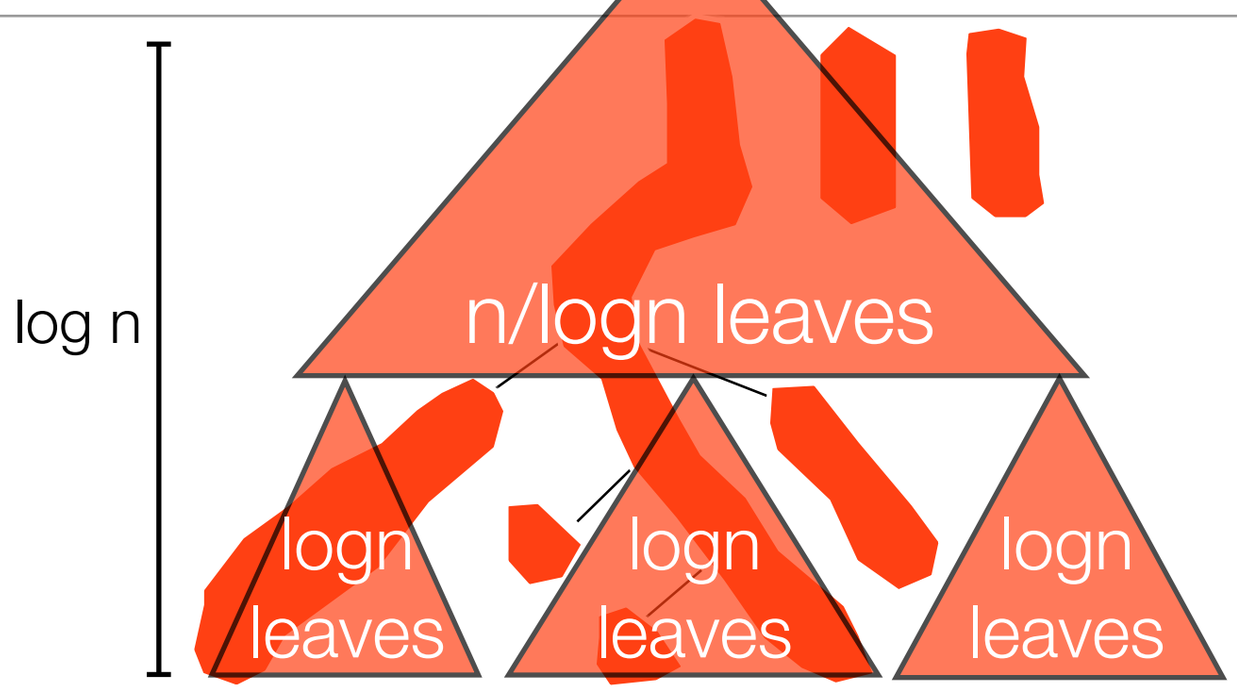




- Search for i on heavy path = lowest ancestor of distance i.
- A heavy path decomposition of heavy path representation.
- In-path: O(log N/x) time, total O(n) space.
- Between-paths: O(log N/x) time, total O(n log n) space.

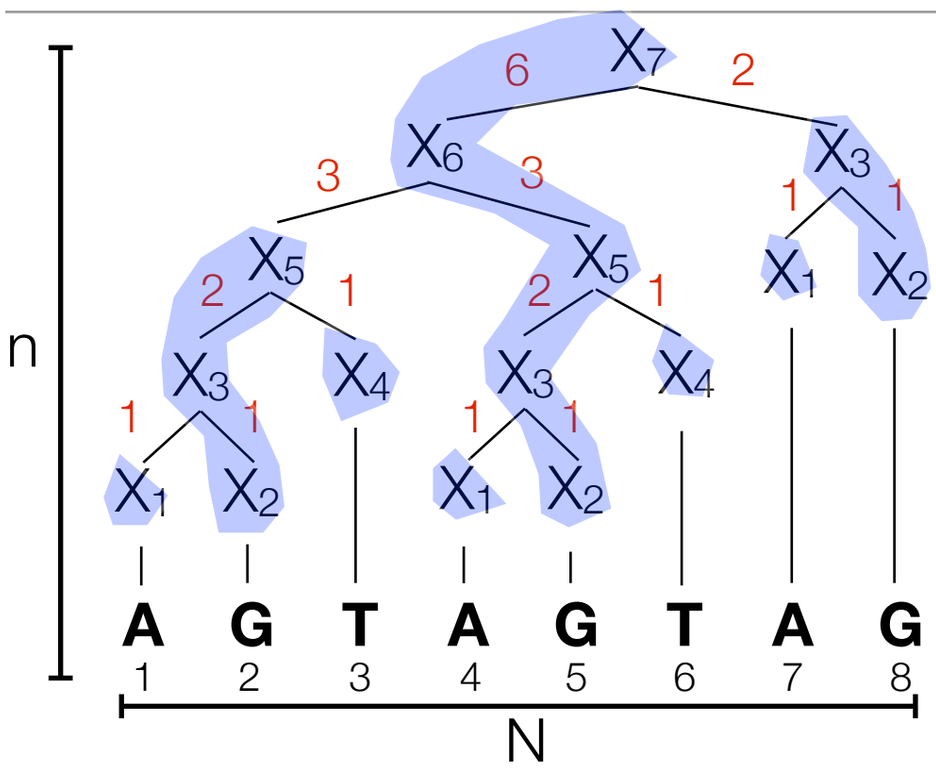
O(n) Representation of Heavy Paths

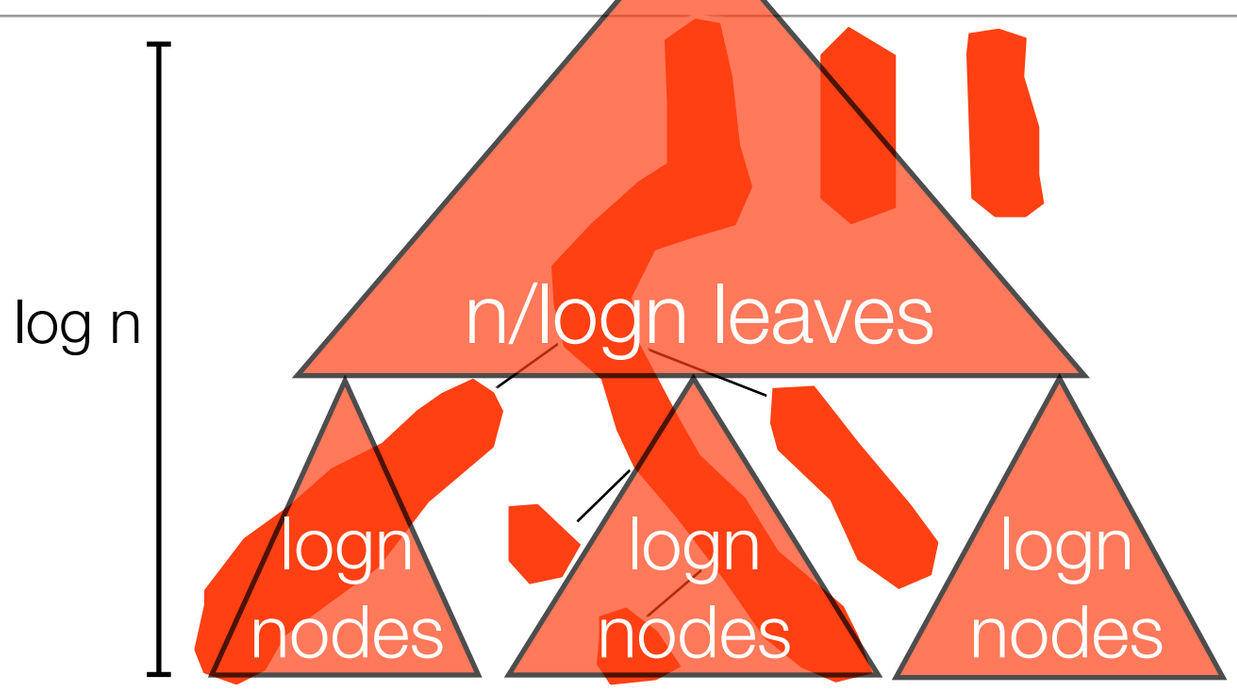




- Search for i on heavy path = lowest ancestor of distance i.
- A heavy path decomposition of heavy path decomposition.
- In-path: O(log N/x) time, total O(n) space.
- Between-paths: O(log N/x) time, total O(n log n) space.

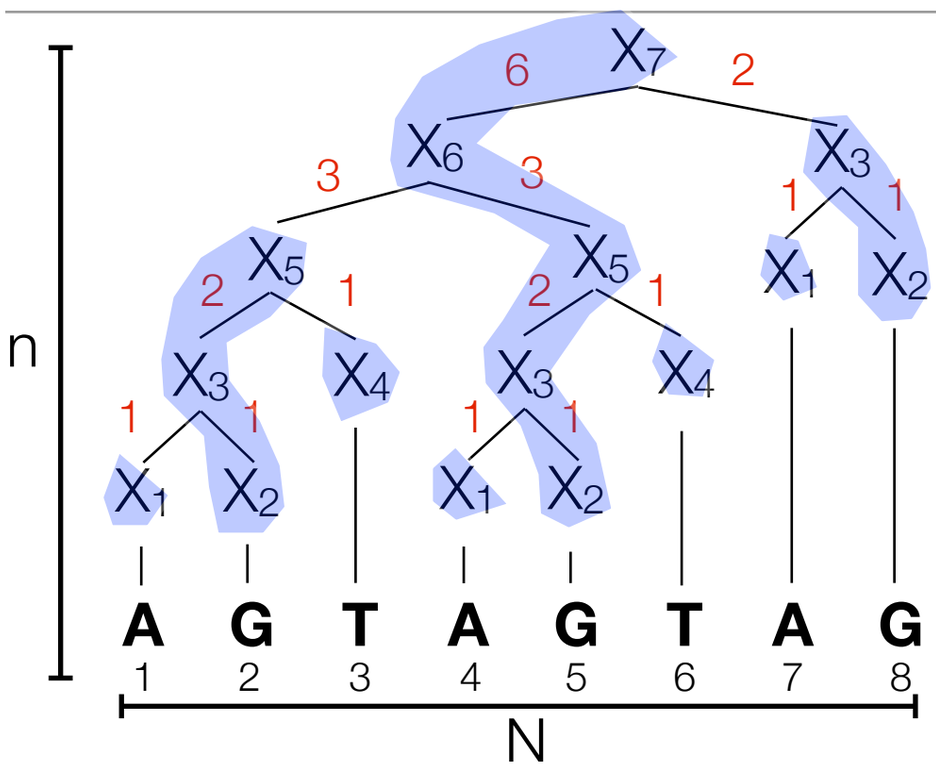
O(n) Representation of Heavy Paths

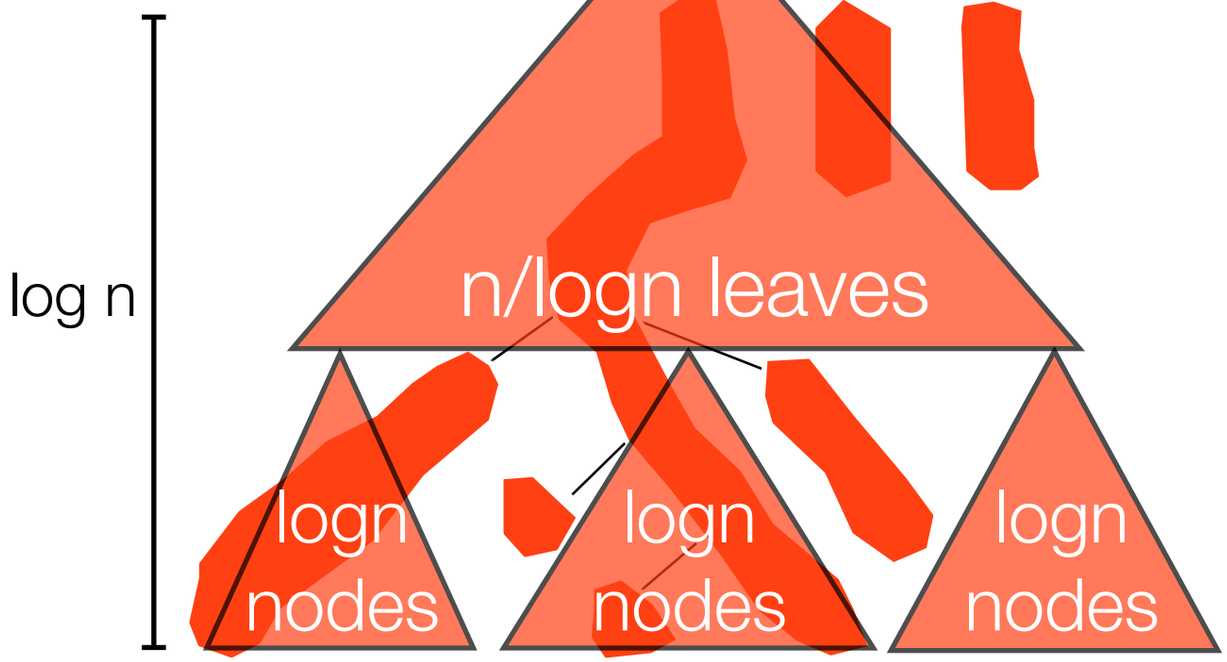




- Search for i on heavy path = lowest ancestor of distance i.
- A heavy path decomposition of heavy path decomposition.
- In-path: O(log N/x) time, total O(n) space.
- Between-paths: O(log N/x) time, total O(n log n) space.

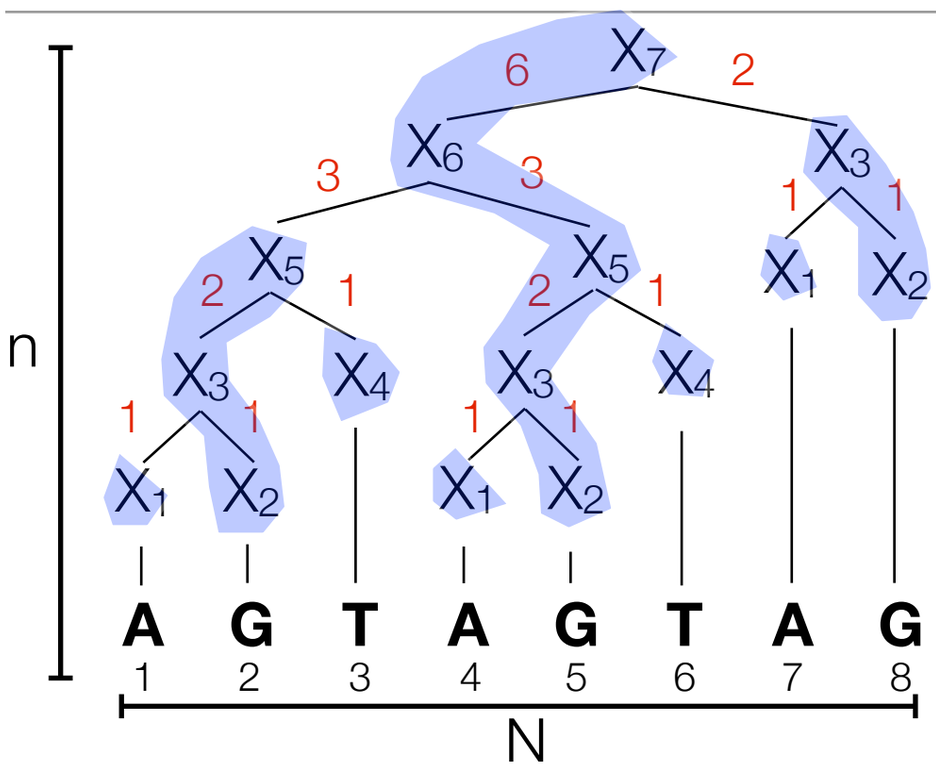
O(n) Representation of Heavy Paths

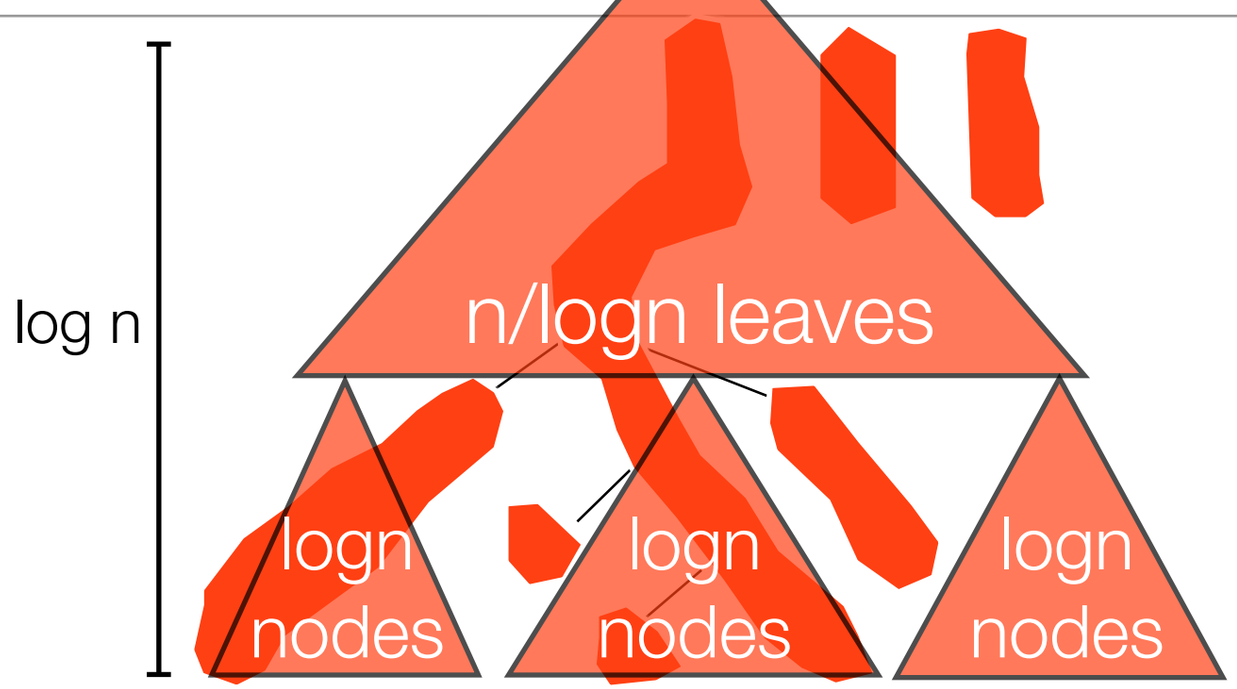




- Search for i on heavy path = lowest ancestor of distance i.
- A heavy path decomposition of heavy path decomposition.
- In-path: O(log N/x) time, total O(n) space.
- Between-paths: O(log N/x) time, total ~~O(n log n)~~ O(nα(n)) space.

O(n) Representation of Heavy Paths





- Search for i on heavy path = lowest ancestor of distance i.
- A heavy path decomposition of heavy path decomposition.
- In-path: O(log N/x) time, total O(n) space.
- Between-paths: O(log N/x) time, total ~~O(n log n)~~ space.

  ~~O(nα(n))~~

  O(n) with bittricks

Summary

- Random access and substring decompression.
  - O(n) space and O(log N + length of substring) time.
- Black compressed (approximate) string matching.
- Random access in compressed trees.