Compact suffix tree given suffix array

[Munro, Raman, Rao – J. Alg. 2001]

6.897
L 22.4
Apr. 28. 2005

- store suffix trie without "skips" (edge lengths) as binary trie on $2n+1$ nodes using $4n + o(n)$ bits as above (balanced parens.)
- as string search proceeds, maintain letter depth of node
- to descend $x \to y$:
  - compute length of edge by finding longest match between leftmost leaf(y) & rightmost leaf(y), starting at letter depth of x
  - check whether P matches these letters

$\Rightarrow O(|P| \times$ cost to lookup in suffix array$)$ time
$+ O(|\text{output}|)$ for #matches / enumerate first k matches
(leaf size; leaf select & leaf rank)

More compact suffix tree given suffix array

- use structure above on every $b = \frac{1}{2}\sqrt{\lg n}$ suffixes $\Rightarrow O(n/b)$ bits
- search narrows to a sequence of size-b blocks:

- need to find first & last match in a block.
- lookup table storing, for any b b-bit strings (in sorted order) and any ($\le b$-bit) query string, the first & last (prefix) match

$\Rightarrow O(2^{b^2} 2^b \lg b) = O(n^{1/4} 2^{\frac{1}{2}\sqrt{\lg n}} \lg\lg n) = O(n^{1/4+\varepsilon})$ bits

- now to search in a block:
  - read next b bits from b suffixes in block (via T) & from P — $O(b)$ time
  - repeat $\lceil |P|/b \rceil$ times

$\Rightarrow O(|P| + b)$ time @ bottom
↳ reducible to $\lg^{\varepsilon} n$ by making b smaller

$O(n/b)$ space total

Improved to $O\left(\frac{|P|}{\lg_{|\Sigma|} |T|} + \underbrace{\lg^{\varepsilon}_{|\Sigma|} |T|}_{\text{lookup cost in suffix array}}\right)$ query [Grossi & Vitter – STOC 2000]