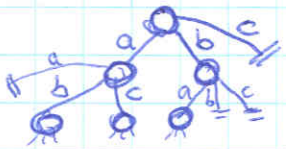
## Connection to suffixes:

allowable fold at pos. $i$
$\Leftrightarrow$ first $\min\{i-2, N-i+1\}$ characters match between
suffix of $\bar{S}$ starting at pos. $i+1$
& suffix of rev($S$) starting at pos. $N-i+2$
AND $S[1]$ or $S[N]$ (as appropriate) is $\leq$ its complement } $O(1)$ check

## Suffix tree:

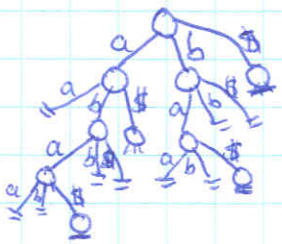
- trie = tree with paths spelling a given set of strings

{ab, ba, ac}



- suffix tree of string $S$ = trie on all suffixes of $S\$$

aba$
ba$
a$
$



- can compute suffix tree in sort(alphabet) + $O(|S|)$ [Farach 1997]
- can preprocess in $O(|S|)$ time to answer least common ancestor (lca) queries in $O(1)$ time (= longest common prefix) [Harel & Tarjan 1984]

## Suffix-tree algorithm for all-layers simple foldability

- sort lengths $\rightsquigarrow O(n)$ time if integers $= O(n)$.
  $\rightsquigarrow O(n \lg n')$ time if $n'$ distinct lengths
  $\rightsquigarrow O(n(\lg\lg n)^2)$ time if integers [Thorup 1998]
- build suffix tree on $\bar{S}$ and rev($S$). & preprocess for lca's
$\Rightarrow$ can test for allowable fold in $O(1)$ time.
(lca($\bar{S}[i+1:]$, rev($S$)$[N-i+2:]$) $\geq \min\{i-2, N-i+1\}$ AND ...)
- simultaneously look for allowable fold near two ends
- if $k$ in from end, pay $O(k)$ time & remove $\Theta(k)$ chars. from $S$
- don't need to recompute suffix tree — just change condition
$\Rightarrow O(n)$ time beyond suffix tree