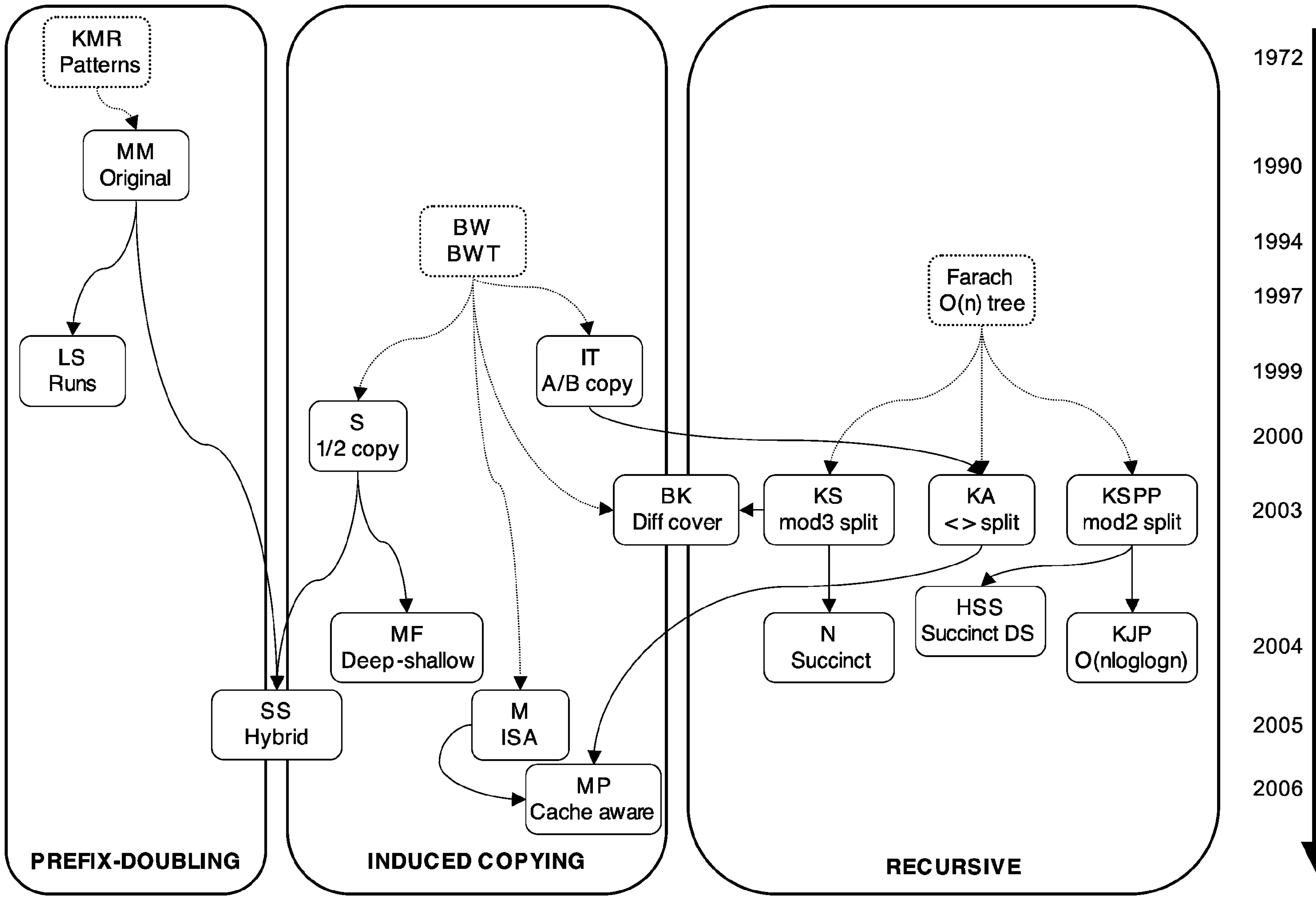
Lecture 2: Construction of Suffix Arrays

Johannes Fischer

Taxonomy



source: Puglisi/Smyth/Turpin ACM Computing Surveys ’07

Induced Sorting

- [Nong/Zhang/Chan DCC'09] **sais**-algorithm:

✓ O(n) in theory

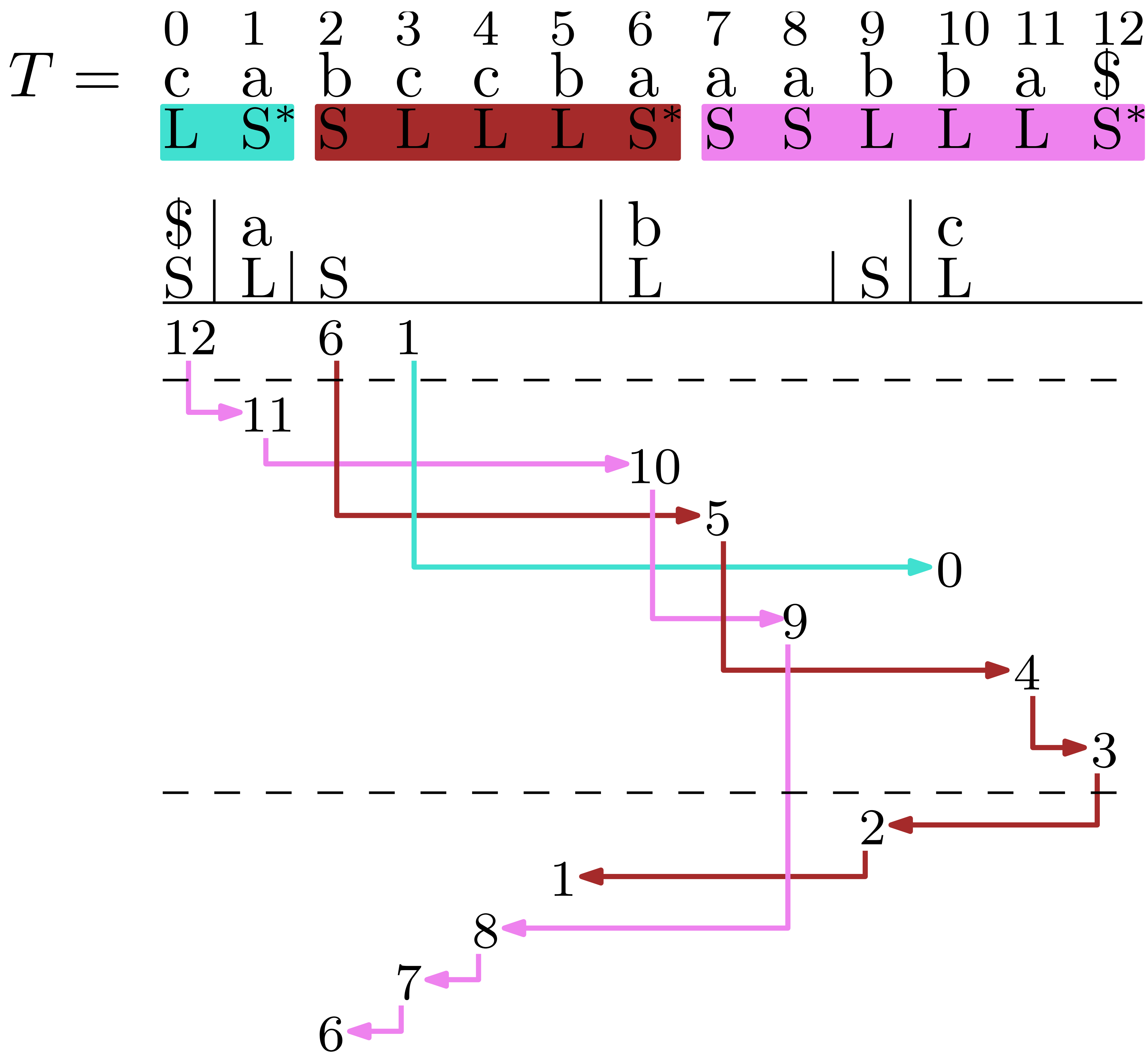
✓ fast in practice

✓ as simple as Kärkkäinen/Sanders DC3

Algorithm sais

- Definition: suffix $T[i,n]$ called
  - **S-type** iff $T[i..n] <_{lex} T[i+1..n]$ (T[n,n]='$' always S)
  - **L-type** otherwise

1. Choose sample: leftmost S (predecessor is L), $|S^*|<1/2n$
2. Sort S^*-suffixes by **recursion**
   - on new text formed by sorted S^*-substrings
3. Scan A from left to right (say we‘re at pos. i):
   - if $T[A[i]-1]$ is **L**, write $A[i]-1$ to 1st pos. in bucket
4. like (3), but sorting **S**-suffixes in a right-to-left scan
   - if $T[A[i]-1]$ is **S**, write $A[i]-1$ to **last** pos. in bucket


T = c a b c c b a a a b b a $
0 1 2 3 4 5 6 7 8 9 10 11 12
L S* S L L L S* S S L L L S*
$ a b c
S L S L S L
12 6 1
11
10
5
0
9
4
3
2
1
8
7
6

Sorting S*-Substrings

- Same algorithm, but with UNSORTED S*-suffixes

1. Choose sample: leftmost S (call them S*), |S*|<1/2n
2. Put S*-substrings in their buckets (in **text** order)
3. Scan A from left to right (say we're at pos. i):
    - if $T[A[i]-1]$ is **L**, write $A[i]-1$ to 1st pos. in bucket
4. like (3), but sorting **S**-substrings in a right-to-left scan

Correctness

- 2 main points:
  - S-substrings > L-substrings in same bucket
  - order of suffixes in reduced substring
    ≜ order in original string
- full proof: consult section 3.2 in:
  - Ge Nong, Sen Zhang, Wai Hong Chan:
    Two Efficient Algorithms for Linear Time Suffix Array Construction.
    IEEE Trans. Computers **60**(10): 1471-1484 (2011)