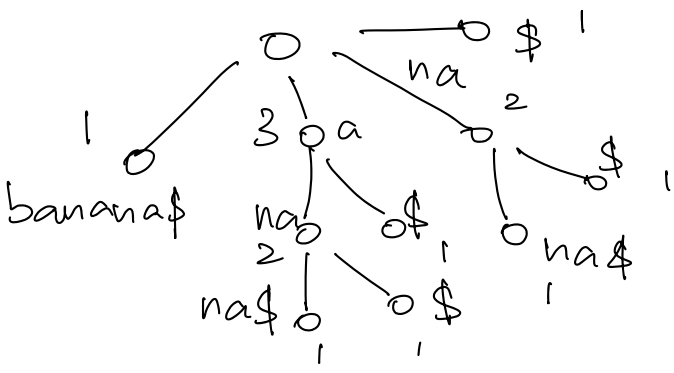
Recall: last time we talked about suffix & suffix tree.



Given string $S = S[0]S[1]\cdots S[n-1]$ over Σ

Suffix: $S_i = S[i:n) = S[i]S[i+1]\cdots S[n-1]$

It helps solve problems:

- pattern matching: find P to T
- longest common substring

Now, the question is:

How to construct suffix tree?

naive solution: create the "complete" tree then compress.

more efficient solution: suffix array.

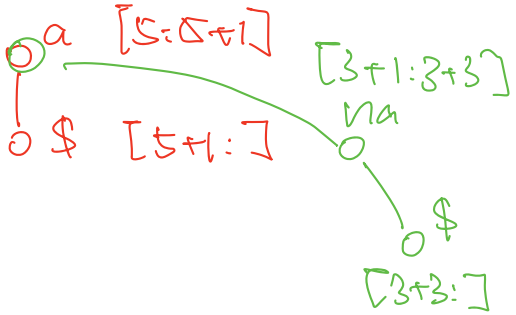
↳ Def: An array of integers $SA[0\cdots n-1]$ that contains a permutation of $[0, n-1]$ such that $S_{SA[0]} < S_{SA[1]} < \cdots < S_{SA[n-1]}$

i.e. $SA[i] = j$ → "the i-th smallest suffix starts at pos. j in the original string.

Example: b a n a n a $

0 1 2 3 4 5 6

1. sorted suffix strings ($ < \Sigma$)



length of common prefix	idx	Suffix
0 <	6	$ → shared
0 <	5	a $ → not shared
1 <	3	a n a $ → not shared
3 <	1	a n a n a $
0 <	0	b a n a n a $
0 <	4	n a $
2 <	2	n a n a $

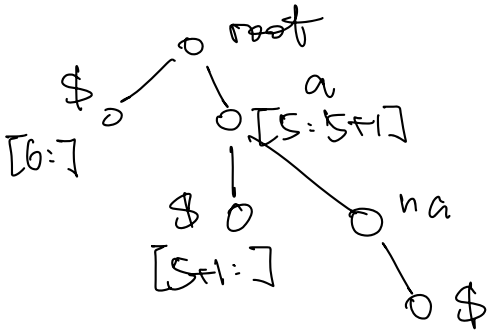
shared with prev. | shared with next

suffix array:

6 5 3 1 0 4 2

LCP: - - -

2. add nodes in order



O(n)

Suffix tree ⇐ Suffix array ← ? — String

O(n) O(n)

Skew Algorithm / "DS3" (intuition)

String: mississippi$ 0

Start index mod 3

0, 3, 6, 9 → Group0

1, 4, 7, 10 → Group1: iss, iss.ipp.i$$ → 3 2 2 1

2, 5, 8 → Group2: $si. ssi. ppi. → 5 5 4

Sort ①

(intermediate) "3321554" t' = concat 1&2

Suffix: 321065 4 ② recurse → SA12: within Group 1, 2.

"Start index in order"

inverse

S: surrogate ranks. ③

"what is the rank of Suffix at pos. j?"

321065 4

14 7 10 2 5 8

i S[i+1]: the rank of Suffix at i+1

Sort {(t[i], S[i+1])} ①

SA 0

0: (m, 3)

3: (s, 2)

6: (s,

merge ④

SA

1. Sort — Radix Sort

1 Sorting Group 0. {(char, int)}.

problem: Sort a list of length n of pairs (x_i, y_i)

where $0 \le x_i, y_i \le n-1$.

algo: 1. create n buckets. put the pairs in buckets by y_i.

2. process buckets in increasing order, putting the new pairs into new buckets by x_i.

3. read off the sorted list

eg: (3,1)(1,5)(5,5)(5,3)(3,1)(1,5)(5,0)

Step 1:

0 : ~~(5,0)~>~~

1 : ~~(3,1)(3,1)~>~~

2

3 : (5,3)→

4

5 : (1,5)(5,5)(1,5)→

6

Step 2:

0

1 : (1,5)(1,5)

2 : (3,1)(3,1)

3

4

5 : (5,0)(5,3)(5,5)

6

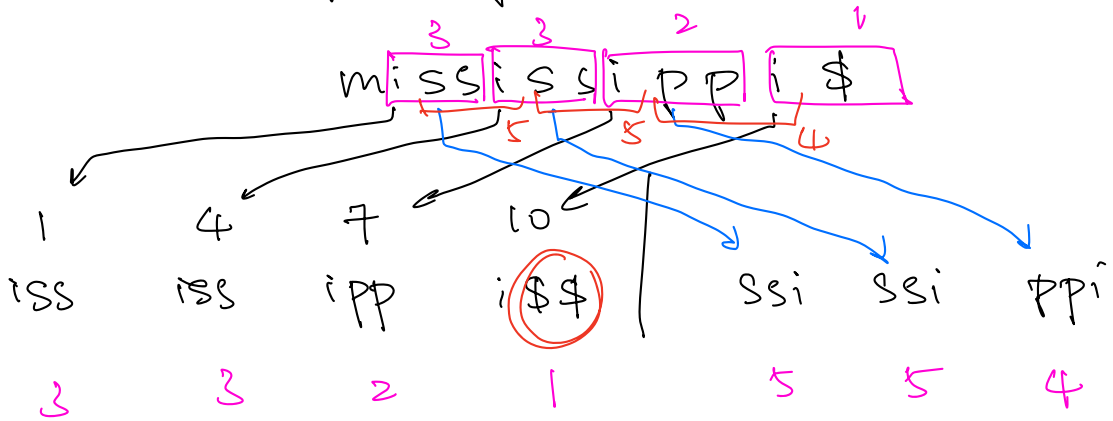
Sorted

2 Sort 3-charac list

{(S[i], S[i+1], S[i+2])}

2. encoded t → suffix array

Claim: t's suffix array is SA^{12}.



Q. $i \bmod 3 = 1$ $j \bmod 3 = 0 = 2$ $= 0 =$

$S_i = (t[i] \cdot S_{i+1})$ $S_j = (t[j] S_{j+1})$

$(S[i], S[i+1], S_{t+2})$ $(S[j], S[j+1], S_{t+2})$

Group 1 Group 2

Runtime $T(n) = O(n) + T(\lceil 2n/3 \rceil)$

recursive calls. ②

$2n/3$ → There can be no more lexicographic names than characters in S^{12}, so the alphabet size in a recursive call never exceeds the size of the string.