As usual, show your work.
Due March 8th, 2022, 11:59PM.

Problem 1

The bus

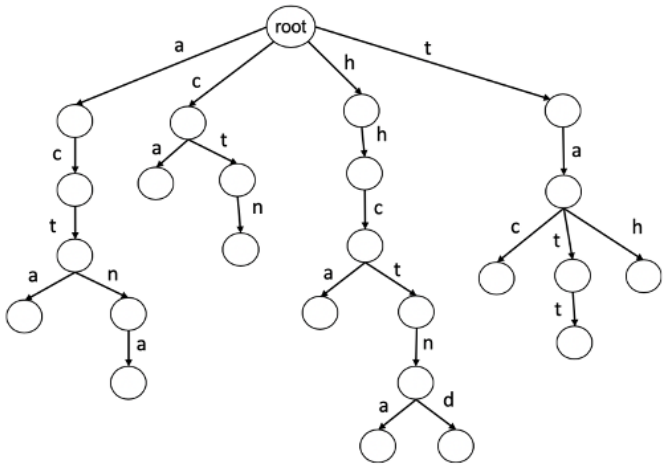
[2 points]

We have following collection of strings called Patterns and a string called Text.
Text := tactnahhctndhhctna
Patterns :=

1. tah
2. acta
3. hhctnd
4. ca
5. hhca
6. tatt
7. tac
8. actna
9. hhctna
10. ctn

Construct *Trie(Patterns)* using the algorithm we learned in class. How many leaves does the trie have?

Solution:



There are 10 leaves, one for each pattern.

Problem 2

Bus stops

[3 points]

Perform TrieMatching between *Text* and *Trie(Patterns)* you generated in question 1 according to the algorithm we learned in class. That is, find all substrings of *Text* that match any strings in Patterns. Show all iterations of the TrieMatching algorithm; at each iteration, show the input and output by TrieMatching. How many matches were found? That is, how many substrings of *Text* matched one of strings in Patterns?

Solution:

7 matches:

i	text	output
1	tactnahhctndhhctna	tac
2	actnahhctndhhctna	actna
3	ctnahhctndhhctna	ctn
4	tnahhctndhhctna	**no match**
5	nahhctndhhctna	**no match**
6	ahhctndhhctna	**no match**
7	hhctndhhctna	hhctnd
8	hctndhhctna	**no match**
9	ctndhhctna	ctn
10	tndhhctna	**no match**
11	ndhhctna	**no match**
12	dhhctna	**no match**
13	hhctna	hhctna
14	hctna	**no match**
15	ctna	ctn
16	tna	**no match**
17	na	**no match**
18	a	**no match**

Problem 3

Suffix trees

[5 points]

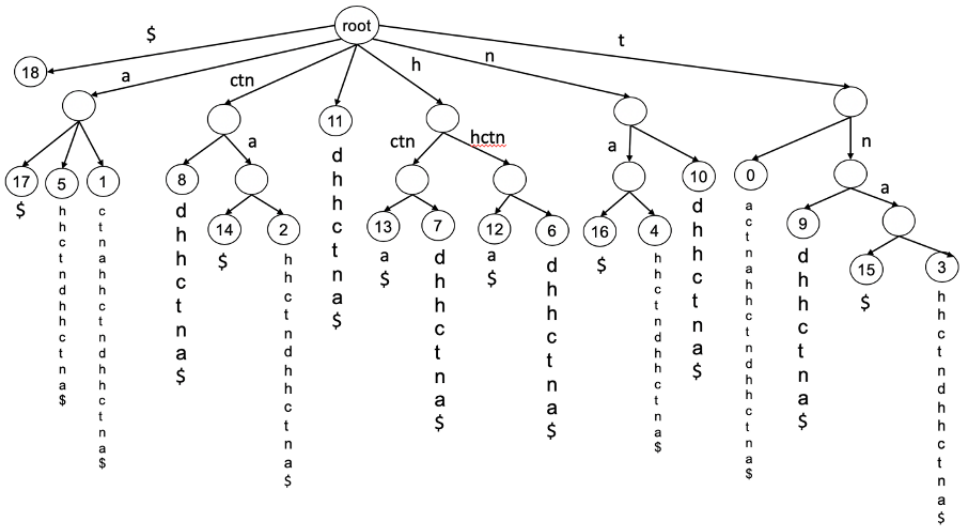
Create a suffix tree of *Text*. Show all suffixes of *Text* first and draw the tree. How many leaves are in the suffix tree? Is it the same as the number of leaves in Trie(Pattern)? Why or why not?

Solution

First, add $ to the string *Text*. Next, list all suffixes. If you want to be pedantic, you should have $ as a suffix, but I won't be upset if you don't. The suffixes:

$
a$
na$
tna$
ctna$
hctna$
hhctna$
dhhctna$
ndhhctna$
tndhhctna$
ctndhhctna$
hctndhhctna$
hhctndhhctna$
ahhctndhhctna$
nahhctndhhctna$
tnahhctndhhctna$
ctnahhctndhhctna$
actnahhctndhhctna$
tactnahhctndhhctna$

Now, the corresponding suffix tree:



There are 19 leaves, and the number of leaves is different between the suffix tree and trie because the number of leaves in the suffix tree is the length of *Text* + 1 while the number of leaves in trie is the number of strings in *Patterns*.

Problem 4

Suffix arrays

[2 points]

Create a suffix array of *Text*. Show sorted suffixes and their starting positions.

Solution:

Sorted suffixes	**Starting positions**
$	18
a$	17
actnahhctndhhctna$	1
ahhctndhhctna$	5
ctna$	14
ctnahhctndhhctna$	2
ctndhhctna$	8
dhhctna$	11
hctna$	13
hctndhhctna$	7
hhctna$	12
hhctndhhctna$	6
na$	16
nahhctndhhctna$	4
ndhhctna$	10
tactnahhctndhhctna$	0
tna$	15
tnahhctndhhctna$	3
tndhhctna$	9

Problem 5

BWT

[5 points]

Create the Burrows-Wheeler transform of *Text*. First, create cyclic rotations matrix. Second, create M(*Text*) matrix. Lastly, output BWT(*Text*) from M(*Text*).

Solution:

First compute the cyclic permutations of tactnahhctndhhctna$

$tactnahhctndhhctna
a$tactnahhctndhhctn
na$tactnahhctndhhct
tna$tactnahhctndhhc
ctna$tactnahhctndhh
hctna$tactnahhctndh
hhctna$tactnahhctnd
dhhctna$tactnahhctn
ndhhctna$tactnahhct
tndhhctna$tactnahhc
ctndhhctna$tactnahh
hctndhhctna$tactnah
hhctndhhctna$tactna
ahhctndhhctna$tactn
nahhctndhhctna$tact
tnahhctndhhctna$tac
ctnahhctndhhctna$ta
actnahhctndhhctna$t
tactnahhctndhhctna$

Now sort to get M(*Text*):

$tactnahhctndhhctn***a***
a$tactnahhctndhhct***n***
actnahhctndhhctna$***t***
ahhctndhhctna$tact***n***
ctna$tactnahhctndh***h***
ctnahhctndhhctna$t***a***
ctndhhctna$tactnah***h***
dhhctna$tactnahhct***n***
hctna$tactnahhctnd***h***
hctndhhctna$tactna***h***
hhctna$tactnahhctn***d***
hhctndhhctna$tactn***a***
na$tactnahhctndhhc***t***
nahhctndhhctna$tac***t***
ndhhctna$tactnahhc***t***
tactnahhctndhhctna***$***
tna$tactnahhctndhh***c***
tnahhctndhhctna$ta***c***
tndhhctna$tactnahh***c***

The Burrows-Wheeler Transform is in the final column of M(*Text*): antnhahnhhdattt$ccc.

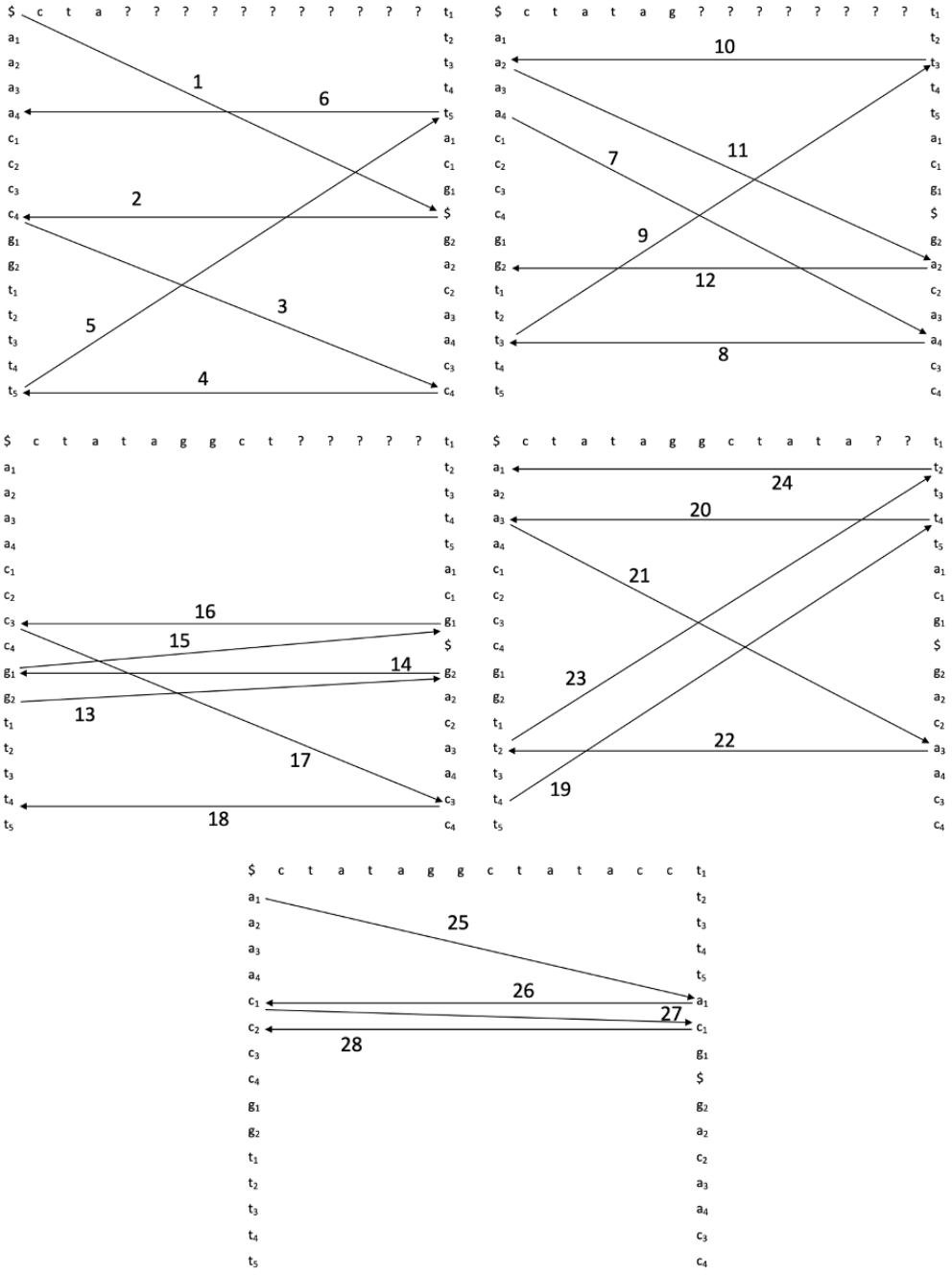
Problem 6

BWT traversal

[5 points]

Reconstruct the string whose Burrows-Wheeler transform is *tttttacg*$*gacaacc*. Show how you reconstruct each letter using the First-Last property in each iteration as we learned in class (refer to Figure 9.12 in textbook). In other words, show the partial M(*Text*) matrix and the two arrows as in Figure 9.12 to indicate each letter you are reconstructing. Be sure to label the edges according to the order of how they should be traversed.

Solution:


$ c t a ? ? ? ? ? ? ? ? ? ? t1
a1 a2 a3 a4 c1 c2 c3 c4 g1 g2 t1 t2 t3 t4 t5
t1 t2 t3 t4 t5 a1 c1 g1 $ g2 a2 c2 a3 a4 c3 c4
1 6 2 3 5 4
$ c t a t a g ? ? ? ? ? ? ? t1
10 11 7 9 12 8
$ c t a t a g g c t ? ? ? ? ? t1
16 15 14 13 17 18
$ c t a t a g g c t a t a ? ? t1
24 20 21 23 22 19
$ c t a t a g g c t a t a c c t1
25 26 27 28

Problem 7

BWMatching

[8 points]

Using the M(*Text*) matrix created in question 5, perform BWMatching between *Text* and the following 2 strings in Patterns. First, create a table that has i, FirstColumn, LastColumn, and LastToFirst(i) as in Figure 9.15 in textbook (or as in lecture slides). Then, for each string in Pattern, show the values of top and bottom variables in each iteration of BWMatching, as in Figure 9.14 in textbook. How many times does each string in Pattern appear in *Text*?

Patterns:

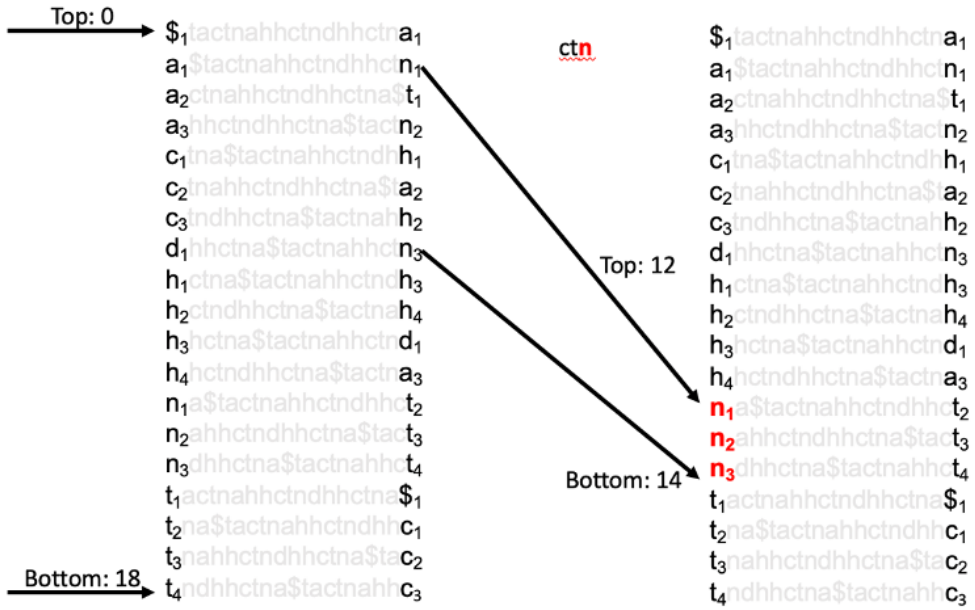
1. ctn
2. cna

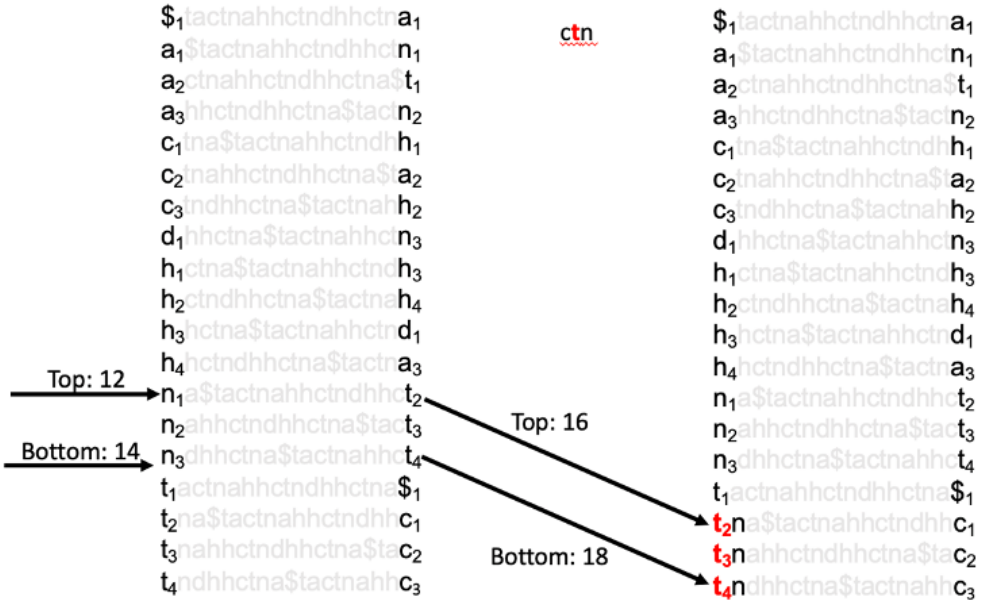
Solution:

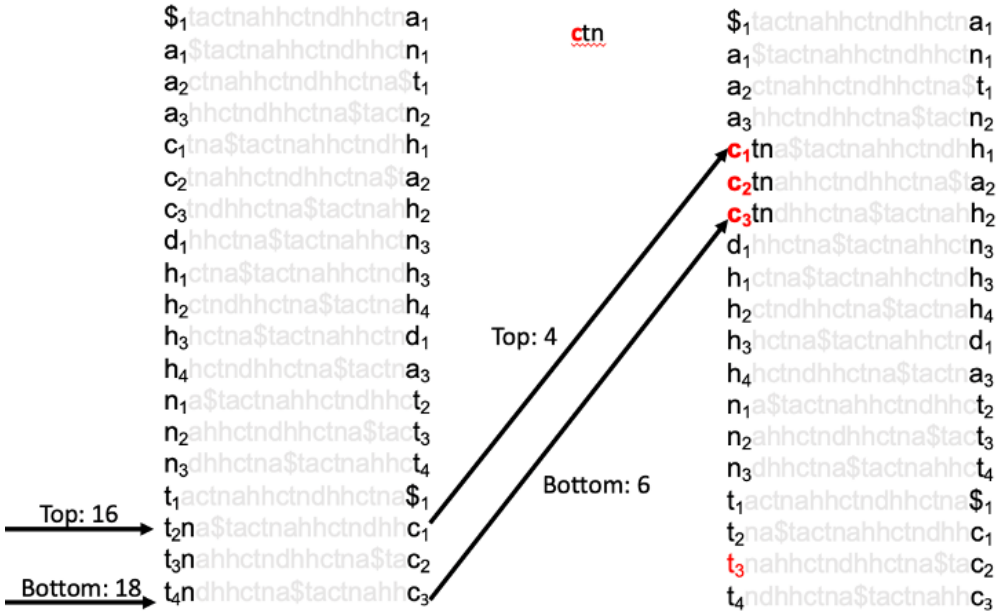
First, compute the mapping of FirstColumn, LastColumn, and LastToFirst.

i	FirstColumn	LastColumn	LastToFirst(i)
0	\$1	a_1	1
1	a_1	n_1	12
2	a_2	t_1	15
3	a_3	n_2	13
4	c_1	h_1	8
5	c_2	a_2	2
6	c_3	h_2	9
7	d_1	n_3	14
8	h_1	h_3	10
9	h_2	h_4	11
10	h_3	d_1	7
11	h_4	a_3	3
12	n_1	t_2	16
13	n_2	t_3	17
14	n_3	t_4	18
15	t_1	$\$_1$	0
16	t_2	c_1	4
17	t_3	c_2	5
18	t_4	c_3	6

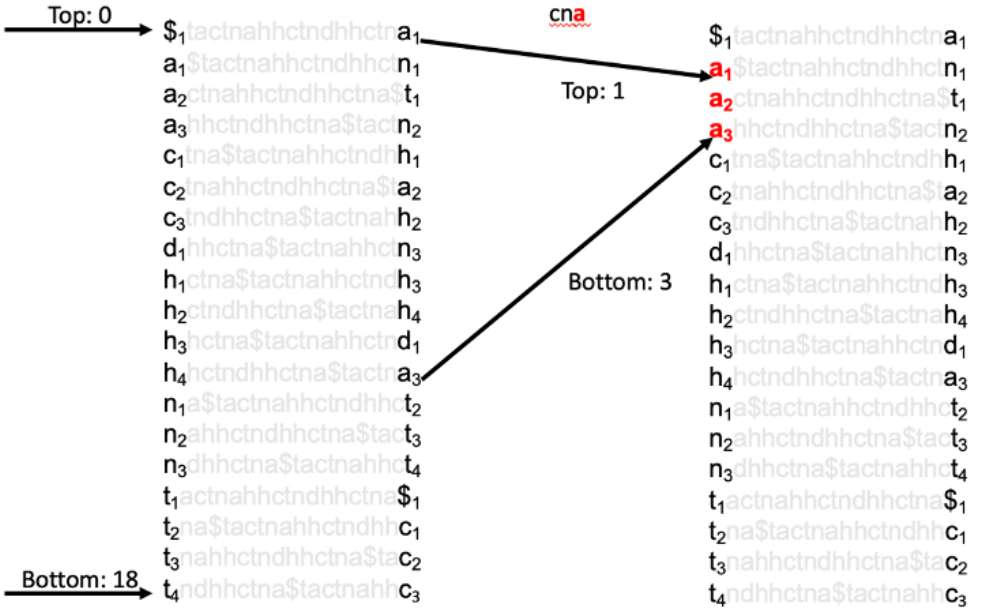
Now the matches for pattern *cta:*

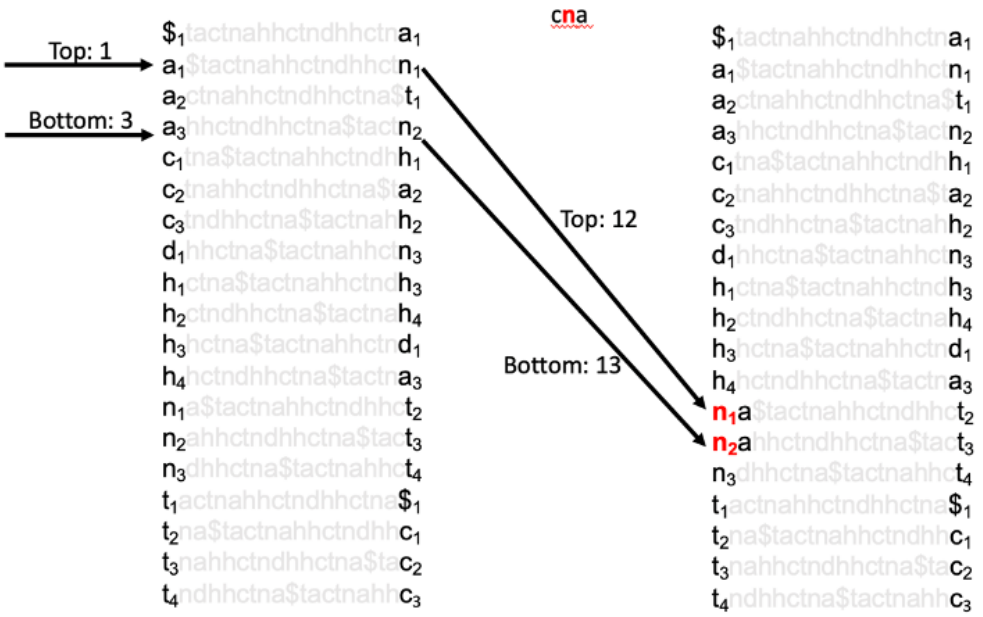

Top: 0
Bottom: 18
ctn
Top: 12
Bottom: 14

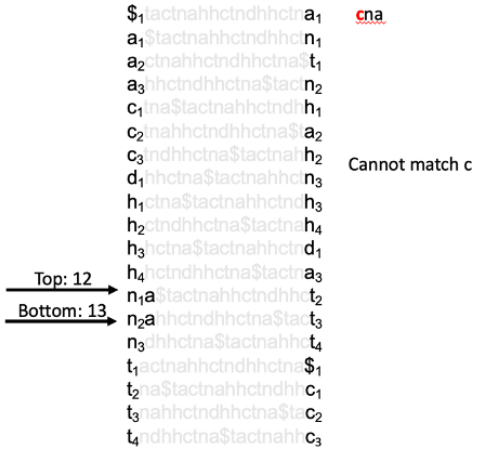

Top: 12
Bottom: 14
ctn
Top: 16
Bottom: 18



Finally, we see that *ctn* appears 3 times in *Text*.
Next, we look at *cna*.







Annoyingly, we get to the end and find no match.