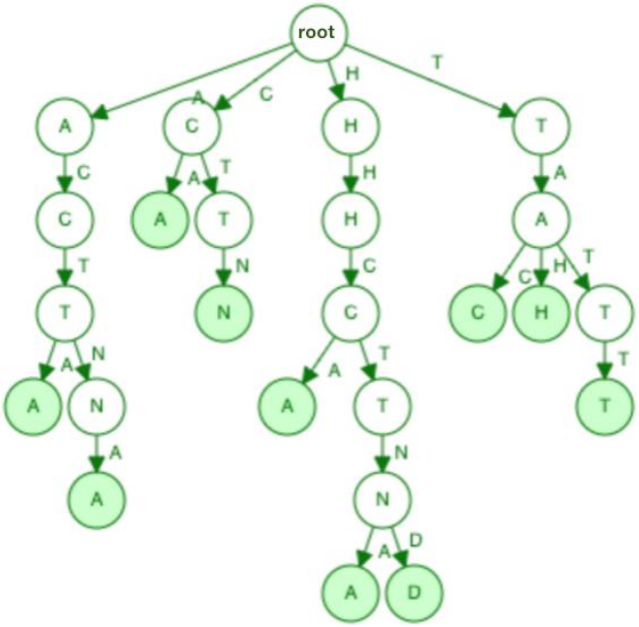
Problem 1. The bus.

Sol: As we can see from the Trie, it has 10 leaves.



Problem 2. Bus stops

Text := tactnahhctndhhctna

First iteration:
Input letter t, go through Trie with node t, then keep going we found pattern tac.
Second iteration:
Input letter a, go through Trie with node a, then sweep through, we found pattern actna.
Third iteration:
Input letter c, go through Trie with node c, then sweep through, we found pattern ctn.
Fourth iteration:
Input letter t, go through Trie with node t, then sweep through, we couldn't find any pattern.
Fifth iteration:
Input letter n, go through Trie, no node n are found.
Sixth iteration:
Input letter a, go through Trie with node a, then sweep through, we couldn't find any pattern.
Seventh iteration:
Input letter h, go through Trie with node h, then sweep through, we found pattern hhctnd.
Eighth iteration:
Input letter h, go through Trie with node h, then sweep through, we couldn't find any pattern.
Ninth iteration:
Input letter c, go through Trie with node c, then sweep through, we found pattern ctn.
Tenth iteration:
Input letter t, go through Trie with node t, then sweep through, we couldn't find any pattern.

Eleventh iteration:

Input letter n, no node n are found.

Twelfth iteration:

Input letter d, no node d are found.

Thirteenth iteration:

Input letter h, go through Trie with node h, then sweep through, we found pattern hhctna.

Fourteenth iteration:

Input letter h, go through Trie with node h, then sweep through, we couldn't find any pattern.

Fifteenth iteration:

Input letter c, go through Trie with node c, then sweep through, we found pattern ctn.

Sixteenth iteration:

Input letter t, go through Trie with node h, then sweep through, we couldn't find any pattern.

Seventeenth iteration:

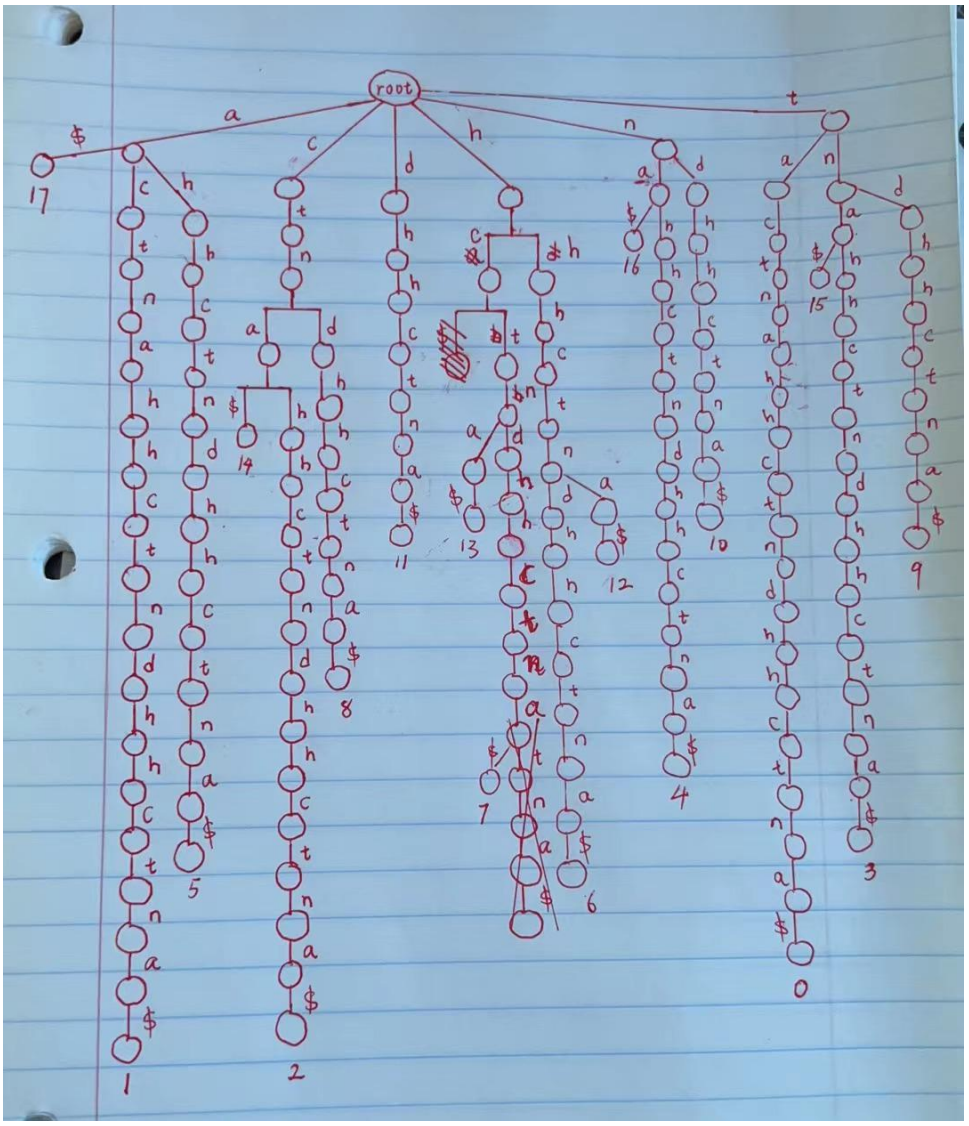
Input letter n, no node n are found.

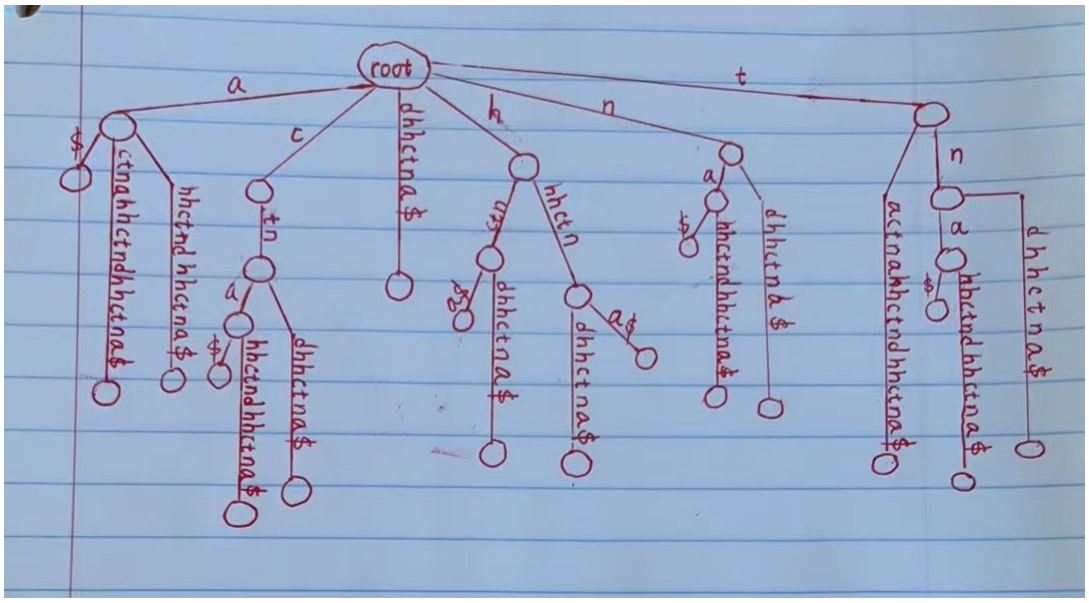
Eighteenth iteration:

Input letter a, go through Trie with node a, then sweep through, we couldn't find any pattern.

In total of 7 matches were found by running the Triematch algorithm.

Problem 3. Suffix trees





It has 18 leaves. No the number of leaves is not the same as the Trie in problem 1 because the Trie is built based on the patterns; therefore, its number of leaves equals to the number of patterns. The suffix tree, on the other hand, is built upon the entire string input, which has 18 characters.

Problem 4. Suffix arrays

Users > zacyou > Desktop > JS test.js > ...

```
1
2
3   function generateSubstrings(string){
4       let res = {};
5
6       for(let i = 0; i < string.length; i++){
7           res[string.substring(i)] = i;
8       }
9
10      return res;
11  }
12
13  function returnSortedSuffixArray(obj){
14      let suffixArr = [];
15      let arr = [...Object.keys(obj)];
16      arr.sort();
17
18      for(let i = 0; i < arr.length; i++){
19          suffixArr.push(obj[arr[i]])
20      }
21
22      return suffixArr;
23  }
24
25  let text = 'tactnahhctndhhctna';
26  let substrings = generateSubstrings(text);
27  let suffixArr = returnSortedSuffixArray(substrings);
28
29  console.log(substrings)
30  console.log(suffixArr)
```

TERMINAL JUPYTER PROBLEMS 5 OUTPUT DEBUG CONSOLE

```
[Running] node "/Users/zacyou/Desktop/test.js"
{
  tactnahhctndhhctna: 0,
  actnahhctndhhctna: 1,
  ctnahhctndhhctna: 2,
  tnahhctndhhctna: 3,
  nahhctndhhctna: 4,
  ahhctndhhctna: 5,
  hhctndhhctna: 6,
  hctndhhctna: 7,
  ctndhhctna: 8,
  tndhhctna: 9,
  ndhhctna: 10,
  dhhctna: 11,
  hhctna: 12,
  hctna: 13,
  ctna: 14,
  tna: 15,
  na: 16,
  a: 17
}
[
  17,  1,  5, 14,  2, 8, 11,
  13,  7, 12,  6, 16, 4, 10,
   0, 15,  3,  9
]

[Done] exited with code=0 in 0.229 seconds
```

Problem 5. BWT

As the result below, the first matrix printed is the cyclic rotation matrix and the second matrix is the M matrix which is sorted, and the BWT text is the last output.

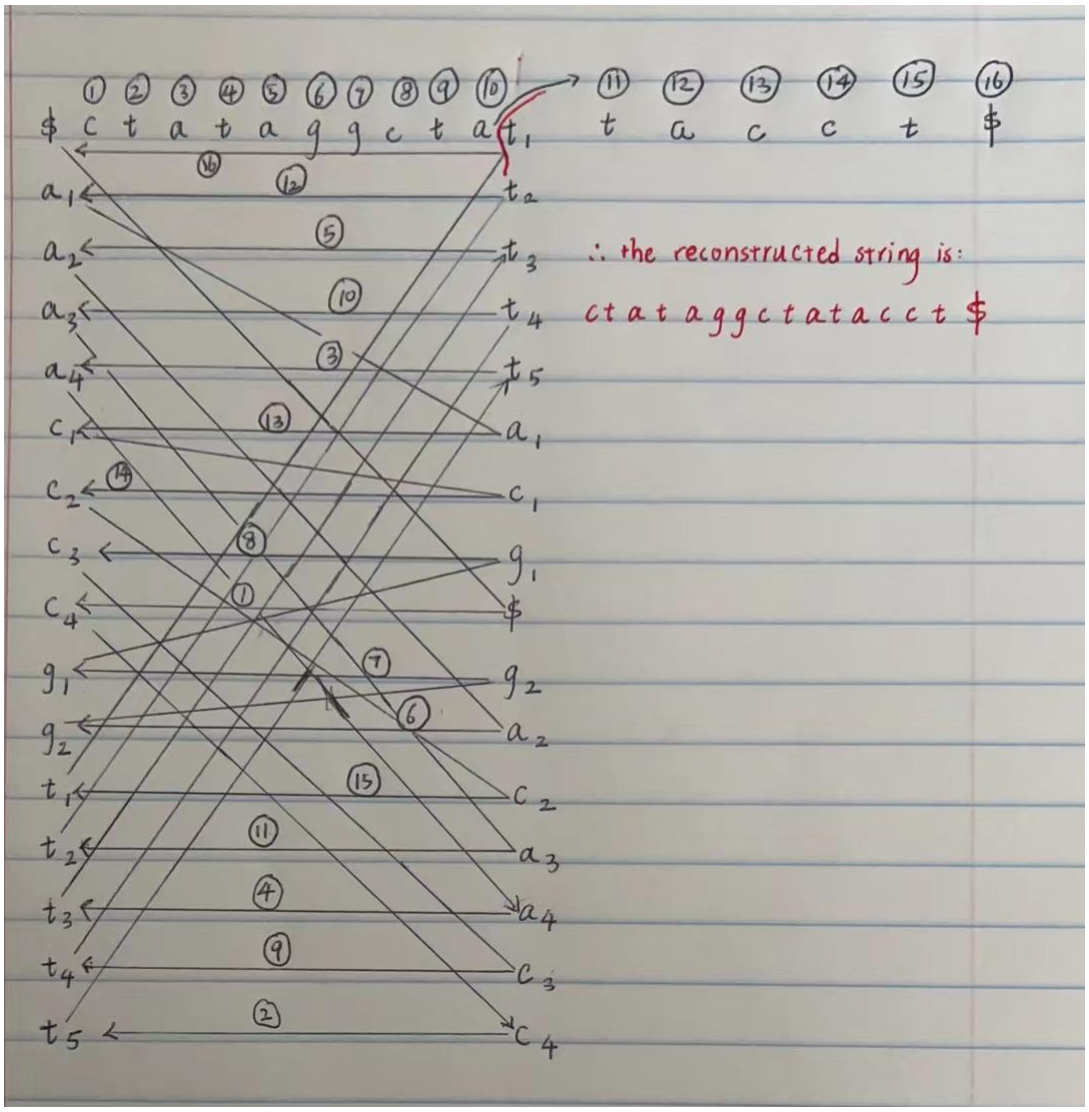
Users > zacyou > Desktop > JS test.js > ...

```
let text = 'tactnahhctndhhctna';
function generateCyclicRotation(text){
    res = [];
    text += '$';
    for(let i = 0; i < text.length; i++){
        let substr = text.substring(0, i)
        let new_arr = text.slice(i) + substr

        res.push(new_arr)

    }

    return res;
}

function getBWTTextByCol(arr, col){
   let res = []
   for(let i = 0; i < arr.length; i++){
        res.push(arr[i].charAt(col))
   }
   return res;
}

bwt = generateCyclicRotation(text)
bwt_sorted = [...bwt].sort()
bwt_text = getBWTTextByCol(bwt_sorted, text.length)
console.log('The cyclic rotation of the text is: \n', bwt);
console.log('The sorted M matrix is: \n', bwt_sorted);
console.log('The BWT text is: \n', bwt_text)
```

TERMINAL JUPYTER PROBLEMS OUTPUT DEBUG CONSOLE

```
The cyclic rotation of the text is:
 [
  'tactnahhctndhhctna$', 'actnahhctndhhctna$t',
  'ctnahhctndhhctna$ta', 'tnahhctndhhctna$tac',
  'nahhctndhhctna$tact', 'ahhctndhhctna$tactn',
  'hhctndhhctna$tactna', 'hctndhhctna$tactnah',
  'ctndhhctna$tactnahh', 'tndhhctna$tactnahhc',
  'ndhhctna$tactnahhct', 'dhhctna$tactnahhctn',
  'hhctna$tactnahhctnd', 'hctna$tactnahhctndh',
  'ctna$tactnahhctndhh', 'tna$tactnahhctndhhc',
  'na$tactnahhctndhhct', 'a$tactnahhctndhhctn',
  '$tactnahhctndhhctna'
]
The sorted M matrix is:
 [
  '$tactnahhctndhhctna', 'a$tactnahhctndhhctn',
  'actnahhctndhhctna$t', 'ahhctndhhctna$tactn',
  'ctna$tactnahhctndhh', 'ctnahhctndhhctna$ta',
  'ctndhhctna$tactnahh', 'dhhctna$tactnahhctn',
  'hctna$tactnahhctndh', 'hctndhhctna$tactnah',
  'hhctna$tactnahhctnd', 'hhctndhhctna$tactna',
  'na$tactnahhctndhhct', 'nahhctndhhctna$tact',
  'ndhhctna$tactnahhct', 'tactnahhctndhhctna$',
  'tna$tactnahhctndhhc', 'tnahhctndhhctna$tac',
  'tndhhctna$tactnahhc'
]
The BWT text is:
 [
  'a', 'n', 't', 'n', 'h',
  'a', 'h', 'n', 'h', 'h',
  'd', 'a', 't', 't', 't',
  '$', 'c', 'c', 'c'
]
```

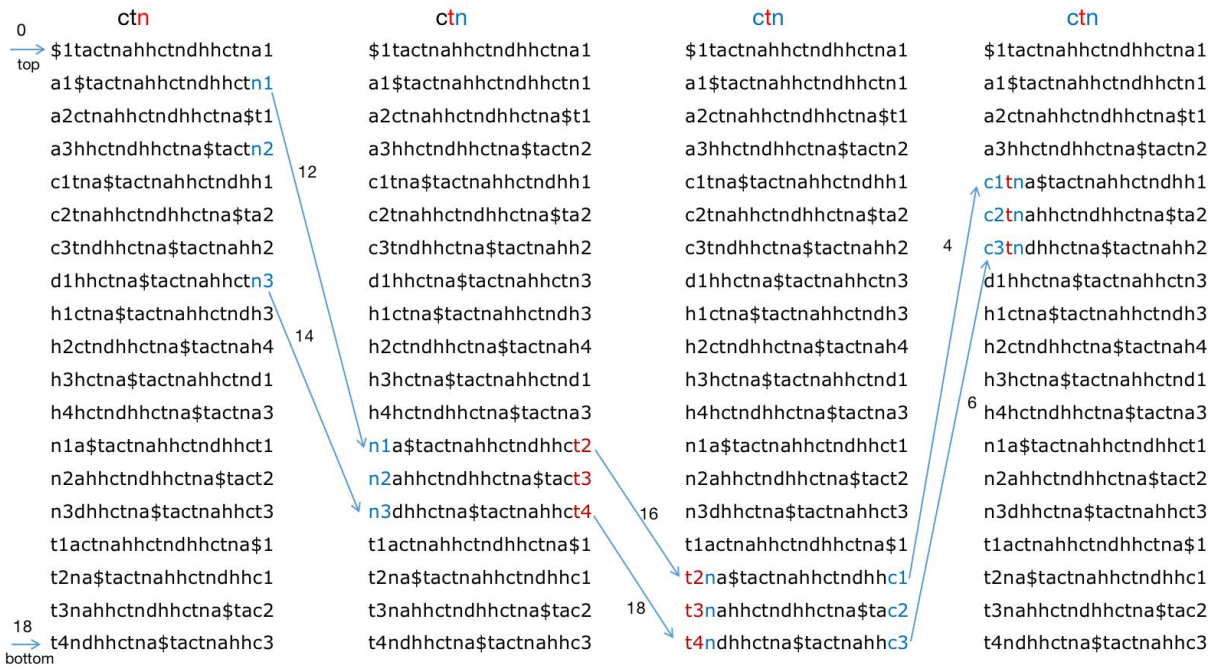
Problem 6. BWT traversal



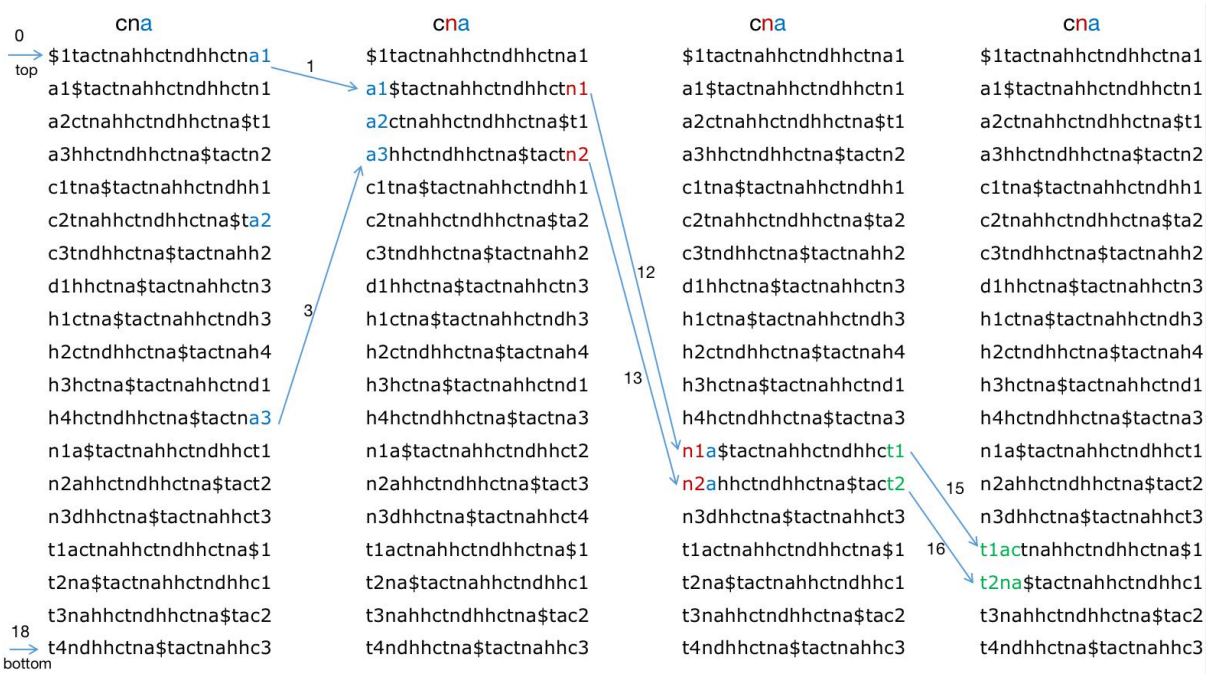
Problem 7. BWMatching

i	FirstColumn	LastColumn	LastToFirst(i)
0	$	a1	1
1	a1	n1	12
2	a2	t1	15
3	a3	n2	13
4	c1	h1	8
5	c2	a2	2
6	c3	h2	9
7	d1	n3	14
8	h1	h3	10
9	h2	h4	11
10	h3	d1	7
11	h4	a3	3

12	n1	t2	16
13	n2	t3	17
14	n3	t4	18
15	t1	$	0
16	t2	c1	4
17	t3	c2	5
18	t4	c3	6



Pattern ‘ctn’ appears 3 times.



No pattern ‘cna’ is found.