The git repo for this project is at:

https://github.com/MrHencke/DAT158-Algoritmer-Oblig1

Problem 1

Compare String	a	a	a	b	a	a	d	a	a	b	a	a	a
Comparisons													
1	a	a	b	a	a	a							
2		a	a	b	a	a	a						
3			a	a	b	a	a	a					
4				a	a	b	a	a	a				
5					a	a	b	a	a	a			
6						a	a	b	a	a	a		
7							a	a	b	a	a	a	
8								a	a	b	a	a	a

Problem 2

a)

Compare String	a	a	a	b	a	a	d	a	a	b	a	a	a
Comparisons													
1	a	a	b	a	a	a							
2		a	a	b	a	a	a						
3								a	a	b	a	a	a

b) From my testing I found that the average number of comparisons per character for a pattern string of length 5. Was in the range of 0.15 to 0.4 comparisons per character, this was both in Norwegian and English. I tested with variable comparison strings, but always a 5 characters long pattern. See the repo for the code written for this **Problem**.

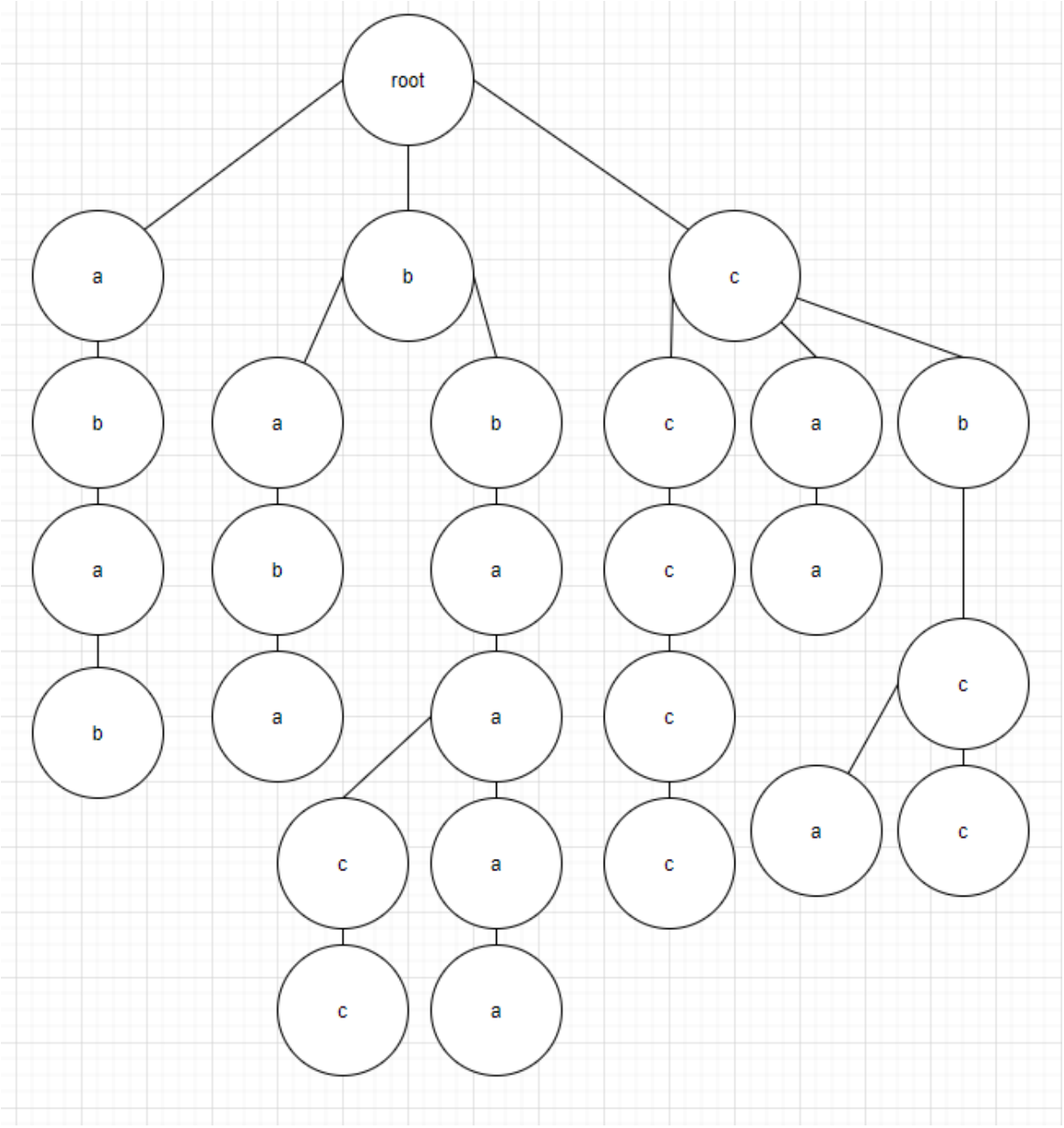
Problem 3

Compare String	a	a	a	b	a	a	d	a	a	b	a	a	a
Comparisons													
1	a	a	b	a	a	a							
2		a	a	b	a	a	a						
3					a	a	b	a	a	a			
4						a	a	b	a	a	a		
5							a	a	b	a	a	a	
6								a	a	b	a	a	a

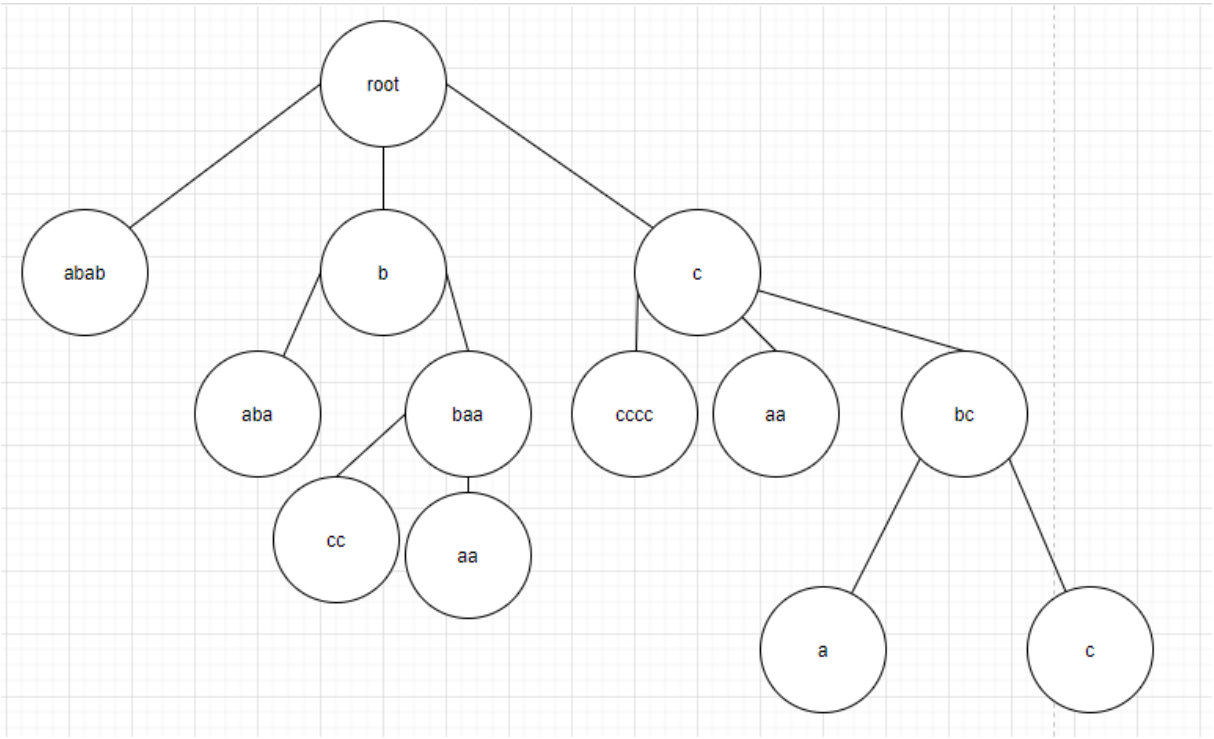
Problem 4

There are 3: “a”, “aa” and “aaa” are the only non-empty prefixes of P that are also suffixes of P.

Problem 5

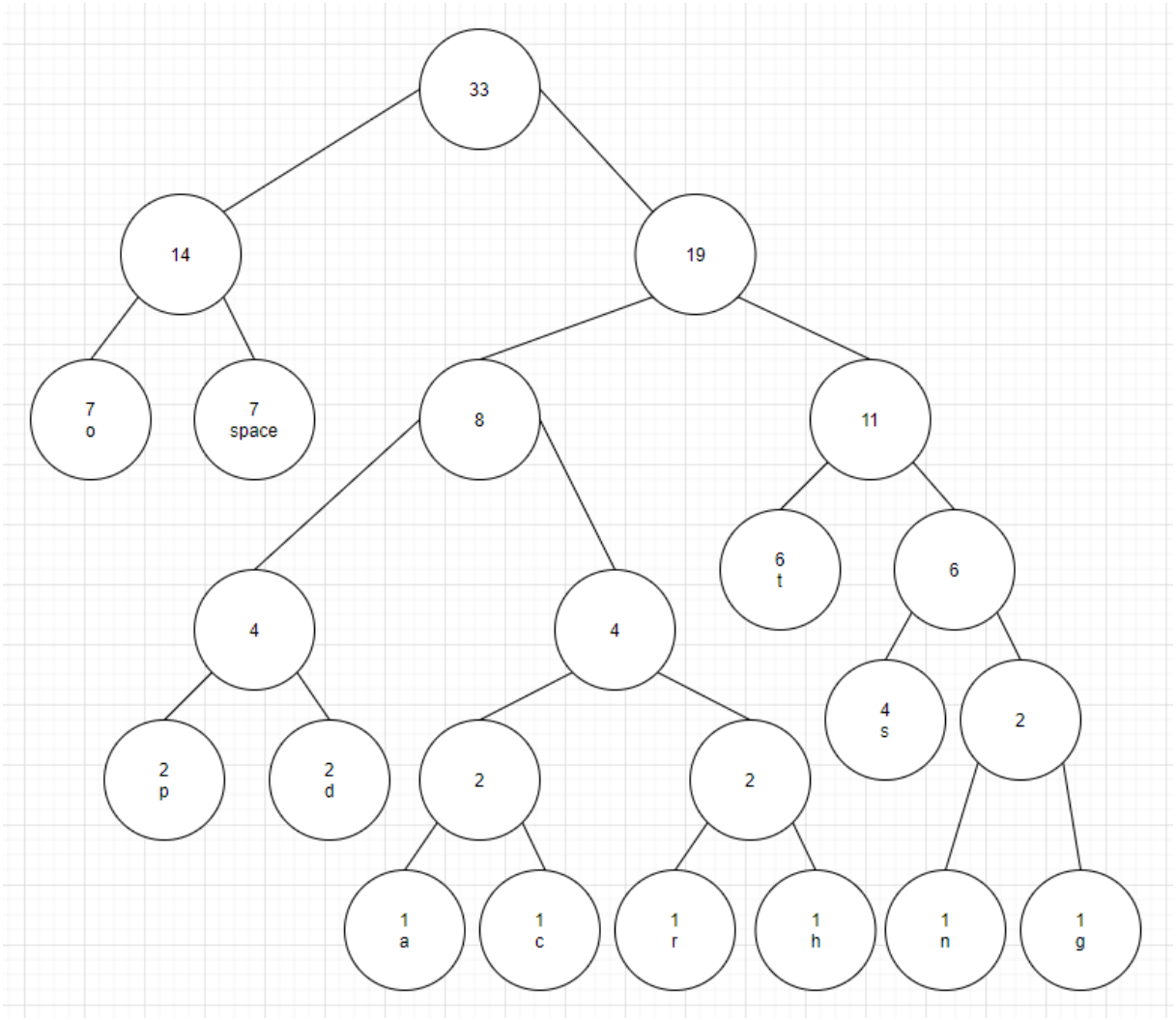


Problem 6



Problem 7

Char	Frequency
space	7
o	7
t	5
s	4
d	2
p	2
g	1
n	1
h	1
r	1
c	1
a	1



Problem 8

LCS		b	b	a	b	b	a	a	a	b
	0	0	0	0	0	0	0	0	0	0
b	0	1	1	1	1	1	1	1	1	1
a	0	1	1	2	2	2	2	2	2	2
b	0	1	2	2	3	3	3	3	3	3
b	0	1	2	2	3	4	4	4	4	4
a	0	1	2	3	3	4	5	5	5	5
b	0	1	2	3	4	4	5	5	5	6
a	0	1	2	3	4	4	5	6	6	6
b	0	1	2	3	4	5	5	6	6	7

The LCS between the strings is “babbaab”

Problem 9

a) To see implementation, check the linked git repo
b) To see implementation, check the linked git repo
c) For strings consisting of two unique characters, the recursive algorithm seems to slow when the strings lengths are around 28 to 32. This happens at length 13-15 when there are more than two unique characters.

The worst case time complexity of the recursive approach is O(2^n+m). This corresponds with my findings, as iterative rounds usually takes 4 times longer than the previous.

This can be explained by increasing X and Y's length with 1 per iteration/round, so in round 2 it will be 2^2+2 = 2^4, and in round 3 it will be 2^6.

$$\frac{2^6}{2^4} = 4$$

Here is a run with only "ab" in the string.

```
Recursive LCS:  19
DP LCS:  19
Time recursive: 1.078125
Time DP: 0.0
----- Finished Round 28 -----

Recursive LCS:  19
DP LCS:  19
Time recursive: 4.1875
Time DP: 0.0
----- Finished Round 29 -----

Recursive LCS:  19
DP LCS:  19
Time recursive: 16.703125
Time DP: 0.0
----- Finished Round 30 -----

Recursive LCS:  19
DP LCS:  19
Time recursive: 65.421875
Time DP: 0.0
----- Finished Round 31 -----
```

Here is a run with all chars in string

```
Recursive LCS:  3
DP LCS:  3
Time recursive: 5.90625
Time DP: 0.0
----- Finished Round 14 -----

Recursive LCS:  3
DP LCS:  3
Time recursive: 22.3125
Time DP: 0.0
----- Finished Round 15 -----

Recursive LCS:  3
DP LCS:  3
Time recursive: 84.859375
Time DP: 0.0
----- Finished Round 16 -----

Recursive LCS:  3
DP LCS:  3
Time recursive: 327.703125
Time DP: 0.0
----- Finished Round 17 -----
```