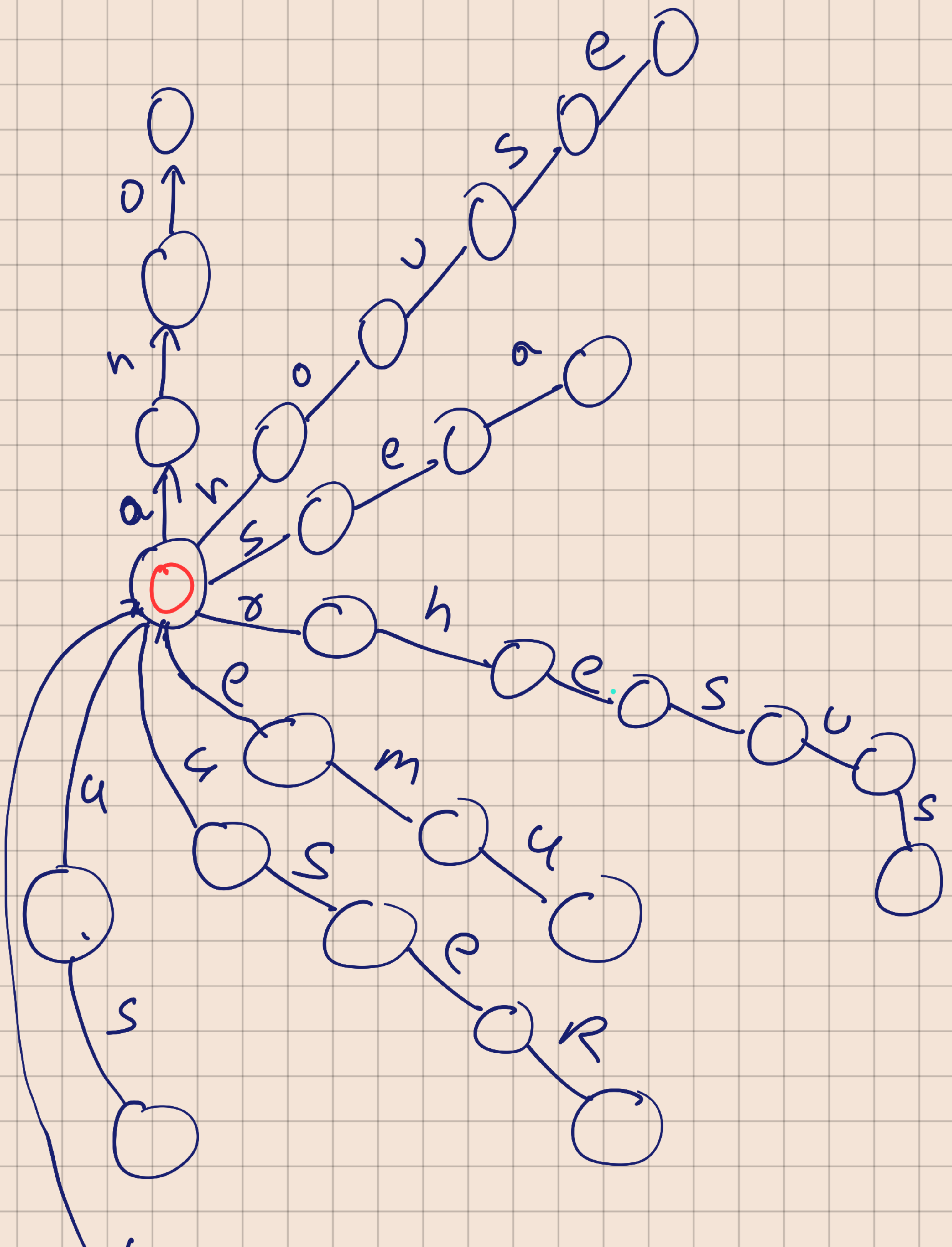
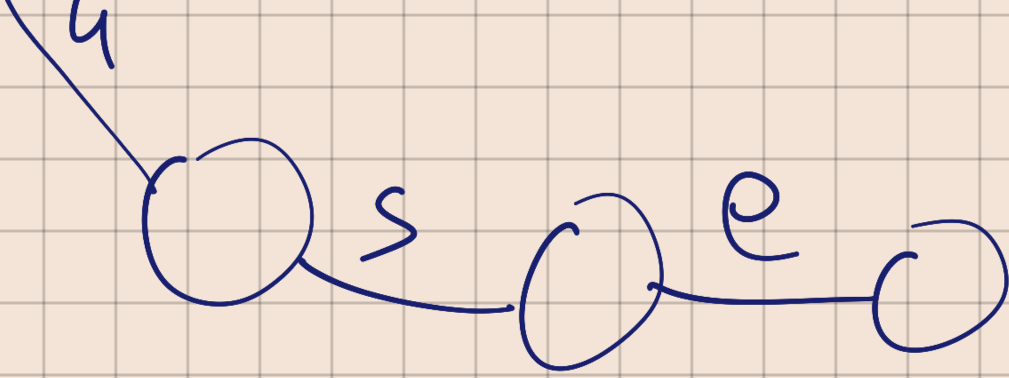
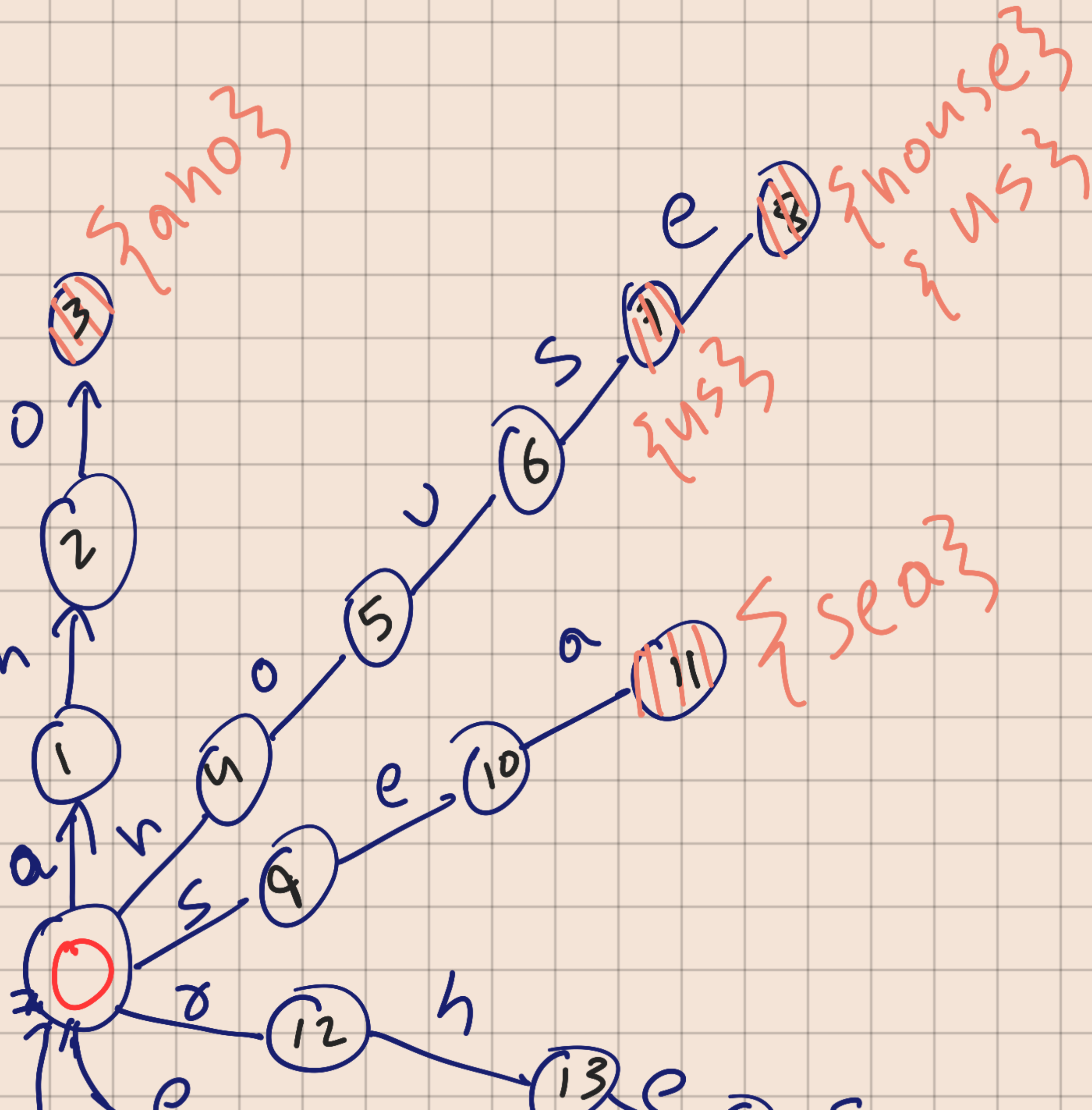
Name :- Shaique Solanki ; 7062850

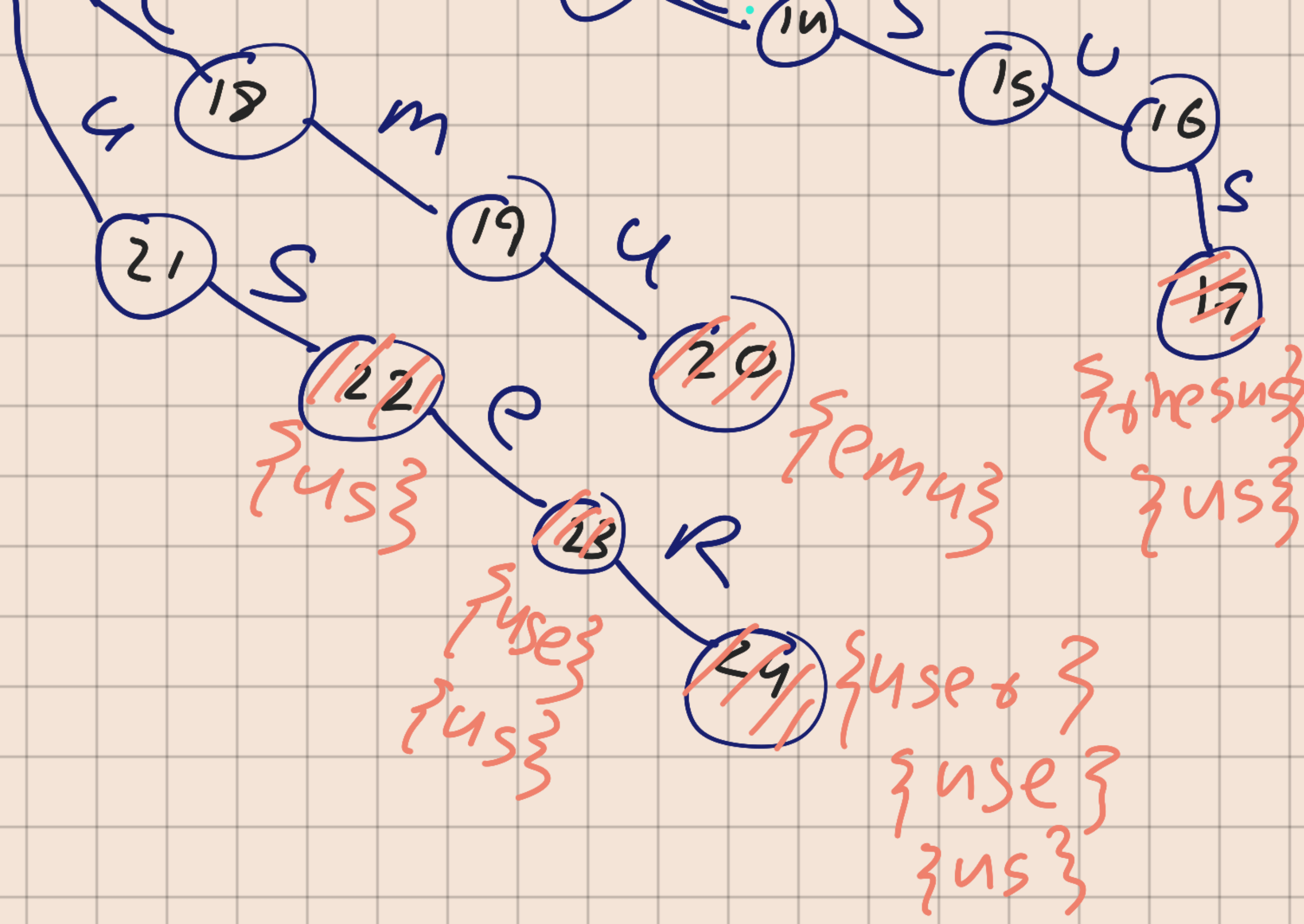




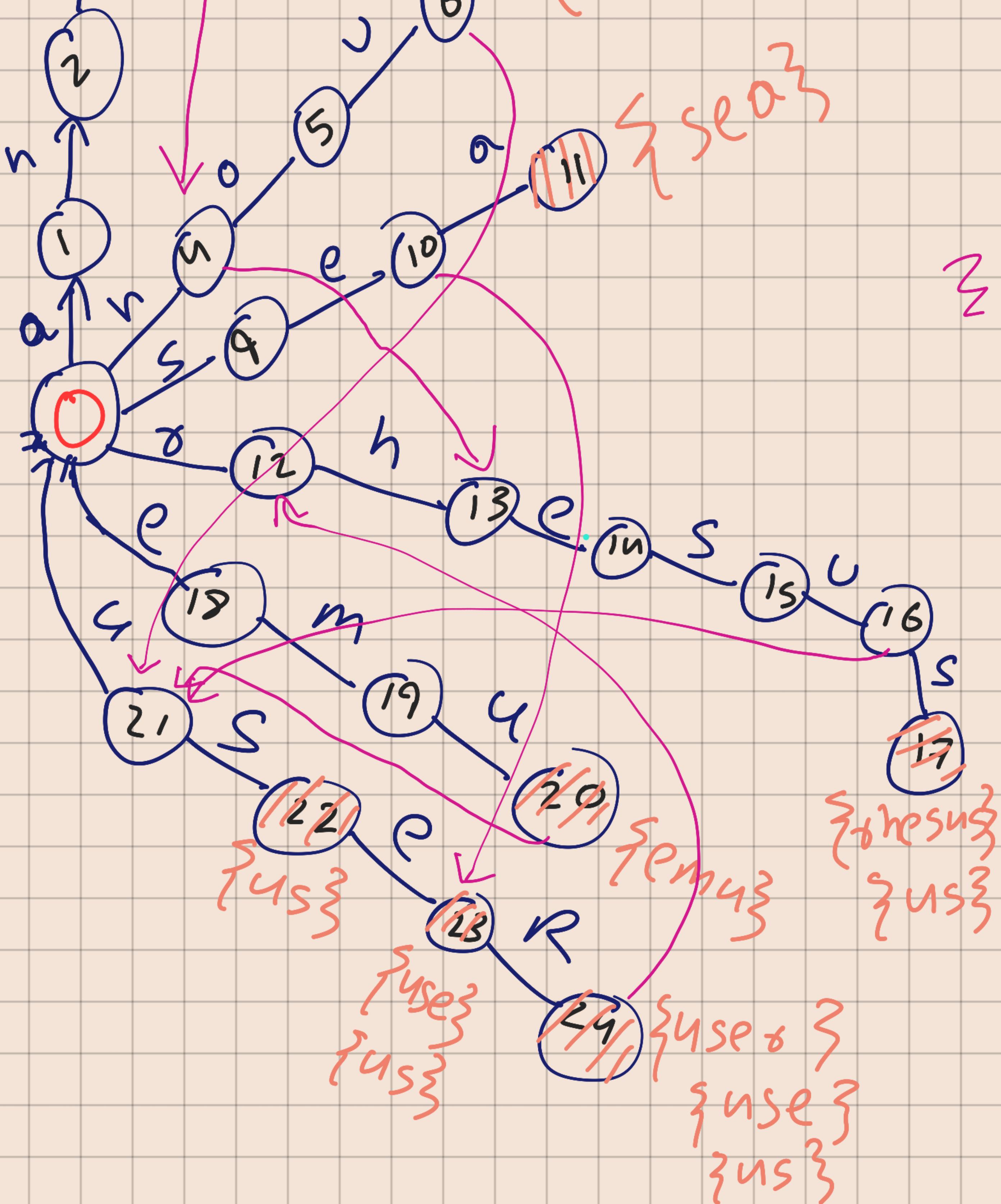
u
s
e



{ano}
{house}
{us}
{us}
{sea}
0
1
2
3
4
5
6
7
8
9
10
11
12
13
a
n
o
h
o
u
s
e
s
e
a
r
h
e



LPS function



LPS function

+

failure nodes marked

Q. 2.2

Recognising Cyclic Permutations.

Given: s & t are two strings of length = (n)

s is cyclic permt. of t if
$s == t[i:] + t[:i]$ for some i

which means, we cut 't' at any point, swap two substrings and we get 's'.

$\therefore$ t ICPO s

To find equivalence, we do following

1. Testing Reflectivity
2. Testing Symmetry
3. Testing Transitivity

Testing Reflectivity

To prove :- Every string is cyclic rotation of itself

Proof :

Let $i = 0$;

$\Rightarrow \quad t[0:n] = t \;;\; t[n:n] = \phi$
(empty

so if we cut t at 0;

string)

we get

$$t = t[0:n] + t[n:n] = t + \phi = t$$

$\therefore$ t ICPO t

Testing Symmetry.

To prove:- if s is rotation of t
then t is rotation of s

Proof:-

let $s = t[0:i] + t[i:n]$

or

$$s = t[i:n] + t[0:i]$$

"we cut t at position i and swap."

we can also cut s at position ~~n~~-i

The first [n-i] chars of 's'

are $t[i:n]$

The last 'i' characters are $t[0:i]$

$s[n-i : n] = t[0:i]$

$s[0 : n-i] = t[i:n]$

Adding both we get

$s = t[0:i] + t[i:n]$

$\therefore$ s (1CPO) t

<u>Testing</u> <u>Transitivity</u>:

Given 3 strings, r, s, t of length <u>'n'</u>

If s is rotation of r

If t is rotation of r

then, s is rotation of t

<u>**Proof:**</u>

$$s = t[i:n] + t[0:i]$$

$$t = r[j:n] + r[0:j]$$

Substituting t in s we get

$$s = (r[j:n] + r[0:j])[i:n] +$$
$$(r[j:n] + r[0:j])[0:i]$$

This effectively means we

are writing the string r at

$(j + i)$ positions.

let $k = j + i$

$(r[j:n] + r[0:j])[i:n] = r[k:n]$

$(r[j:n] + r[0:j])[0:i] = r[0:k]$

$\therefore\ s = r[k:n] + r[0:k]$

which proves s ICPO t

Example using "cat"

1. Reflectivity.

let $s =$ "cat"

cut at position 0

$s[0:3] + s[3:3] = \text{"cat"} + \text{""}$

"cat" ICPO "cat"

2. Symmetry.

→ $t = \text{"cat"}$

→ Rotate 't' by cutting at $i = 1$

$s = t[1:3] + t[0:1] = \text{"at"} + \text{"c"}$

$= atc$

$= \text{'s'}$

Now cut atc at $(n-i) = (n-1)$

$= 2^{nd}$ position

$\text{"atc"}[2:3] + \text{"atc"}[0:2]$

$= \text{"c"} + \text{"at"}$

$= cat.$

$\therefore$'s' ICPO 't'

Now, Algorithm to find whether

s ICPO t in $O(n)$ time.

From 01-3 lecture slide, we know that KMP algorithm has optimal Running time $O(m+n)$ (slide 19)

Now. for s ICPO t, we can say that s is a cyclic rotation of t iff s is in $t + t = t'$

Let $t =$ cat ; $t' =$ catcat

by observation cyclic rotation of t is atc which appears in

t' (cat cat)

Algorithm Steps

Step 1 while $len(t) \neq len(s)$

$t' = t + t$

This step takes $O(n)$ time

Step 2: while $len(t) \neq len(s)$

search for s in t' using KMP

LPS table creation takes $O(n)$ (refer slide 19 of 01-3)

And searching through KMP takes

$O(n+n) = O(2n) \approx O(n)$ time

Step 3 while $len(t) \neq len(s)$

If s is in t' by using kmp

we can prove that

s KMP t'