3-4 geek symbols

A C G T

T G C A

gene is a part of DNA which comes out as RNA – converted to protein

3 of the characters make one protein molecule (amino acid molecule).

4 × 4 × 4

64 possible

20 in nature (redundency) Many to one mapping

cell's behaviour – mediated by protein

set of amino acid molecules.

DNA is a string

RNA is also a 1D string of characters

So is protein. (3D structure, van Der Waal forces)

It gets folded.

(lowest energy state).

Its shape determines which reaction it will catalyse.

1000 – 5000 characters – 1 gene.

30,000 genes in a human.

10% of genome is useful, some genes don't come out as RNA.

Scanned by CamScanner

some changes in nucleotides.
dominant genes, recessive genes.

same molecule can be generated by different genes.
64 → 20.

Sequence one full DNA string - 2001.
sequence organisms eg: HIV virus
↓
changes itself every 3 days.

Biological Warfare
Gene editing

Given a big DNA - finding a gene in it.
Bigger string में chota string dhundna.

KMP algorithm starter

Scanned by CamScanner

Page No.:
Date:
YOUVA

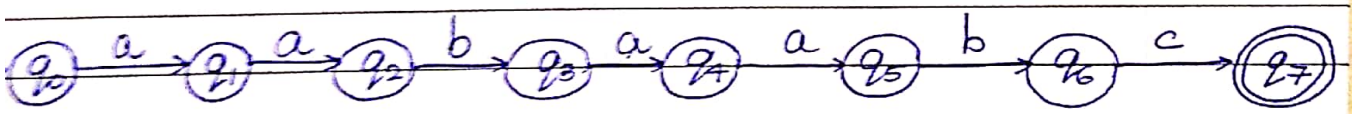
```
for (i=0 ; i<n ; i++)
{
    for (j=0 ; j< m ; j++)
    {
        if (a[i+j] != b[j])
            break ;
    }
    if (j == m)
        printf ("found string at %d", i) ;
}
```

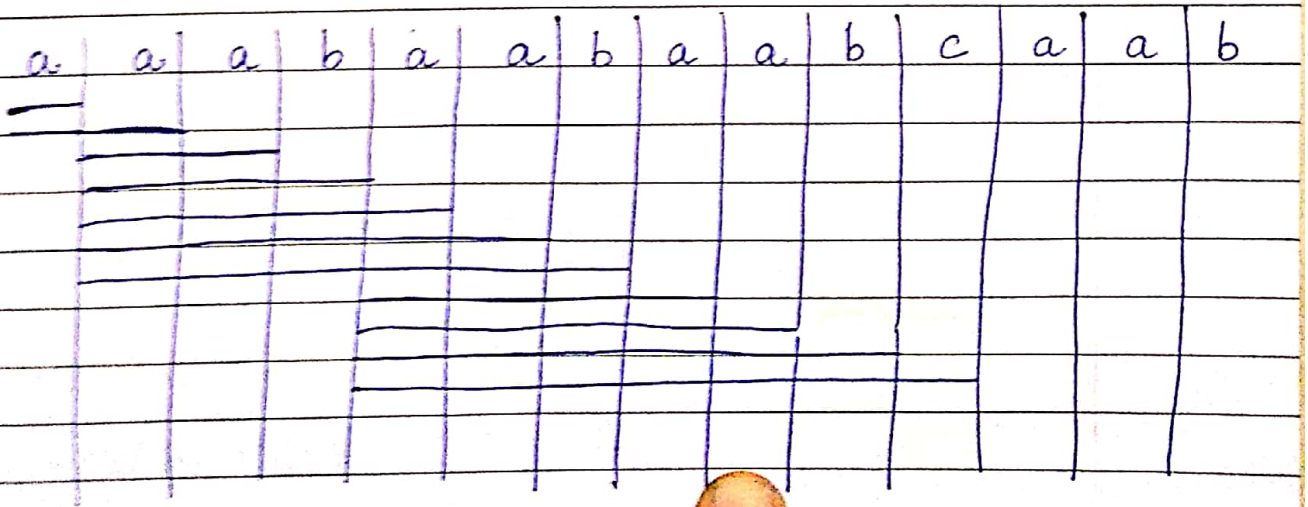
$O(nm)$

Construct a finite automata for it.

Str to search = " a a b a a b c "

str given = " a a a b a a b a a b c a a b "



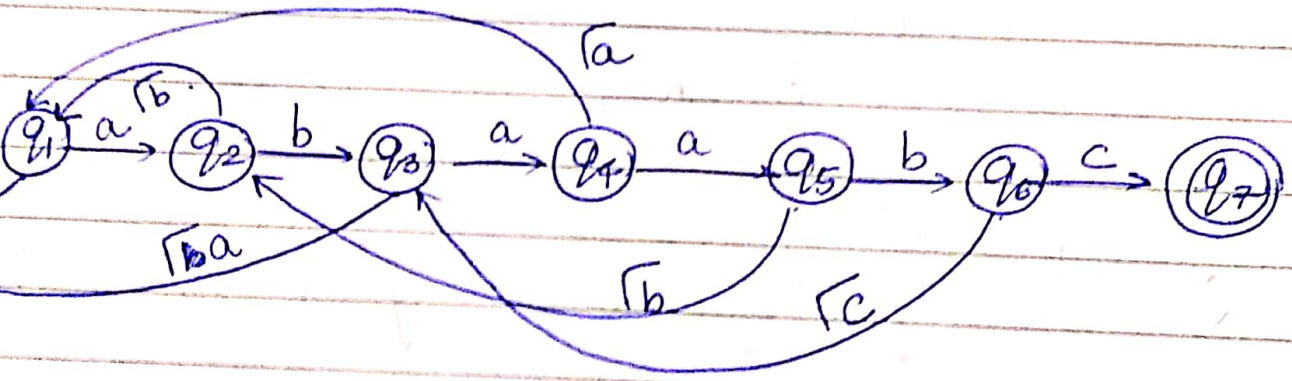


Scanned by CamScanner

If we fail at some stage, which state should we jump to ?

What is max^m initial part of string possible at the point of failure.

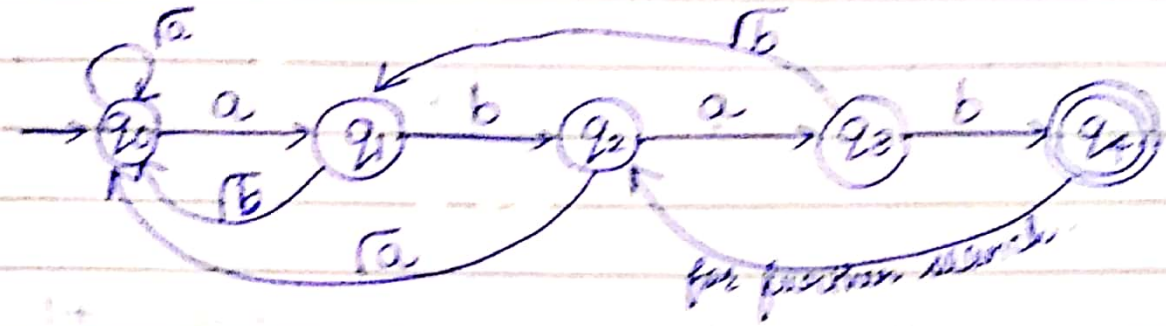
maintain failure table – failure lead to which state ?
maintain index to check last matched.



Building of failure table
KMP matching

Scanned by CamScanner

Pattern = "abab"
Text = "ababaaab"



1	2	3	4
a	b	a	b

1	2	3	4
0	0	1	2

failure states

For this algo, array starts from 1 and string from 1 to m.

Failure Table Algorithm
O/P = Array π

```
Compute_failure_table (string p)
{
    m = length (p)
    π[1] = 0
    k = 0
    for q = 2 to m
    {
        while (k>0 && p[k+1] ≠ p[q])
        {
```

Scanned by CamScanner

O(m) - Amortized Analysis

π

0	0	1	2	3	1
1	2	3	4	5	6

P

a	b	a	b	a	a
1	2	3	4	5	6

Text: "a b b a b a b a a"
"a b a c a b a b a a"

$\pi[j]$ - failure at j, go to $\pi[j]$

$\pi[j]$ - failure at $j+1$, start comparison of text string at next position with pattern at $\pi[j]$. ?

```
KMP_matcher ( string T, string P )
{
    n = length(T)
    m = length(P)
    π = compute_failure_table(p)
    q = 0
    for ( i = 1 to n)
    {
        while (q>0 && p[q+1] != T[i])
        {
            q = π[q]
        }
```

Scanned by CamScanner

07/08/19

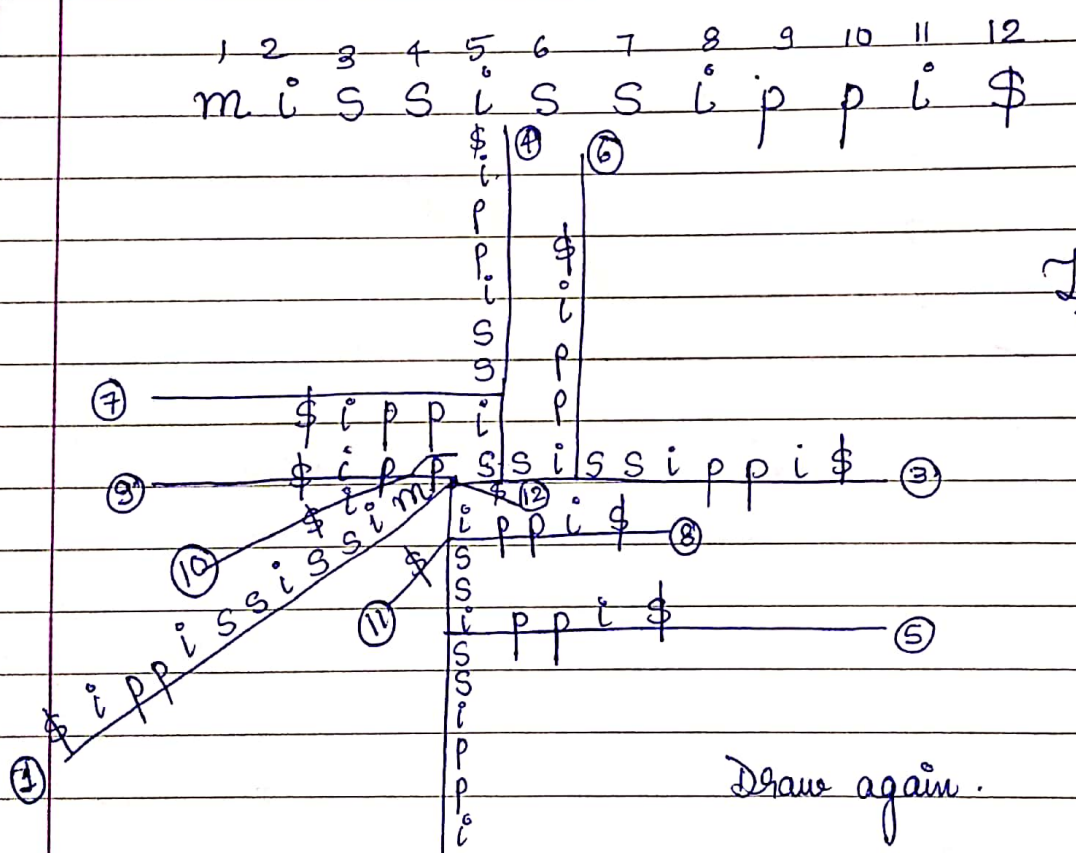
Even the DNA/bacteria — genes that they carry can affect our brain

Suffix Trees

Mississippi

suffixes – i, pi, ppi, ...

one marker is placed at end of string



Scanned by CamScanner

Labels → suffix array

12	11	2	5	2	1	10	8	7	4	6	3
i$	ippi$	issippi$	ississippi$	mississippi$	pi$	ppi$	sippi$	sissippi$	ssippi$	ssissippi$	$

(Alphabetical order of suffixes increasing)

arranging in lexicographical

suffix array is labels on top
naive write common elements between two adjacent strings

LCP array → common prefix
↓
longest common prefix

Scanned by CamScanner

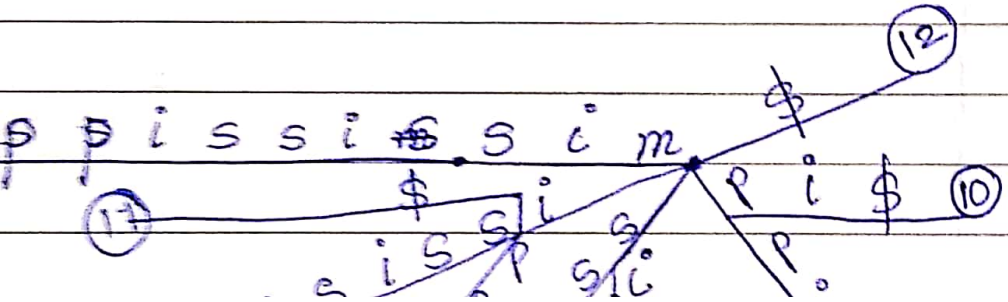
Again.

m i ss i ss i pp i $

suffixes

Suffix Tree

Suffix Trees.



Scanned by CamScanner

Page No.
Date
YOUVA

Burrows - Wheeler Transformation

Every substring of S is a prefix of some suffix of S.

check whether q is a substring of T.
→ follow path of q starting from root.
If q is exhausted, then q in T.

check whether q is suffix of T.
→ follow path of q starting from root.
if we end at leaf node, then q is a suffix of T.

Check # of occurrences of a character in T. (or q (string))
→ follow the path of q starting from root
→ number of leaves under the node reached after exhausting q is # of occurrences

longest repeat in T
→ deepest node that has atleast 2 leaf nodes

lexicographically first suffix.
→

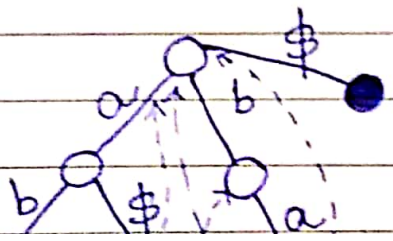
Scanned by CamScanner

suffix links
- reach a node
- remove first + letter on this path
- link to that suffix's node.

Eg) a a b linked to a b
b a a linked to a a.

Walk down the tree following q.
If you hit a dead end, save current depth and follow suffix link from the current node.
When you exhaust q. return the longest substring found.

a b a $.

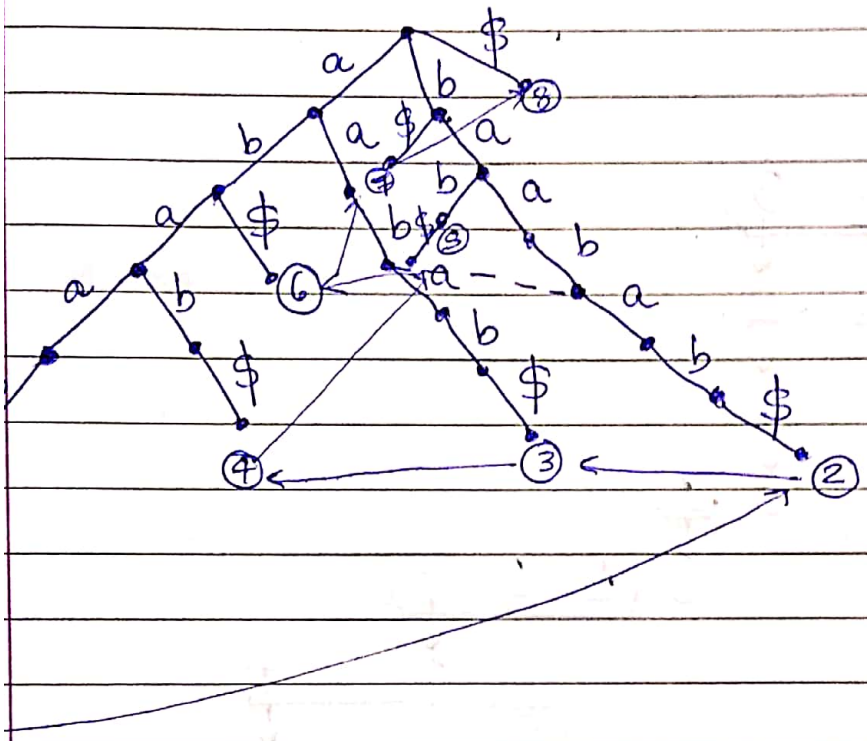


Ukkonen's Algorithm
↓
constructing suffix tree

Scanned by CamScanner

Page No.: Date: YOUVA

Suffix tree for abaabab$ — T



creates suffix links for every possible suffix

abaabab$
baabab$.
aabab$
abab$
bab$
ab$
b$
$.

baab. — take baab
remove first character
if we have remaining string in path, we connect baab to aab.

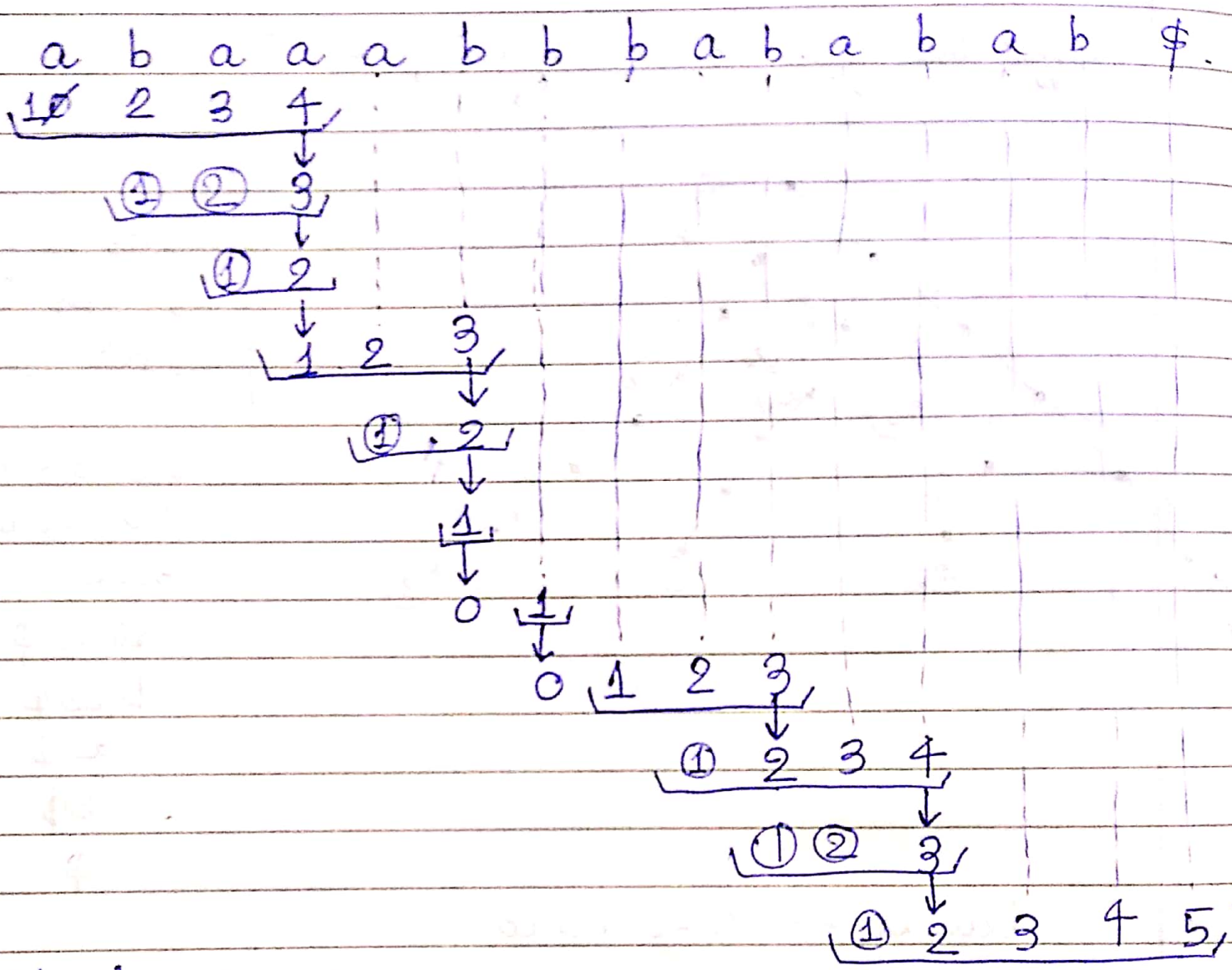
aab. ab ko aab.
ab b ko ab.

Connect every node to corresponding suffix link node.

abaaabbbababab$ — q

Find longest common substring of q and T.

Scanned by CamScanner



We have

a baa	baa
aab	aa.
b	ab
bab.	
abab	
abab$.	

Scanned by CamScanner

How many links ?

n characters in string

$$n + (n-1) + (n-2) + \ldots\ldots$$
$$= O(n^2)$$

Building the tree – $O(n^2)$.
Trie with links – $O(n^2)$
Just the trie – $O(n)$.

If string $1 \to n$ (build suffix tree)
string $2 \to m$

$\Rightarrow$ max^m depth we can go is m in the tree and come back

$\therefore O(2m)$

$\Rightarrow O(m)$.

Uses

How closely two gene strings are related.

Scanned by CamScanner

Page No.:
Date:

O(m) – Amortized Analysis

π

0	0	1	2	3	1
1	2	3	4	5	6

P

a	b	a	b	a	a
1	2	3	4	5	6

d

Text: "a b b a b a b a a"
"a b a c a b a b a a"

$\pi[j]$ – failure at j, go to $\pi[j]$

$\pi[j]$ – failure at j+1, start comparison of text string at next position with pattern at $\pi[j]$. ?

```
KMP_matcher (string T, string P)
{
    n = length(T)
    m = length(P)
    π = compute_failure_table(p)
    q = 0
    for (i = 1 to n)
    {
        while (q>0 && p[q+1] ≠ T[i])
        {
            q = π[q]
        }
        if (p[q+1] == T[i])
            q++ ;
        if (q==m)
        {
            printf (" found substr ") ;
            q = π[q]
        }
    }
}.
```

Scanned by CamScanner