1. Data Structure

Data Structure :-

The logical and mathematical model of a particular organisation of data is called Data structure.

Data item :-

Refer to a single unit of values.

Data item
- Group item
  - e.g Name – first, middle, Last
- Elementary item
  - e.g :- Student id. Adhar no.

- Feild, records, files

Feild - Single elementery unit of information representing an attribute of an entity.
e.g single practical property

Records :- collection of feild.
e.g collection of 10 practical

Files :- collection of Records.
collection of all practical

- Classfication of Data structure

data structure

- Primitive
  - int
  - Float
  - char
  - double
- Non-primitive
  - Linear
    - array → similar datal elements
    - linked list → nodes
    - Stack → LIFO
    - queue → FIFO
  - Non Linear
    - Tree
    - graph

- Algorithmic notation

1. Identifiying number
2. steps, control, Exit
3. comments & variable name.
4. Assigment statement (: =)

- Linear search.

e.g DATA = {22, 65, 1, 99, 32 17, 74, 49, 33, 2
Linear search (Data, item, N, K, loc
counter variable

- Steps to write Algorithm

1. [Intialize] set K:= 1 and LOC := 0
2. Repeat Step : ③ & ④
   while LOC := 0 and K ≤ N.
3. if item := DATA [K] then
   set loc := K.
4. Set K := K + 1
5. if loc := 0 then
6. write : item is not in the array DATA
7. else.
8. write : item is present at location loc.
9. Exit

- Largest element in array

DATA = { 22, 65, 1, 99, 32, 17, 74, 49, 33, 2 }

largest Element (DATA, N, K, loc, MAX)

step ① [Intialization] set MAX := DATA [1], k = 1 and loc := 1

② Repeat steps ③ & ④ while k ≤ N

control ③ if MAX < DATA [K] then :
Set loc := K and MAX := DATA [k]

④ Set k := k + 1

⑤ write loc, MAX.

exit ⑥ Exit

Page No.
Date : / /

• Asymptotic Notation

1] Omega Notations (Ω)

The omega Notation is used when function g(n) define lowerbound ~~of~~ for the function f(n).

$$f(n) = \Omega\, g(n)$$

If their exist a positive integer No n_0 and a positive number m search that $|f(n)| \geq M\, |g(n)|$ for all n $\boxed{n \geq n_0}$

2] Theta Notations (θ)

The theta Notation is used when the function f(n) is bounded both from above and below by the function g(n).

$$f(n) = \theta \cdot g(n)$$

If their exist two positive constant C_1 and C_2 and a positive integer n_0 Such that C_1 ~~an~~ $|g(n)| \leq C_2\, |g(n)|$ for all $n \geq n_0$

3] Oh Notation :-

$f(n) = Oh \cdot g(n)$

The oh Notation is used when the function ~~to~~ g(n) define upperbound for the function f(n).

Q. Consider the complexity function c(n) which measures the no. of time loc and max are updated in Step III

Determine the complexity of Algorithm

Q. a) Described and find c(n) for the worst case.

b) Described and find c(n) for the best case

c) Find c(n) for the average case when n = 3. assuming all arrangement of element in Data are equally likely

Ans:- a) ⇒ The worst case complexity occur when the elements are in increasing order where each comparision of max with DATA [K] forces value update complexity will be

loc and max to be updated. In this case the complexity $C(n) = n-1$

Ans:- b) $\Rightarrow$

The best case occur when largest element appear first. So when comparision of max with Data of k never forces loc and max to be updated accordingly the complexity $C(n) = 0$

Ans:- c) $\Rightarrow$

P	ABC	ACB	BAC	BCA	CAB	CBA
nP	0	0	1	1	1	2

A $\rightarrow$ L
B $\rightarrow$ N
C $\rightarrow$ S

$$C(n) = \frac{0+0+1+1+1+2}{6}$$

$$= \frac{5}{6}$$

- String processing / operations

A finite sequence s(o) or more character is called as string.
The number of charachters in a string is called it's length.
The string with zero character is called empty string or null string.

e.g i] THE END → length 7.
ii] TO BE OR NOT TO BE
→ length → 18.

- String operation

a] ~~Sub~~ SUBSTRING.

Accesing a SUBSTRING from a given string requiered 3 pieces of information

i] The name of the string or string itself
ii] The position of the first character of the SUBSTRING
iii] The length of SUBSTRING

- Syntax.

SUBSTRING (string, intial, length)

e.g.
i) SUBSTRING ("TO BE OR NOT TO BE", 4, 7)

⇒ BE□OR□N

ii) SUBSTRING ("THE END", 4, 4)

⇒ □END.

b) INDEXING

INDEXING is also called as pattern matching.
Refers to finding the position where a string pattern p first appear in a given string Text T.

- Syntax

INDEX (Text, Pattern)

e.g.
T ⇒ HIS FATHER IS THE PROFESSOR
INDEX (T, THE) ⇒ 7
INDEX (T, THEN) ⇒ 0
INDEX (T, '□THE□') ⇒ 14.

Page No.
Date : / /

c) Concatenation (merge)

Let S_1 and S_2 be string then Concatenation of S_1 and S_2 is indicated by.

Syntax :- $S_1 // S_2$

e.g ⇒ S_1 = THE & S_2 = END
$S_1 // S_2$ ⇒ THEEND
$S_1 // \square // S_2$ ⇒ THE END.

d) Length

(including Space)

The number of characters in a string is called it's length.

e.g T = THE END Space
LENGTH (T) = 7.

x. Let S and T be a character variable such that.

S = ' JOHN PAUL JONES'
T = ' A THING OF BEAUTY IS A JOY FOREVER'

Determine.
i) LENGTH (S) = 15

Page No.______
Date : / /

ii) LENGTH (T) = 34.
iii) SUBSTRING (S, 4, 8) = N□PAUL□J
iv) SUBSTRING (T, 10, 5) = F□BEA
v) INDEX (S, JO) = 1
vi) INDEX (S, 'JOY') = 0
vii) INDEX (S, '□JO') = 10
viii) INDEX (T, 'A') = 7
ix) INDEX (T, '□AD□') = 21
x) INDEX (T, 'THE') = 0
xi) SUBSTRING (S, 11, 5) // [ ] // SUBSTRING (S, 1, 5)
= JONES JOHN PAUL
xii) SUBSTRING (T, 28, 3) // GIVEN
= FORGIVEN

Ex. Let S and T be a character variable such that S = 'WETHE PEOPLE'
T = 'OF THE UNITED STATES'

Determine

i) LENGTH (S) = 13
ii) LENGTH (T) = 20
iii) SUBSTRING (S, 4, 8) = ~~HE□PEOPL~~ THE□PEOP
iv) SUBSTRING (T, 10, 5) = ITED□
v) INDEX (S, 'P') = ~~7~~ 8
vi) INDEX (S, 'E') = ~~5~~ 2
vii) INDEX (S, 'THE') = ~~3~~ 4

viii) INDEX (T, 'THE') = 4
ix) INDEX (T, 'THEN') = 0
x) INDEX (T, 'TE') = 11
xi) SUBSTRING (S, 4, 10) // □ ARE □ //
SUBSTRING (T, 8, 6)
= THE PEOPLE ARE UNITED.

⇒ Word / Text processing.

1] Insertion

Suppose the given text T. we want to insert a string S so that S begins in postion K.

Syntax :-

INSERT (Text, Position, String)

e.g ⇒
INSERT ('ABCDEFG', 3, 'XYZ')
Ans :- ABXYZCDEFG

INSERT ('ABCDEFG', 6, 'XYZ')
Ans :- ABCDEXYZFG

The INSERT function can be implemented by using string operation as follows :-

Syntax :

INSERT (T, K, S) ⇒ SUBSTRING (T, 1, K-1) // S // SUBSTRING (T, K, LENGTH (T) - K + 1)

e.g. I INSERT ($\overset{T}{\text{ABCDEFG}}$, $\overset{k}{3}$, $\overset{S}{\text{XYZ}}$)

~~INSERT (T, K, S) ⇒ SUBSTRING (T, 1, K-1)~~

~~INSERT~~
⇒ SUBSTRING ('ABCDEFG', ~~3~~1, 2) // xyz // SUBSTRING ('ABCDEFG' 3, 5)

⇒ ABXYZCDEFG

e.g II INSERT ($\overset{T}{\text{ABCDEFG}}$, $\overset{k}{6}$, $\overset{S}{\text{XYZ}}$)

⇒ SUBSTRING ('ABCDEFG', 1, 5) // XYZ // SUBSTRING ('ABCDEFG', 6, 2)

⇒ ABCDEXYZFG

Page No ______
Date : __/__/__

2) DELETION

Suppose in a given Text T we want to delete the substring which begins at position k and has length L

- Syntax :

DELETE (text, position, length)

e.g
DELETE (ABCDEFG, 4, 2)
Ans :- ABCFG

e.g DELETE (ABCDEFG, 2, 4)
Ans :- AFG

e.g DELETE (ABCDEFG, 0, 2)
Ans :- ABCDEFG

The DELETE Operation can be implemented using string operation as follows.

(I) Syntax :-
DELETE (T, K, L) ⇒ SUBSTRING (T, 1, K-1)
// SUBSTRING (T, K+L, LENGTH(T) - K - L + 1)

e.g :- DELETE ($\overset{T}{\text{ABCDEFG}}$, $\overset{K}{4}$, $\overset{L}{2}$)
⇒ SUBSTRING (ABCDEFG, 1, 3) //

SUBSTRING (ABCDEEFG, 6, 2)
=> ABCFG

e.g DELETE (ABCDEFG, 2, 4)

=> SUBSTRING (ABCDEFG, 1, 1) //
SUBSTRING (ABCDEFG, 6, 2)

=> AFG.

- Suppose Text T and Pattern P are given and we want to delete from T the first occurrence of the Pattern T.

Syntax :-
DELETE (T, INDEX (T, P), Length (P))

e.g T = ABCDEFG, P = CD . INDEX (T, P) = 3 and length (P) = 2

=> DELETE (ABCDEFG, INDEX (ABCDEFG, CD), 2)

=> DELETE (ABCDEFG, 3, 2)

=> ABEFG

It will deletes only first appearing pattern

e.g T = XABYAB2 , P = AB

⇒ DELETE (XABYAB2, INDEX (XABYAB2, AB) , 2)

⇒ DELETE (XABYAB2, 2 , 2)

⇒ XYAB2

- If we want to delete all apearence of pattern P from the text Then we have to use Algorithim which is as follows

i] Set K := INDEX (T, P)
ii] Repeat while K ≠ 0
 a] Set T = DELETE (T, INDEX (T, P), length (P))

 b] Set K = INDEX (T, P)

iii) Write := T
iv) Exit

3) ~~St~~ Replacement :-

Suppose in a given Text T. we want to replace first occurance of pattern P1 by a pattern P2 we denote this operation by

Syntax :-

REPLACE (Text, Pattern 1, Pattern2)

e.g ⇒

REPLACE ('XABYABZ', 'AB', 'C')
⇒ XCYABZ

e.g ⇒ REPLACE ('XABYABZ', 'BA', 'C')
⇒ XABYABZ

→ Using string operation replacement is explain as follows :-

k:= INDEX (T, P_1)
T; = DELETE (T, k, length (P_1))
INSERT (T, K, P_2)

Ans:- k:= INDEX (T, P_1) = INDEX ('XABYABZ', AB) ⇒ 2
T:= DELETE (T, K, length (P_1) ⇒ DELETE (XABYABZ, 2, 2)
⇒ XYABZ

INSERT (T, K, P_2) - INSERT ('X~~AB~~YABZ', 2, C)
⇒ XCYABZ

- In order
- To replace the every occurrence of Pattern 1 by Pattern 2, we have to use following algorithm

i) Set K := INDEX (T, P_1)
ii) Repeat while K ≠ 0
iii) a) set T := Replace (T, P, Q)
b) Set K := INDEX (T, P)
iii) Write : T
iv) Exit

e.g T = XAAABBBY, P = AB, Q = C
e.g T = XABYABZ, P AB, Q = C

Q1. INSERT ('AAAAA', 1, 'BBB') ⇒ BBBAA
Q1. INSERT ('AAAAA', 3 'BBB') ⇒ AABBB
Q1. INSERT ('KKKKK' 6, BBB') ⇒ KKKKK BBB

Q.2 Suppose T is the is the text "THE STUDENT IS Ill". Use insert operation to read T as follows
i) THE STUDENT IS VERY Ill
ii) THE STUDENT IS ILL TODAY
iii) THE STUDENT IS VERY Ill TODAY

Q.3

- DELETE ('AAABBB', 2, 2) = ABBB
- DELETE ('JOHN PAUL JONES', 6, 5)
  = JOHN JONES

Q.4

- REPLACE ('AAABBB', 'AA', 'BB') = BBABB
- REPLACE ('JOHN PAUL JONES', 'PAUL', 'DAVID')
  = JOHN DAVID JONES

Q.2

Ans:- i) INSERT ('THE STUDENT IS ILL', ~~16~~ 15, '□VERY')

Ans:- ii) INSERT ('THE STUDENT IS ILL', ~~20~~ 19, '□TODAY')

Ans:- iii) INSERT (INSERT (T, 15, □VERY), 24, □ TODAY)

OR

iii) INSERT (INSERT (T, 19, □ Today), 15, □VERY)

Q.5) i) DELETE ('AAABBB', 3, 3) = AAB

ii) DELETE ('AAABBB', 1, 4) = BB

iii) DELETE (' S, 31, 3)

Let, S = WE THE PEOPLE

Ans :- THE PEOPLE

iv)

Q.6) i) REPLACE ('ABABAB', B, 'BAB')
= ABABABAB

ii) REPLACE (S , 'WE', 'ALL')
Let, S = WE THE PEOPLE
= ALL THE PEOPLE

iii) REPLACE (T, 'THE', 'THESE')
Let, T =

Q.7) i) INSERT (AAA , 2, B)
= ABA

ii) INSERT ('THE BOY', 5, BIG)

• Pattern Maching

1. First pattern Matching / Slow pattern Matching.

e.g
T = COMPUTER
P = POT

MAX ⇒ Length(T) − Length(P) + 1
⇒ 8 − 3 + 1
⇒ 6

Fourth Work

S_1 ⇒ COM
S_2 ⇒ OMP
S_3 ⇒ MPO
S_4 ⇒ POT → 1+1+1−3
S_5 ⇒ OTE
S_6 ⇒ TER

C ⇒ 1 + 1 + 1 + 3
⇒ 6.

INDEX (T, P) = 4

Pattern is present at location 4.

e.g ii) T = $(ab)^5$ ⇒ ababababab
P = abc

MAX ⇒ Length(T) − Length(P) + 1
⇒ 10 − 3 + 1
⇒ 8

C ⇒ 3 + 1 + 3 + 1 + 3 + 1 + 3 + 1
⇒ 16

INDEX (T, P) = 0

Page No.
Date : / /

e.g (iii) $T = (CD)^{10} \Rightarrow CDCDCDCDCDCDCDCDCDC$
$P = AABA$

$MAX \Rightarrow Length(T) - Length(P) + 1$

$\Rightarrow 20 - 4 + 1$

$\Rightarrow 17$

Rough work

$S_1 = CDCD$
$S_2 = DCDC$
$S_3 = CDCD$
$S_4 =$

$C \Rightarrow 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1$
$1 + 1 + 1 + 1 + 1 + 1 + 1$
$+ 1 + 1 + 1$
$= 17(1)$
$= 17$

INDEX (T, P) = 0

e.g (iv) $T = (a)^{20}$
$P = aaab$

$MAX \Rightarrow Length(T) - Length(P) + 1$
$\Rightarrow 20 - 4 + 1$
$\Rightarrow 17$

$C \Rightarrow 17 \times 4$
$\Rightarrow 68$

INDEX (T, P) = 0

T = ababaaba
P = aaba

S_1 = abab
S_2 = baba
S_3 = abaa
S_4 = baab
S_5 = aaba

MAX = length(T) - length(P) + 1
= 8 - 4 + 1
= 5

C ⇒ 2 + 1 + 2 + 1 + 4
⇒ 10

INDEX (T, P) = 5

Note :-
Length (T) ⇒ S , Length (P) ⇒ R ,
Text (counter variable) ⇒ K .
Pattern (counter variable) ⇒ L .

If not found incr value of K

• Algorithm

If match found incr value of L

1] Set K := 1 and MAX := S - R + 1
2] Repeat step 3 to 5 while K ≤ MAX.
3] Repeat for L : 1 to R
 if P[L] ≠ T[K+L-1] then go to step 5.
4] Set INDEX = K and Exit. → (when match found)
5] Set K := K + 1 // 2, 3, 4, 5
6] Set INDEX = 0
7] Exit (when not match found)

e.g Consider the pattern $P = abc$ using Slow pattern matching algorithm find the no. of comparision C and INDEX of P ~~an~~ in each of the following text T.

i) $(a)^{10}$
ii) $(aba)^{10}$
iii) $(cbab)^{10}$
iv) $(D)^{10}$

$\Rightarrow$ i) $T = (a)^{10} \Rightarrow aaaaaaaaaa$
$P = abc$

MAX = length(T) - Length(P) + 1
= 10 - 3 + 1
= 8

C. $\Rightarrow 2 + 2 + 2 + 2 + 2$
$+ 2 + 2 + 2$
$\Rightarrow 16$

INDE(T, P) = 0

Pattern is not found in the text
or Pattern ~~is no~~ P = abc is not prent in Text $T = (a)^{10}$

Page No. ______

Date: __/__/__

ii) $T = (aba)^{10}$

$= abaabaabaabaabaabaabaaba$
$abaaba$

$P = abc$.

$MAX = \text{length}(T) - \text{Length}(P) + 1$

$= 30 - 3 + 1$

$= 28$

$C \Rightarrow (3 \times \overset{2}{28} + 1 + 2)$

$C \Rightarrow 84$

$\text{INDEX}(T, P) = 0$.

Patteen is not found in the text.

iii) $T = (cbab)^{10}$

$= cbabcbabcbabcbabcbabcbab$
$cbabcbabcbabcbab$.

$P = abc$.

$MAX = \text{Length}(T) - \text{Length}(P) + 1$

$= 40 - 3 + 1$

$= 38$

$C \Rightarrow 1 + 1 + 3$

$\Rightarrow 5$.

$\text{INDEX}(T, P) = 3$

iv) $T = (D)^{10}$ = DDDDDDDDDD
$P = abc$.

$MAX = LENGTH(T) - LENGTH(P) + 1$
$= 10 - 3 + 1$
$= 8$.

$C = 1 + 1 + 1 + 1 + 1 + 1 + 1$
$= 8$

$INDEX(T, P) = 0$

Patter is not found in the Text
$T (D)^{10}$

The complexity of first patteen matching algoeithim is $O(n2)$.

Turin machine
Q_0 Q_1 state
each state has two input

P = aab bb
Two values are result

Page No.____
Date: / /

2. Second pattern matching / Fast pattern matching.

e.g. $P = aaabb$

First list the intial segments of P.

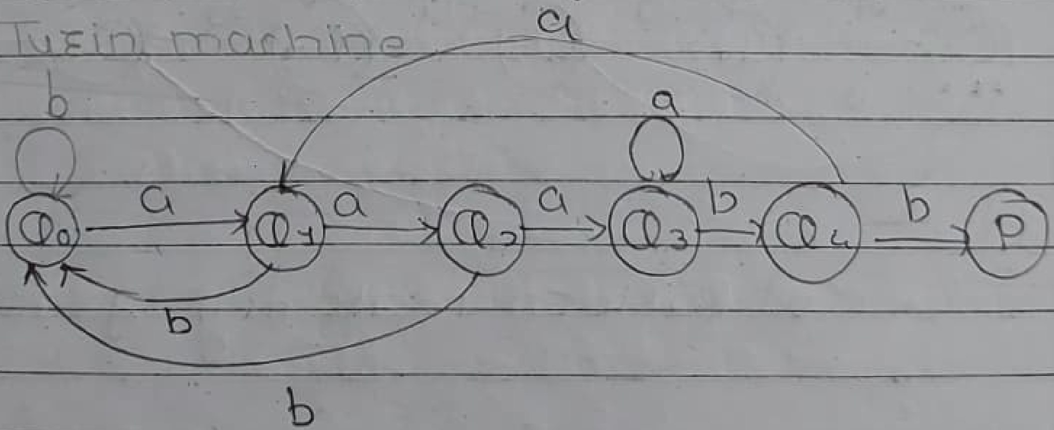
⇒ $Q_0 = \pi$, $Q_1 = a$, $Q_2 = aa$, $Q_3 = aaa$
$Q_4 = aaab$, $P = aaabb$.

(Rough work)

State	a	b	a	b
Q_0	Q_1	Q_0	πa	πb
Q_1	Q_2	Q_0	aa	ab
Q_2	Q_3	Q_0	aaa	aab
Q_3	Q_3	Q_4	$aaaa$	$aaab$
Q_4	Q_1	P	$aaaba$	$aaabb$

a) Pattern matching table

Turin machine



b) Pattern matching graph.

e.g. 2) P = aaaba

First list the 5 intial segments of P

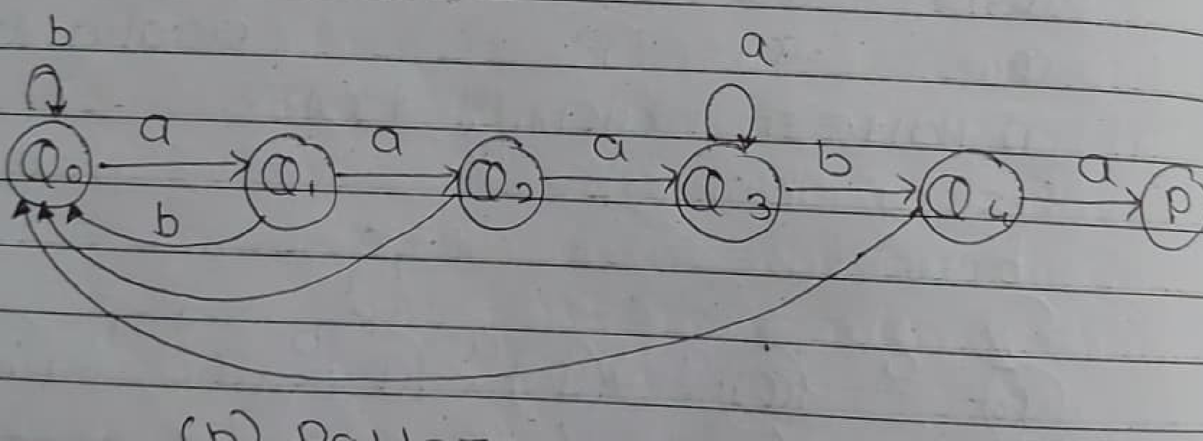
⇒ $Q_0 = \pi$, $Q_1 = a$, $Q_2 = aa$, $Q_3 = aaa$

$Q_4 = aaab$, $P = aaaba$

State	a	b	Rough work a	Rough work b
Q_0	Q_1	Q_0	πa	~~π~~b
Q_1	Q_2	Q_0	aa	~~a~~b
Q_2	Q_3	Q_0	aaa	~~a~~ab
Q_3	Q_3	Q_4	~~a~~aaa	aaab
Q_4	P	Q_0	aaaba	~~aa~~abb

(a) Pattern matching table

Turing machine



(b) Pattern matching graph

E.g 3] $P = aaba$

ababab

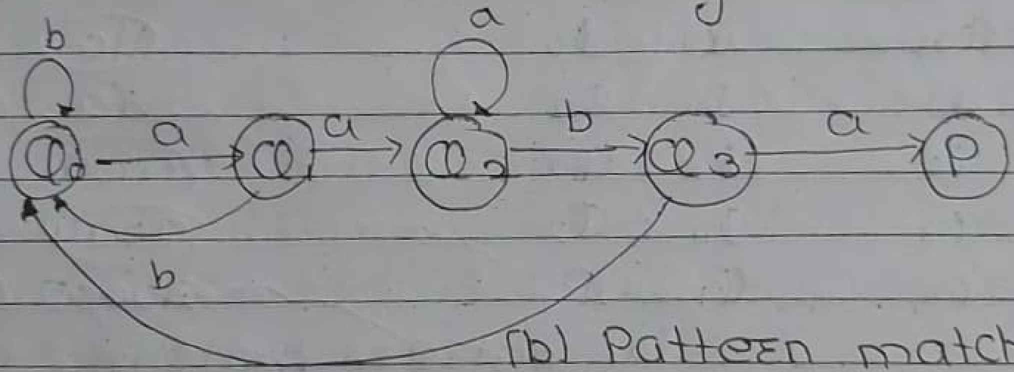
First list the intial segments of P.

$\Rightarrow Q_0 = \pi,\ Q_1 = a,\ Q_2 = aa\ \ Q_3 = aab$

~~Q_4~~ $= P = aaba$

State	a	b	a	b
Q_0	Q_1	Q_0	πa	$\cancel{\pi}b$
Q_1	Q_2	Q_0	aa	$\cancel{a}b$
Q_2	Q_2	Q_3	$\cancel{a}aa$	aab
Q_3	P	Q_0	$aaba$	$aabb$

(a) Pattern matching table



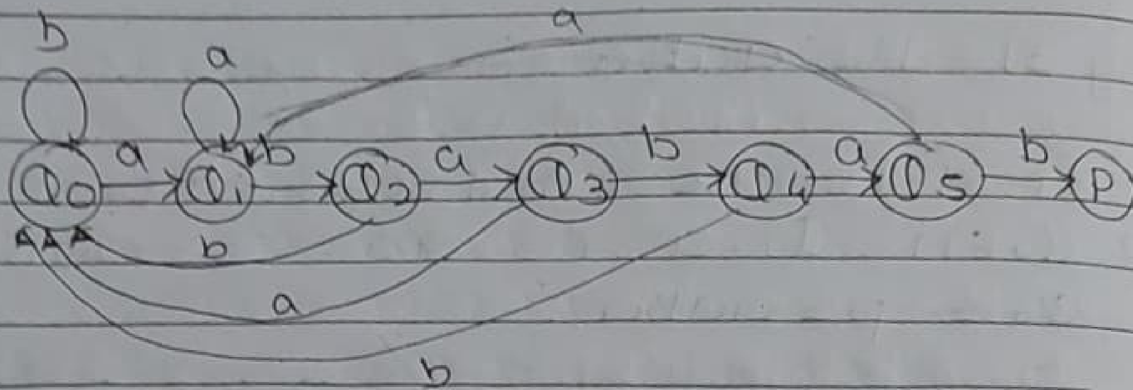
(b) Pattern matching graph

E.g 4] $P = ababab$

$\Rightarrow Q_0 = \pi,\ Q_1 = a,\ Q_2 = ab,\ Q_3 = aba,$

$Q_4 = abab,\ Q_5 = ababa,\ \ P = ababab$

State	a	b	a	b
Q_0	Q_1	Q_0	πa	πb
Q_1	Q_1	Q_2	aa	ab
Q_2	Q_3	Q_0	aba	abb
Q_3	Q_1	Q_4	$abaa$	$abab$
Q_4	Q_5	Q_0	$ababa$	$ababb$
Q_5	Q_1	[illegible]	$ababaa$	$ababab$



Ex. P = abaab.

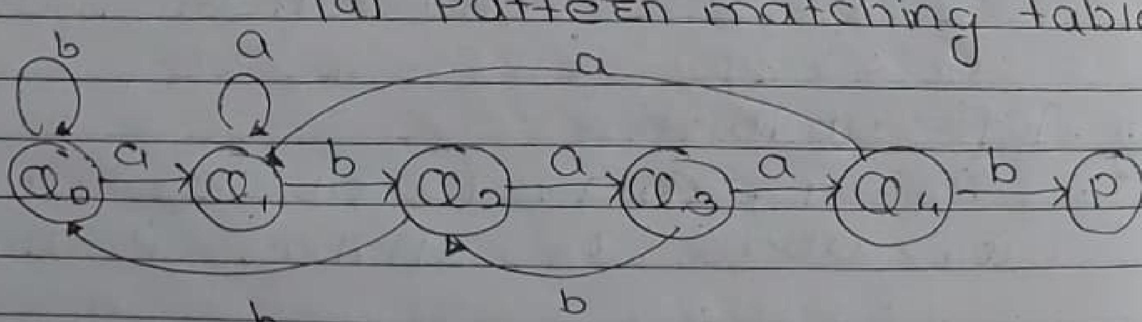
First list the intial segments of P

$Q_0 = \Lambda$, $Q_1 = a$, $Q_2 = ab$, $Q_3 = aba$

$Q_4 = abaa$, $P = abaab$.

State	a	b	a	b
Q_0	Q_1	Q_0	Λa	~~Λb~~
Q_1	Q_1	Q_2	~~a~~a	ab
Q_2	Q_3	Q_0	aba	~~abb~~
Q_3	Q_4	Q_2	abaa	~~ab~~ab
Q_4	Q_1	P	~~abaa~~a	abaab

(a) Pattern matching table



(b) Pattern matching graph

The complexity of this pattern matching algorithm is $O(n)$