UNIT → 5

☆ String-Matching :-
- Naive
- KMP
- Rabin Karp
- finite automata
- Boyer Moore

① Naive Algo :- It is the simplest method that works using Brute force approach (just do it approach). This algo performance checking of pattern in all position in the text b/w 0 to m-n where n is the text length & where m = Pattern length

```
NAIVE-ALGO (T[1---n], P[1---m])
for (s ← 0 to n-m)
{
  if((P[1---m]) == T[s+1 ---- s+m])
  {
    print ("Pattern find with shift", s)
  }
  else
  {
    print ("Not found")
  }
}
```

average) Max complexity can be O(n)
worst case O(~~m/n~~) (m×n)

Date : __/__/____ Page No:

$\pi[s]$ =

0	1	2	3	4	5
a	b	a	d	a	b
0	0	1	0	1	2

$p = \epsilon$	a	ab	aba	abad	ababa
$s = \epsilon$	b	ab	dab	adab	badab

2

Isme prefix & suffix nikalte time common dekhte hai to jiski length jyada aati hai vo digit likhte like above ex

find the compute PREFIX function for ababaca

Solve :-

$\pi[a]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0						

$\pi[1]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0					

$p - \epsilon$	a
$s - \epsilon$	b

$\pi[2]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0	1				

$p - \epsilon$	a	ab
$s - \epsilon$	a	ba

↓

Date : __/__/____ Page No.: ____

$\Pi[3]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0	1	2			

	∈	a	ab	aba
p -				
s -	∈	b	ab	bab

2

$\Pi[4]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0	1	2	3		

	∈	a	ab	aba	abab
p -					
s -	∈	a	ba	aba	baba

↓ 3

common long ~~digit~~ string

$\Pi[5]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0	1	2	3	0	

	∈	a	ab	aba	abab	ababa
p -						
∈ -	∈	c	ac	bac	abac	babac

$\Pi[6]$ =

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0	1	2	3	0	1

	∈	a	ab	aba	abab	ababac
p -						
s -	∈	a	ca	aca	baca	abaca
						babaca

Now the prefix Π is:-

0	1	2	3	4	5	6
a	b	a	b	a	c	a
0	0	1	2	3	0	1

Ans

Date : __/__/__ Page No.:

KMP-MATCHER (T, P)

1) n T.length
2) m P.length
3) Π(compute Prefix function (P))
4) q = 0
5) for i = 1 to n
6) while q > 0 & P[q+1] ≠ T[i]
7) q = Π[q]
8) if P[q+1] == T[i]
9) q = q+1
10) if q == m
11) Print "Pattern occur with shift i-m"
12) q = Π[q]

Que-1 T = badbabababadaab

Que-1 T = badbababababadaab length n=(15)
P = ababada m=(7)

find compute Prefix function -

Π(6) =

0	1	2	3	4	5	6
a	b	a	b	a	d	a
0	0	1	2	3	0	1

q - index value of Π
i - ——— T

T =

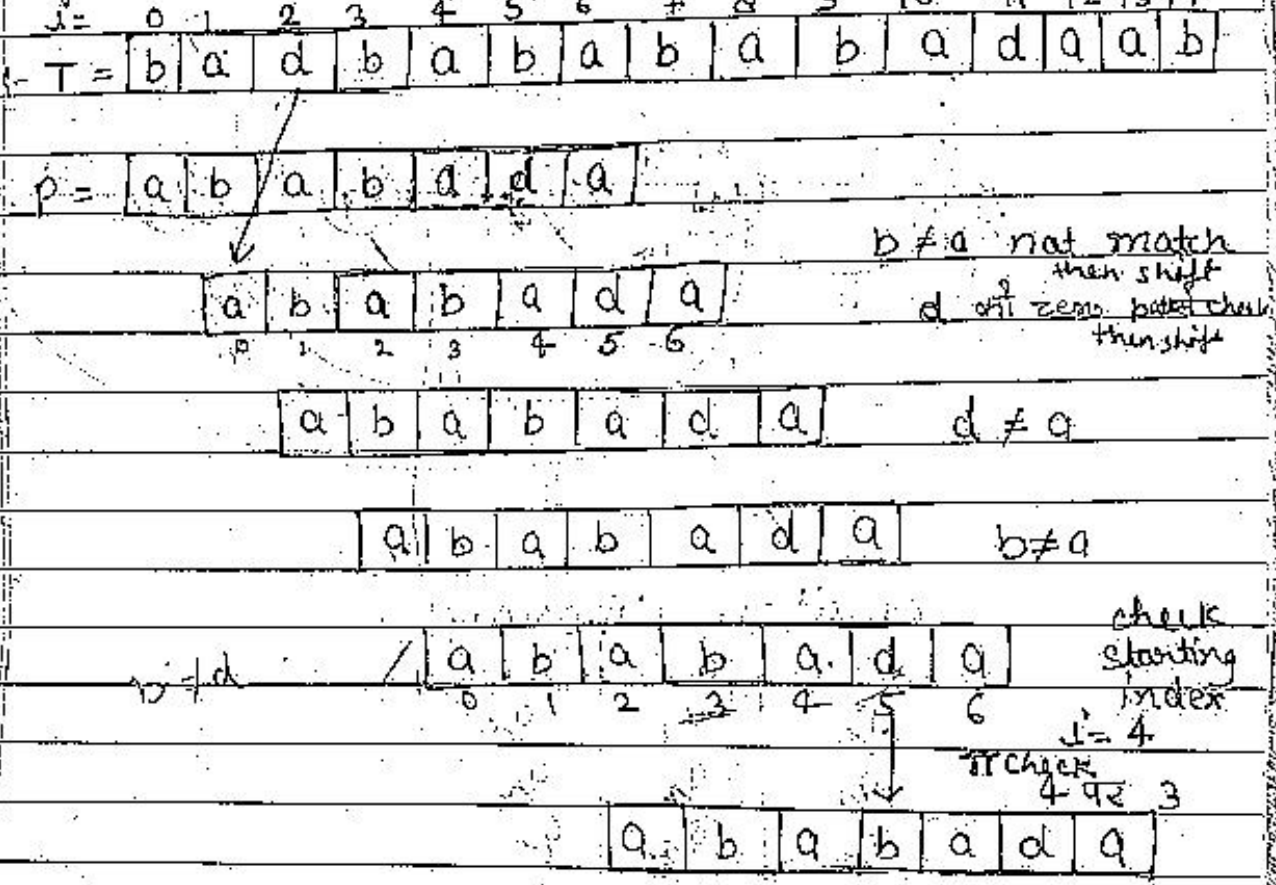
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	b	a	d	b	a	b	a	b	a	b	a	d	a	a	b

P =

a	b	a	b	a	d	a

Date : __/__/____ Page No.: ____

Agar Text, Pattern ke sath match nahi ho rha hai to i ki value increase kar de & q ki value bhi increase hoge



Ans

☆ String Matching Using finite Automation :-

finite automata is a five tuples tehen

Q - (Q is a set of finite state)

q_0 - (Initial state)

q_f - (final state)

Σ = (set of inputs)

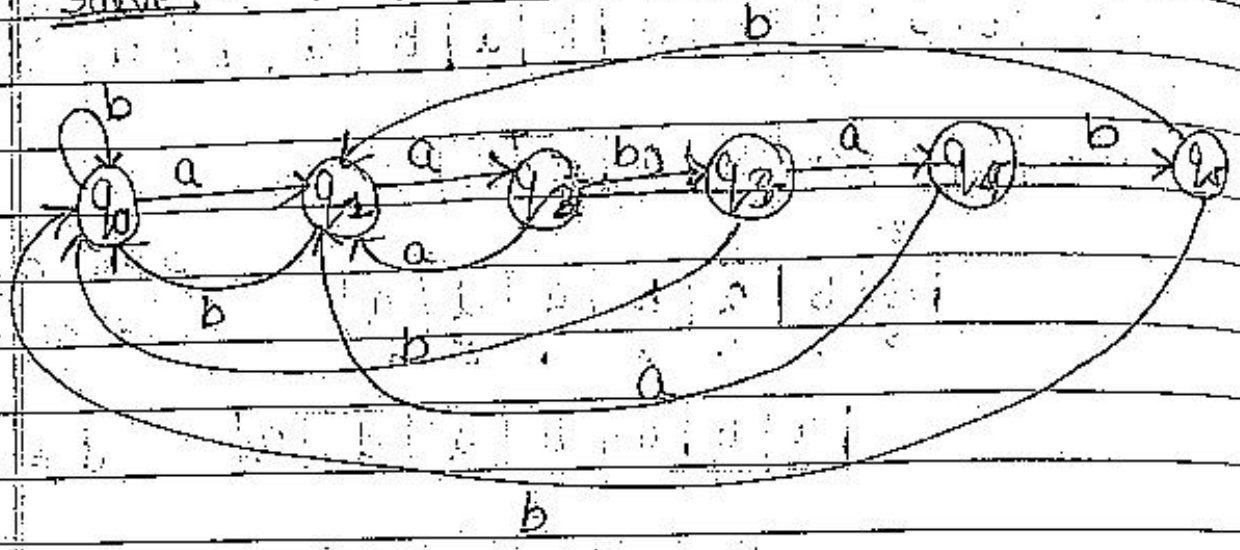
δ - (Transition function

$\delta \rightarrow Q \times \Sigma$

Date : __/__/__ Page No.:

Que-1 Construct a finite automata for matching the pattern for aababab and Text T = aaababaabaababaab.

Solve:-



Transition diagram

	a	b
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_3
q_3	q_4	q_0
q_4	q_2	q_5
q_5	q_1	q_0

String is accepted

★ String Matching Using Rabin Karp (T, P, d, q)

(1) n = T.length
(2) m = P.length
(3) $h = d^{m-1} \bmod q$
(4) p = 0
(5) $t_0 = 0$

(6) for i = 1 to m
(7) p = (dp + P[i]) mod q
(8) to = (dto + T[i]) mod q
(9) for s = 0 to n-m
(10) if p == ts
(11) if P[1 -- m] == T[(s+1) --- (s+m)]
(12) Print Pattern occurs with shift s
(13) if s < n-m
(14) $t_{s+1} = (d(t_s - T[s+1] \cdot h) + T[s+m+1]) \bmod q$

23/10/18 Que-1 T = 3141592653589793
P = 26
q = 11
Solve:- m = 2 (m = P.length)

formula: $t_{s+1} = (10(t_s - T[s+1]h) + T(s+m+1)) \bmod q$ (decimal no. system (base 10))

$t_{s+1} = (10(t_s - d_{10}^{m-1} \cdot$ old higher order digit) + (new low order digit))

Solve:-
find mod q of P
26 mod 11 = 4 (26/11, rem = 4)

find mod q of every two digits,
divide by 11

T = 31 41 59 26 53 58 97 93

9 3 8 4 4 4 4 10 9 2 3 1 9 2 5

means Period मतलब

check mode 4
26 ≠ 15, 26 ≠ 59, 26 ≠ 92, 26 = 26

Date : tenth/of 2 Text के 6 के बाद क्या आ रहा है Page No.:

Using formula = $10(26 - 10\times2) + 5$

$10(6) + 5 = 60 + 5$

$= 65 \mod 11 = 10$

↑ Term of 31

$T_{s+1} = 10(31 - 10\times3) + 4$ → ones of 14

$= 14 \mod 11 =$

$T_{s+1} = 10(14 - 10\times1) + 1$

$T_{s+1} = 41 \mod 11 =$

$T_{s+1} = 10(41 - 10\times4) + 5 =$

$T_{s+1} = 15 \mod 11 =$

$T_{s+1} = 10(15 - 10\times1) + 9$

$T_{s+1} = 59 \mod 11 =$

$T_{s+1} = 10(59 - 10\times5) + 2$

$= 92 \mod 11 =$

So on

T = 3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3

9 3 8 4 4 4 4 10 9 2 3 1 9 2 5

Ans

We need to match 26 at mod 4 then we see 26 is match at mod 4.

Date : __/__/____ Page No.: ____

Ques-1 $T = 2\,3\,5\,9\,0\,2\,3\,1\,4\,1\,5\,2\,6\,7\,3\,9\,9\,2\,1$

$P = 31415$

$Q = 13$

Solve : $m = 5$ $\quad m-1 = 4$

find mod $(31415 \bmod 13) = 7$

$T = 2\,3\,5\,9\,0\,2\,3\,4\,4\,1\,5\,2\,6\,7\,3\,9\,9\,2\,1$

mod 8

$T = 2\,3\,5\,9\,0\,2\,3\,1\,4\,1\,5\,2\,6\,7\,3\,9\,9\,2\,1$

$\text{mod } (13) = 8$

~~$t_{s+1} = (10(2359$~~

$t_{s+1} = (10(23590 - 10^{5-1} \cdot 2) + 2$

$= 10(23590 - 20000) + 2$

$= 235902 \bmod 13 = 8$

$t_{s+1} = 10(35902 - 10^4 \times 3) + 3)$

$t_{s+1} = 59023 \bmod 13 =$

$t_{s+1} = 10(59023 - 10^4 \times 5) + 1)$

$= 90231 \bmod 13 =$

$t_{s+1} = 10(90231 - 10^4 \times 9) + 4)$

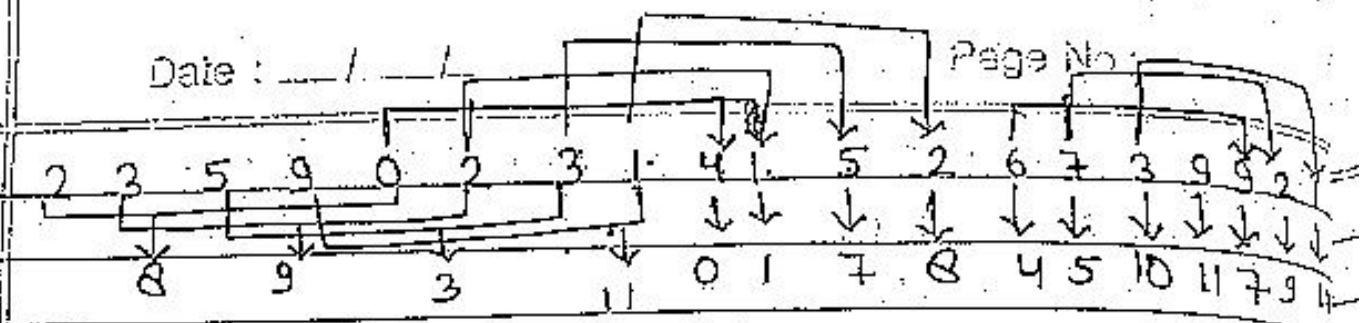
$= 2314 \bmod 13 =$

So on ...

$T = 2\,3\,5\,9\,0\,2\,3\,1\,4\,1\,5\,2\,6\,7\,3\,9\,9\,2\,1$

8

Date : __/__/__ Page No :



We need to match Pat mod 7 then we see 31415 is match at mod 7.

Ans

Imp ☆ NP Completeness :- All the algorithm that we have study till now, known as polynomial time algo, because they can be solved on I/P size(n) and order of (n^k) time, k is constant.
But all problem can't be solved in polynomial time. There is a interesting class of problem called non-polynomial (NP) complete problem whose status is unknown and which are no harder than graph searching & sorting algorithm.
NP time algo have yet been discovered for an NP complete problem.

→ Class P Problem :- which can be solve in polynomial time.

→ Class NP :- consist of those problem that are variable in polynomial time, it means that we are sure about the correctness of the soln & we can verify in polynomial time.

Date : __/__/____ Page No.: ____

Based on this way problem of P is also in NP.

→ NPC :- Any problem in NPC, if it is in NP and is as hard as many problem in NP. If any NPC problem can be solved in polynomial time Every NPC can be solved also in P time.
If some how we can establish a problem to be NP complete then it is better spend the time to develop a ~~approx~~ approximation algo.

★ Showing problem to be NP-complete :-

We are making the statement about "How hard it is" and not about "How easy it is". We are not trying to proof the existance of an efficient algorithm. But rather than no efficient algorithm is likely to exist. There are three key concept in showing a prob to be NP complete

(i) Decision (Feasible) Problem Vs optimization (Test) problem:

Optimization problem are those problem in which their exist many ~~physical~~ feasible solution each with and associated values and we wish to find ~~physical~~ feasible solution with the best value.

for eg:- shortest path Problem.

Decision problem

☆ Approximation algo:- If a problem is NP complete we are unlikely to find a polynomial time algo for solving it exactly.
An exponential time algo may be satisfactory if the actual inputs are small. Also it is possible to find a near optimal solution in polynomial time that is a good approach. An algo that produce a near optimal soln is called approximation algo.

Performance ratio for approximation algo:-

Suppose we are working on an optimization problem in which each potential solution has a positive post and we wish to find near optimal post. now depending on the problem the soln may be the one with minimum possible cost or max. possible cost. there may be called minimization or maximization cost.

An algo has an approximation ratio $P(n)$ if for any input of size n the cost C of the soln produce by the algo is with in the factor $P(n)$ of the post is

Date : __/__/__ c ? optimal

C^* of n optimal solution

then

$$\max\left(\frac{c}{c^*}, \frac{c^*}{c}\right) \leq P(n)$$

for maximization problem,

$0 < c < c^*$ and

ratio $\frac{c^*}{c}$ gives the factor by which the cost of an optimal soln is larger than the cost of approximate soln.

for minimization prob,

$0 < c^* < c$

ratio, $\frac{c}{c^*}$ gives the factor by which the cost of the approximate soln is larger than the cost of optimal soln. We alg also called an algo that achieve an approximation algo ratio from P(n) → approximation algo.

★ **TSP & Triangle Inequality** :- In this problem we must find a hamiltonian cycle of G with minimum cost. Let C(A) denotes the total cost of the edges in the subset $A \subseteq E$ and $C(A) = \sum_{(u,v) \in A} c(u,v)$.

Date : __/__/____ Page No.:

Triangle Inequility :- In many practical application it is always cheapest to go directory from one place u to another place w rather than via intermediary stop v. This means going u to w is cheapest than u to v or v to w. In another word we cut intermediate stop (not increasing the cost). This is known as triangle inequality.

$$c(u,w) \leq c(u,v) + c(v,w)$$

T = WELCOME TO SURANACOLLEGE

P = SURANA
0 1 2 3 4 5

Boyer Moore Algorithm

N = T = 22
M = P = 6

M Not here then see * = 6

6 place

SURANA

3 place Right's

... SURANA

BAD MATCH TABLE

P =

S	U	R	A	N	*
5	4	3	6	1	6

Length of Pattern

Value = Length − Index − 1

S → 6 − 0 − 1 = 5
U → 6 − 1 − 1 = 4
R → 6 − 2 − 1 = 3

A = last character
= * = Length of Pattern

P is Present at 4th Position

[BOYER MOORE ALGO]

T = WELCOME TO SURAM H COLLEGE

P = COLLEGE

when there is a mismatch after E we will only see Bad match table corresponding to E

shift 7 place

N = T = 22

M = P = 7

BAD MATCH TABLE

C	O	L	E	G	K
6	5	3	7	1	7

Value = Length - Index - 1

C O L L E G E

1 2 3 4 5 6 7

COLLEGE

COLLEGE

P is present at (15)