ment 3

0

0069 (F) starting point

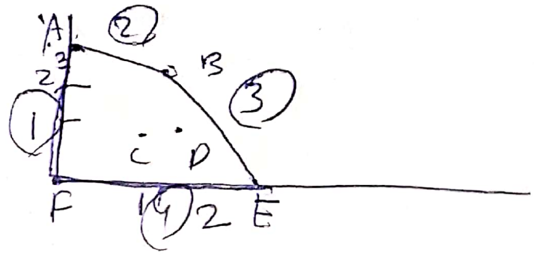
step	Current point p	Candidate q	Checkup (r)	Orientation p,q,r	Result	more cw?	new candidate
1	F(0,0)	A(0,3)	B(2,2)	-6	Cw	No	A(0,3)
2	F(0,0)	A(0,3)	C(1,1)	-3	Cw	no	A(0,3)
3	F(0,0)	A(0,3)	D(2,1)	-6	Cw	no	A(0,3)
4	F(0,0)	A(0,3)	E(3,0)	-9	Cw	no	A(0,3)
5	A(0,3)	B(2,2)	C(1,1)	-3	Cw	no	B(2,2)
6	A(0,3)	B(2,2)	D(2,1)	-2	Cw	no	B(2,2)
7	A(0,3)	B(2,2)	E(3,0)	-3	Cw	no	B(2,2)
8	A(0,3)	B(2,2)	F(0,0)	-6	Cw	no	B(2,2)
9	B(2,2)	C(1,1)	D(2,1)	1	Ccw	yes	D(2,1)
10	B(2,2)	D(2,1)	E(1,0)	-1	Cw	No	D(2,1)
11	B(2,2)	D(2,1)	C(1,1)	1	ccw	yes	E(3,0)
12	B(2,2)	E(3,0)	D(2,1)	-1	Cw	no	E(3,0)
13	B(2,2)	E(3,0)	F(2,1)	1	Cw	no	E(3,0)
14	B(2,2)	E(3,0)	F(0,0)	-6	Cw	no	F(3,0)
15	B(2,2)	E(3,0)	A(0,3)	-3	Cw	no	E(3,0)
16	E(3,0)	A(0,3)	B(2,2)	-3	Cw	no	A(0,3)
17	E(3,0)	A(0,3)	E(1,1)	3	ccw	yes	C(1,1)
18	E(3,0)	C(1,1)	D(2,1)	-1	cw	no	C(1,1)
19	E(3,0)	C(1,1)	F(0,0)	3	ccw	yes	F(0,0)
20	E(3,0)	F(0,0)	A(0,3)	-9	Cw	no	F(0,0)
21	E(3,0)	F(0,0)	B(2,2)	-6	Cw	no	F(0,0)
22	E(3,0)	F(0,0)	C(1,1)	-3	Cw	no	F(0,0)
23	E(3,0)	F(0,0)	D(2,1)	-3	cw	n	F(0,0)

23400GP

Hull points

F start 4 point | A iter 5 | B 9 | E 15 | F 24 pa iter

F → A → B → E → F



Q2

Pattern. P = "ababaca"

```
bf(P,m){
    f = array of size m
      f[0]=0
   for j from 1 to m-1:
       f[j]=0;

     for k from 1 to m-1:
          if P[0..k-1] = P[j-k+1 .. j]:
                 F[j]=k
                 return F
```

T.C

Outer loop → j   runs O(m)
inner loop → k   runs O(m)
Comop   Prefix length k suffix length O(k) time

$TC = O(m^2)$

2) Optimized KMP Algo.

```
optimized_kmp_Algo(P, m){
      F = array of size m
      F[0] = 0
      k = 0
   for j from 1 to m-1:
     while k>0 and && p[j] != P[k].
        k = F[k-1];
```

If P[j] = P[k]

k = k + 1

F[j] = k

return F

]

Time complex:

loop j → walk m times

TC : O(m)

Show computation

j	substring P[0..j]	prpu prefix	prpu suffixes	Matches (prefix=suffix)	Length [F[j]]
0	a	—	—	—	0
1	ab	a	b	—	0
2	aba	a, ba	ba, a	a	1
3	abab	a, ab, aba	bab, ab, b	ab	2
4	ababa	abab, a, ab, aba	baba, aba, ba, a	a, aba	3
5	ababac	ababa, a, ab, aba, ababa	babac, abac, bac, ac, c	—	0
6	ababaca	ababac, ababa, ab, baba, ab, a	bababaca, ba, baba, baca, aca, ca, a	a	1

j	0	1	2	3	4	5	6
P[j]	a	b	a	b	a	c	a

final failure function:

F = [0,0,1,2,3,0,1]

optimized KMP computation.

j	P[j]	k_before	Compar	Action	F[j]
0	a	–	–	k=0 initialize	0
1	b	0	P[0]=a≠b	no match	0
2	a	0	P[0]=a=a	k=1	1
3	b	1	P[1]=b=b	k=2	2
4	a	2	P[2]=a=a	k=3	3
5	c	3	P[3]=b≠c	no match	0

k=F[2]=1

↳ P[1]=b≠c

↳ k=F[0]=0

k=0

b	a	0	P[0]=a=a	k=1	1

~~final failure function~~

~~F=[0,0,1,2,3,0,1]~~

final failure function

F = [0,0,1,2,3,0,1]

both KMP and brute force have same failure function

4) Summarized

Criteria	Brute-force	KMP optim
TC	$O(m^2)$	$O(m)$
Compar	Repeated	for Each character ≤2
Range of Prev	Non	F[k-1] backtrack
Output	F=[0,0,1,2,3,0,1]	F[0,0,1,2,3,0,1]

Q3 Coinage problem

N = 13

S coins = {1, 5, 6, 8}

j (sum)(Change required)

(i) coin	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1+1 =2	1+1 =2	1+1 2	1+1 2	1+1 2	1+2 3	1+2 3	1+2 3	1+2 3
6	1	1	1	1	1	2	2+1 3	2+1 3	2+1 3	2+1 3	3+1 4	3+2 5	3+3 6	3+3 6
8	1	1	1	1	1	2	3	3	3+1 4	3+1 4	4+1 5	5+1 6	6+1 7	6+2 (8)

if change-req > coin{
copy above number}

Pseudo code

```
a[i][0]=1
for(i=0; i<=coins.length, i++){
    for(j=0 ; j<=change-requi; j++){
        if(coins[i]>j){
            a[i][j] = a[i-1][j];
        else{
            a[i][j] = a[i-1][j] + a[i][j-coin[i]]}
        }
```

upper, same (row)

Ans = 8

→ numbers to make

(5)

b) Levenshtein Distance

Find min no of operations

Str1 = "KITTEN"

Str2 = "SITTING"

K I T T E N

		0	1	2	3	4	5	6
S		1	1	2	3	4	5	6
I	2	2	k	2	3	4	5	
T	3	3	2	1	2	3	4	
T	4	4	3	2	1	2	3	
I	5	5	4	3	2	2	3	
N	6	6	5	4	3	3	2	
G	7	7	6	5	4	4	3	

operation

G N I T T I S

deleted kept replaced kept kept kept replaced

(1) K E (3)

(2)

Total no of operations 3

minimum

```
if[ str1[i] == str2[j]){
   diagonal Element  dp[i][j] = T[i-1][j-1];
   else { if value not same
   dp[i][j] = min(dp[i-1][j], T[i-1][j-1], T[i][j-1]);
           T[i][j]++;
   }
```

0, 1, 2

minim

j is calculate us his taraf back track

(Rod cutting problem)

→ Rod lenght = 8

class

price	length	0	1	2	3	4	5	6	7	8
1	1	0	1	2	3	4	5	6	7	8
5	2	0	1	5	6	10	11	15	16	20
8	3	0	1	5	8	10	13	16	18	21
9	4	0	1	5	8	10	13	16	18	21
10	5	0	1	5	8	10	13	16	18	21
16	6	0	1	5	8	10	13	16	18	21
18	7	0	1	5	8	10	13	16	18	21
20	8	0	1	5	8	10	13	16	18	21

21 ← maximum profit

m.

lenght used 2, 3, 3

5 + 8 + 8

= 21

logic

```
if (j >= i) {
    dp[i][j] = max( T[i-1][j], T[i][j-1] + profit[i] )
else
    // copy above value
    dp[i][j] = dp[i-1][j] };
```

d) Word break

len = 10

Input ilikeapple

0–9

	0	1	2	3	4	5	6	7	8	9
0	T	F	F	F	F	T	F	F	F	T
1		F	F	F	T	F	F	F	F	T
2		T	F	F	F	F	F	F	F	F
3			F	F	F	F	F	F	F	F
4				F	F	F	F	F	F	F
5						F	F	F	F	T
6							F	F	F	F
7								F	F	F
8									F	F
9										F

1st check single, then double, triple

& so on

= ilikeapple → at 0 if after comple table, boolean

= i like apple → at 1, 9 info

= apple → at 5, 9 info

i like → at 0, 4

& so on

ans → i like apple

Q4 Knapsack 0/1 Problem

SE Element	term
Course	item
Credits	value
Studyhours	weight
weeklystudylimit	knapsack capacity
Maxpossiblecredits	Maxprofit

Course	credits	studyhour
AI	3	2
ML	4	3
CV	6	1
oop	5	4

$n = 4$
$w = 8$

0 1 2 3 4 5 6 7 8

pi wi 0 0 0 0 0 0 0 0 0 0

2 3 1 0 0 0 2 2 2 2 2 2

3 4 2 0 0 0 2 3 3 3 5 5

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

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

Chose course oop and AI