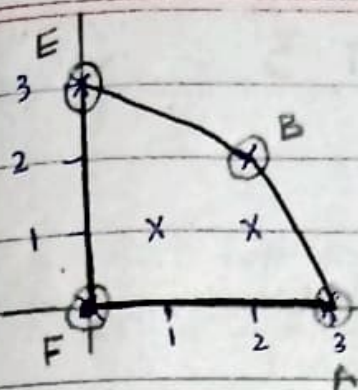


# DAA ASSIGNMENT 3

Q1:

Step	Current point P	Candidate (q)	checking point (r)	orientation (p, q, r)	More CCW?	More Candidate
1	F(0,0)	A(0,3)	B(2,2)	$(0-0)(2-0) - (3-0)(2-0) = 0 - 6 = -6$	No	-
2	F(0,0)	A(0,3)	C(1,1)	$(0-0)(1-0) - (3-0)(1-1) = 0 - 0 = 0$	No	-
3	F(0,0)	A(0,3)	D(2,1)	$(0-0)(1-0) - (3-0)(2-0) = 0 - 6 = -6$	No	-
4	F(0,0)	A(0,3)	E(3,0)	$(0-0)(0-0) - (3-0)(3-0) = 0 - 9 = -9$	No	-
5	A(0,3)	B(2,2)	C(1,1)	$(2-0)(1-3) - (2-3)(1-0) = -2 - (-3) = 1$	No	-
6	A(0,3)	B(2,2)	D(2,1)	$(2-0)(1-3) - (2-3)(2-0) = -2 - (-6) = 4$	No	-
7	A(0,3)	B(2,2)	E(3,0)	$(2-0)(0-3) - (2-3)(3-0) = -6 - (-9) = 3$	No	-
8	A(0,3)	B(2,2)	F(0,0)	$(2-0)(0-3) - (2-3)(0-0) = -6 - 0 = -6$	No	-
9	B(2,2)	C(1,1)	D(2,1)	$(1-2)(1-2) - (1-2)(2-2) = 1 - 0 = 1$	Yes	D
10	B(2,2)	D(2,1)	E(3,0)	$(2-2)(0-2) - (1-2)(3-2) = 0 - (-1) = 1$	Yes	E
11	B(2,2)	E(3,0)	F(0,0)	$(3-2)(0-2) - (0-2)(0-2) = -2 - 0 = -2$	No	-
12	B(2,2)	E(3,0)	A(0,3)	$(3-2)(3-2) - (0-2)(0-2) = 1 - 0 = 1$	No	-
13	B(2,2)	E(3,0)	C(1,1)	$(3-2)(1-2) - (0-2)(1-2) = -1 - 0 = -1$	No	-
14	E(3,0)	F(0,0)	A(0,3)	$(0-3)(3-0) - (0-0)(0-3) = -9 - 0 = -9$	No	-
15	E(3,0)	F(0,0)	B(2,2)	$(0-3)(2-0) - (0-0)(2-3) = -6 - 0 = -6$	No	-
16	E(3,0)	F(0,0)	C(1,1)	$(0-3)(1-0) - (0-0)(1-3) = -3 - 0 = -3$	No	-
17	E(3,0)	F(0,0)	D(2,1)	$(0-3)(1-0) - (0-0)(2-3) = -3 - 0 = -3$	No	-

F → A → B → E → F



$F \rightarrow A \rightarrow B \rightarrow E \rightarrow F$

Q2)

Brute force:

brute Force FF(P):

$m = \text{length}(P)$

F = array of size m

for  $j = 0$  to  $m-1$ :

$F[j] = 0$

for  $k = j$  down to 1:

match = True

for  $i = 0$  to  $k-1$ :

if  $p[i] \neq p[j-k+1+i]$ :

match = false

break

if match:

$F[j] = k$

break

return F



P = ababaca

j	Substring P[0..j]	longest proper prefix = suffix	F[j]
0	a	-	0
1	ab	-	0
2	aba	a	1
3	abab	ab	2
4	ababa	aba	3
5	ababac	-	0
6	ababaca	a	1

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

Time Complexity:

outer loop =  $O(m)$

middle loop =  $O(m)$  in worst case

inner loop =  $O(m)$  in worst case

Total:  $O(m^3)$  worst case

Optimized KMP:

Compute Failure KMP(P):

$m = \text{length}(P)$

F = array of size m

F[0] = 0

i = 0

For  $j=1$  to  $m-1$ :

while  $i > 0$  and  $P[i] \neq P[j]$ :

$i = F[i-1]$

if  $P[i] == P[j]$ :

$i = i + 1$

else:

$i = 0$

$F[j] = i$

return  $F$

We compute  $F[j]$  using previously computed  $F$  values.

We maintain a pointer  $i$  which is the length of the current longest prefix which is also a suffix for

$P[0 \dots j-1]$ . For next  $j$ , we try to extend by comparing  $P[i]$  with  $P[j]$ . If they match,  $F[j] = i + 1$ .

If not we set  $i = F[i-1]$  and repeat until match or  $i = 0$ .

Time Complexity:

The while loop runs at most  $O(m)$  total across all  $j$  because  $i$  increases at most by 1 per step, and decreases only via  $i = F[i-1]$  which is less than  $i$ .

So  $TC = O(m)$



j	P[j]	comparison	i <sub>before</sub>	i <sub>after</sub>	F[j]
0	-	-	0	-	0
1	b	P[0] != P[1]	0	0	0
2	a	P[0] == P[2]	0	1	1
3	b	P[1] == P[3]	1	2	2
4	a	P[2] == P[4]	2	3	3
5	c	P[3] != P[5]	3	1	
6		P[1] != P[5]	1	0	
		P[0] != P[5]	0	0	0
6	a	P[0] == P[5]	0	1	1

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

Comparison:

	Brute - Force	Optimized KMP
Time complexity	$O(m^3)$	$O(m)$
Character comparisons	Upto $O(m^3)$	$O(m)$
Reuse of computed info	No reuse	Reuses F values via backtracking
Practical efficiency	Slow for long patterns	Fast, used in real KMP matcher

Q3)

(a)

$S = \{1, 5, 6, 8\}$  desired change is 13

$dp[i] \rightarrow$  no. of ways to make amount  $i$

$dp[0] = 1 \rightarrow$  base case (one way: use no coins)

For each coin  $c$  update:

$dp[i] += dp[i - c]$  for  $i = c$  to amount

coins \ w	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	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

Final  $dp[13] = 8$

There are 8 combinations to make amount 13 with coins

$\{1, 5, 6, 8\}$

(b)

$str1 = KITTEN$

$str2 = SITTING$

$dp[i][j] \rightarrow$  min operations to convert first  $i$  chars of  $str1$  to first  $j$  chars of  $str2$

$dp[i][0] = i \rightarrow$  delete all  $i$  chars  
 $dp[0][j] = j \rightarrow$  (insert  $j$  chars)

} Base cases



if  $str1[i-1] == str2[j-1]$ :

$$dp[i][j] = dp[i-1][j-1]$$

else:

$$dp[i][j] = 1 + \min(dp[i-1][j], dp[i][j-1], dp[i-1][j-1])$$

(delete, insert, substitute)

	O	S	I	T	T	I	N	G
O	0	1	2	3	4	5	6	7
K	1	1	2	3	4	5	6	7
I	2	2	1	2	3	4	5	6
T	3	3	2	1	2	3	4	5
T	4	4	3	2	1	2	3	4
E	5	5	4	3	2	3	3	4
N	6	6	5	4	3	3	2	3

Final Edit Distance = 3

Edit distance = 3

Operations:

- 1)  $K \rightarrow S$  (substitute)
- 2)  $E \rightarrow I$  (substitute)
- 3) Insert G at end