CSC 226 Assignment 4 Xiaoqing Zhang

1. x = skullandbones y = lullabybabies

llcs[,]		s	k	u	l	l	a	n	d	b	o	n	e	s
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
l	0	←0	←0	←0	↖1	↖1	←1	←1	←1	←1	←1	←1	←1	←1
u	0	←0	←0	↖①	←1	←1	←1	←1	←1	←1	←1	←1	←1	←1
l	0	←0	←0	↑1	↖②	↖2	←2	←2	←2	←2	←2	←2	←2	←2
l	0	←0	←0	↑1	↖2	↖③	←3	←3	←3	←3	←3	←3	←3	←3
a	0	←0	←0	↑1	↑2	↑3	↖4	←4	←4	←4	←4	←4	←4	←4
b	0	←0	←0	↑1	↑2	↑3	↑4	←4	←4	↖5	←5	←5	←5	←5
y	0	←0	←0	↑1	↑2	↑3	↑4	←4	←4	↑5	←5	←5	←5	←5
b	0	←0	←0	↑1	↑2	↑3	↑4	←4	←4	↖5	←5	←5	←5	←5
a	0	←0	←0	↑1	↑2	↑3	↖④	←4	←4	↑5	←5	←5	←5	←5
b	0	←0	←0	↑1	↑2	↑3	↑4	←4	←4	↖⑤	←5	←5	←5	←5
i	0	←0	←0	↑1	↑2	↑3	↑4	←4	←4	↑5	←5	←5	←5	←5
e	0	←0	←0	↑1	↑2	↑3	↑4	←4	←4	↑5	←5	←5	↖⑥	←6
s	0	↖1	←1	←1	↑2	↑3	↑4	←4	←4	↑5	←5	←5	↑6	↖⑦
				u	l	l	a			b			e	s

∴ the longest common subsequence of the two strings is:
u l l a b e s

2. ① Create a character array to store the lcs string lcs[

② Start from the right most bottom corner and one by one check and store character in lcs[]

③ If current character in X[] and Y[] are same, then current character is part of LCS

④ If not same, then find the larger of L[i-1][j] and L[i][j-1] and go in the direction of larger value.

④ print lcs[].

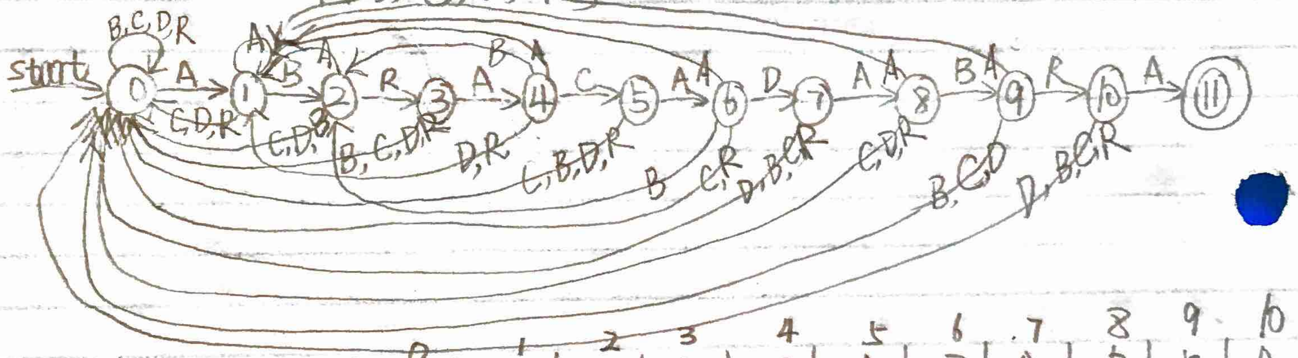
3. ⓐ ABRACADABRA = pattern

Σ = {A, B, C, D, R}

C		A	B	R	A	C	A	D	A	B	R	A	right [c]
		0	1	2	3	4	5	6	7	8	9	10	
A	-1	0			3		5		7			10	10
B	-1		1							8			8
C	-1					4							4
D	-1							6					6
R	-1			2							9		9

ⓑ pattern ABRACADABRA

Σ = {A, B, C, D, R}



dfa [] []	A	B	R	A	C	A	D	A	B	R	A
	0	1	2	3	4	5	6	7	8	9	10
A	1	1	1	4	1	6	1	8	1	1	11
B	0	2	0	0	2	0	2	0	9	0	0
C	0	0	0	0	5	0	0	0	0	0	0
D	0	0	0	0	0	0	7	0	0	0	0
R	0	0	3	0	0	0	0	0	0	10	0

```
4. import java.util.*;
   class Blank {
        public static void main(String args[]) {
             String txt;
             int M;
             Scanner s = new Scanner(System.in);
             System.out.println("enter string and value for M
             txt= s.nextLine();
              M = s.nextInt();
             int length = txt.length();
             int j = 0;
             int i = 0;
             while (i<length){
                 if (txt.charAt(i) = = ' '){
                      i = j;
                      while (j<length){
                           if (txt.charAt(j)==' '){
                                  j=j+1;
                           }else{
                                  break;
                           }
                      }
                      if (j-i>=M){
                           System.out.println("string found
                                        at" + i);
                           break;
                      }
                      i=j;
                 }else{
                      i=i+1;
                 }
             }
             if (i==1){
                 System.out.println("string not found and
                                   length =" + I);
             }
   }    }
```

Analysis Running time:

charAt()→constant time

# of iteration → n

∴ it's O(n)

5. T = text
p = Pattern
d = domain of letters of characters
q = prime number
k = k^{th} character of Pattern

Algorithm:

```
n = T.length
m = p.length
h = $d^{m-1}$ mod q
p = 0
$t_0$ = 0
for i = 1 to m
    if(i != k)
        p = (dp + p[i]) % q
        $t_0$ = (d$t_0$ + T[i]) % q
for s=0 to n-m
    if p = $t_s$
        for j=1 to m
            if j != k && T[s+j] != p[j]
                break;
        if j == m
            print "Pattern occurs with s"
    if s < n-m
        $t_{s+1}$ = (d($t_s$ - T[s+1]h) + T(s+m+1)) mod q
    $t_{s+1}$ = [d($t_s$ - T[s+k+1] * h) + T[s+k]] mod q
```