Longest Common Subsequence (LCS)

- Y= “ABCE”
- X=“ABEC”

- Longest common substring=AB
- Longest Common subsequence = ABC or ABE

Y =	A	B	E	J

X =	A	J	B	C	E

Y, j

X, i

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

Cost Matrix (c)

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

Direction Matrix (b)

1 = Diagonal (NW) [if X_i=Y_j]

2 = Up (N)[if upper value is greater or equal to left value]

3 = Left (W) [if left value is greater than upper value]

```
LCS-Length(X, Y)

    m <- length[X]
    n <- length[Y]

    for i <- 1 to m
        c[i,0] <- 0
    for j <- 1 to n
        c[0,j] <- 0

    for i <- 1 to m
        for j <- 1 to n
            if (x_i == y_j) {
               c[i,j] <- c[i-1,j-1] + 1
               b[i,j] <- NW
               }
            else if (c[i-1,j] >= c[i,j-1]) {
               c[i,j] <- c[i-1,j]
               b[i,j] <- N
               }
            else {
               c[i,j] <- c[i,j-1]
               b[i,j] <- W
               }
```

Y =	A	B	E	J

X =	A	J	B	C	E

X, i \ Y	0	1A	2 B	3E	4J
0	0	0	0	0	0
1A	0	1	1	1	1
2J	0	1	1	1	2
3B	0	1	2	2	2
4C	0	1	2	2	2
5E	0	1	2	3	3

Cost Matrix (c)

Y, j

X, i \ Y	0	1	2	3	4
0	0	0	0	0	0
1	0	NW	W	W	W
2	0	N	N	N	NW
3	0	N	NW	W	N
4	0	N	N	N	N
5	0	N	N	NW	W

Direction Matrix (b)

1 = Diagonal (NW), **2 = Up (N),** **3 = Left (W)**

```
 s <- length [c[m,n]] -1

//Back track direction matrix to find LCS
i<-m
J<-n
While s>=0
    If (b[i,j]=1) {
            lcs[s]=X_i
            i--
            j--
            s--
      } else if ((b[i,j]=2) then i--
        else if ((b[i,j]=3) then j--
```