Longest Common Subsequence

- Problem
  - Find the longest common subsequence given two sequences

Definitions

- Sequence and subsequence
  - Sequence $X = \langle x_1, x_2, ..., x_m \rangle$
  - $Z = \langle z_1, z_2, ..., z_k \rangle$ is a subsequence of X if

    $\exists$a strictly increasing sequence $\langle i_1, i_2, ..., i_k \rangle, \forall 1 \le j \le k, x_{i_j} = z_j$

  - Example Z = <B, D, Y> is a subsequence of X = <A, B, C, D, E, F, Y, Z>
- Prefix
  - The i_{th} prefix of $X = \langle x_1, x_2, ..., x_m \rangle$ is

    $X_i = \langle x_1, x_2, ..., x_i \rangle$

Characterizing a LCS

- Let $X = \langle x_1, x_2, ..., x_m \rangle$ and $Y = \langle y_1, y_2, ..., y_n \rangle$ be sequences and let $Z = \langle z_1, z_2, ..., z_k \rangle$ be any LCS of X and Y
  1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS X_{m-1} and Y_{n-1}
  2. If $x_m \neq y_n$, and $z_k \neq x_m$, then Z is an LCS X_{m-1} and Y
  3. If $x_m \neq y_n$, and $z_k \neq y_n$, then Z is an LCS X and Y_{n-1}

The optimal substructure of LCS

- Let $c[i,j]$ be the length of an LCS of the sequences X_i and Y_j, we obtain the following recurrence

$$c[i,j] = \begin{cases} 0 & if \quad i = 0 \,||\, j = 0 \\ c[i-1, j-1] + 1 & if \quad i, j > 0 \,\&\&\, x_i = y_j \\ \max(c[i, j-1], c[i-1, j]) & if \quad i, j > 0 \,\&\&\, x_i \neq y_j \end{cases}$$

An implementation using dynamic programming

```
LCS(X, Y)
{
    for (i=1; i<=m; i++)  c[i][0]=0;
    for (j=1; j<=n; j++)  c[0][j] =0;

     for (i=1; i<m; i++)
       for (j=1; j<n; j++) {
            if (x[i] == y[j]) {
               c[i][j] = c[i-1][j-1] +1;
               b[i][j] = '↖'
            } else if (c[i-1][j] >= c[i][j-1]) {
               c[i][j] = c[i-1][j];
               b[i][j] = '↑'
            } else {
               c[i][j] = c[i][j-1];
               b[i][j] = '←'
            }
       }
}
```

Example

		0	1	2	3	4	5	6
		y	B	D	C	A	B	A
0	x	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

Construct an LCS

- In our algorithm, the direction array *b* tracks the construction

```
PrintLCS(b, X, i, j)
{
    if (i==0 || j==0)
       return;
    switch (b[i][j]) {
       case '↖':
            PrintLCS(b, X, i-1, j-1)
            print x[i];
            break;
       case '↑':
            PrintLCS(b, X, i-1, j)
            break;
       case '←':
            PrintLCS(b, X, i, j-1)
            break;
    }
}
```