Influence in Running Time

- The running time of the recursive algorithm is as follows
  - Initialization steps or the steps following recursive calls are O(1) time.
  - This means $T(1) \geq 1$
  - But the recursion is carried out in loop. Thus

$$T(n) \geq 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1)$$

  - Which is

$$T(n) \geq 2 \sum_{i=1}^{n-1} T(i) + n$$

Exponential Running Time

- By method of iterative substitution we can prove $T(n) = \Omega(2^n)$.
- For base case we have $T(1) \geq 1 = 2^0$.
- Inductively for $n \geq 2$

$$
\begin{aligned}
T(n) &\geq 2\sum_{i=1}^{n-1} 2^{i-1} + n \\
&= 2\sum_{i=0}^{n-2} 2^{i} + n \\
&= 2(2^{n-1} - 1) + n \\
&= (2^n - 2) + n \\
&\geq 2^{n-1}
\end{aligned}
$$

Memoization

- Caching a straightfoward way to speed up certain slow functions by remember certain values which may be reused in computing such functions.
- Memoization is essentially same as caching.
- In dynamic programming we memoize the solutions to subproblems once and reuse several times.
- Memoize version indicates explicit reuse of solutions of subproblems.
- In that sense it is another variation of dynamic programming technique.

Overlapping Subproblems

```
memoizedMatChain(p) {
    n = length[p] − 1;
    for (i = 1; i ≤ n; k++)
        for (j = i; j ≤ n; j++)
            m[i, j] = ∞;
    return lookupChain(p, 1, n);
}

lookupChain(p, 1, n) {
    if (m[i, j] < ∞) return m[i, j];
    if (i == j) m[i, j] = 0;
    else for (k = i; k < j; k++) {
        q  =  lookupChain(p, i, k)
              +lookupChain(p, k + 1, j)
              +p_{i−1}p_k p_j;
        if (q < m[i, j]) m[i, j] = q;
    }
}
```


Longest Common Subsequence

- Given a sequence X, another sequence Y is considered as a subsequence of X if there exist a strictly increasing sequence of indices $< i_1, i_2, \ldots, i_k >$ such that for $j = 1, \ldots k$ we have $x_{i_j} = y_j$.
- Given Two sequences X and Y find the longest common subsequence Z.
- $< a, b, c, b, d, a, b >$ and $< b, d, c, a, b, a >$ are give sequences.
- $< b, c, a >$ is a common subsequence of above two sequences.
- $< b, c, b, a >$ and $< b, d, a, b >$ are LCSs as we cannot find a common subsequence of length 5.

Characterization of LCS

- Brute force approach is to find all subsequences of X and check each to find whether it is also a subsequence of Y and find the longest among these.
- Since a sequence of length m has 2^m subsequences, the above approach is exponential.
- LCS problem has an optimal substructure property.

Let $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, and $Z = \{z_1, z_2, \ldots, z_k\}$ be any LCS of X and Y then

1. If $x_m = y_n$ then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$ then $z_k \neq x_m$ implies Z is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$ then $z_k \neq y_n$ implies Z is an LCS of X and Y_{n-1}.

Proof of LCS Theorem

- If $z_k \neq x_m$ then we can always add $x_m = y_n$ to Z and obtain a sequence of length $k + 1$, contradicting length of LCS to be k.
- The prefix Z_{k-1} of Z must be a common subsequence of prefixes X_{m-1} and Y_{n-1}.
- Clearly, Z_{k-1} must be a LCS of X_{m-1} and Y_{n-1}.
- If there were a LCS of longer length, then x_m can be appended to this LCS to find a LCS which is longer than Z.

Proof of LCS Theorem

- If $x_m \neq y_n$, then Z is a subsequence of X_{m-1} and Y.
- If there were a LCS of length greater than k for X_{m-1} and Y, then this must also be a LCS of X and Y.
- Proof of the last part is identical.

Recursive Characterization

- From LCS theorem we find there are two possible ways the subproblems can be generated to solve LCS.
  - If $x_m = y_n$ then find LCS of X_{m-1} and Y_{n-1}.
  - Else solve two LCS problems, namely for X_{m-1} and Y and for X and Y_{n-1} and find the longer of the two LCSs.
- Overlapping subproblem property in LCS
  - Solve LCS for (X, Y_{n-1}) and (X_{m-1}, Y).
  - To solve LCSs for each of the above solve LCS for X_{m-1} and Y_{n-1}.