CS208 Lab8 Practice

12312110 李轩然

DDL: May.18

Fibonacci DP

Code

```
class Solution {
public:
    int Fibonacci(int n) {
        int a[n + 1];
        a[1] = 1; a[2] = 1;

        for (int i=3; i<=n; i++)
        a[i] = a[i-1] + a[i-2];

        return a[n];
    }
};
```

Analysis

Time complexity: Obviously $O(n)$.

LCS DP

Code

```
class Solution {
public:
    int LCS(string s1, string s2) {
        int n = s1.length(), m = s2.length();
        int dp[n + 1][m + 1];
        for (int i = 0; i <= n; i++)
        dp[i][0] = 0;
        for (int i = 0; i <= m; i++)
        dp[0][i] = 0;

        for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
        {
```

```
            if (s1[i - 1] == s2[j - 1]) dp[i][j] = dp[i - 1][j - 1] + 1;
            else dp[i][j] = max(dp[i][j - 1], dp[i - 1][j]);
        }
        return dp[n][m];
    }
};
```

Analysis

First consider the border cases, when the length of $s1$ or $s2$ is 0, the LCS must be 0.

Then use two pointers, let's say i and j, to traverse $s1$ and $s2$, if $s1[i] == s2[j]$, then the length of LCS now is the length of LCS of last step plus 1, else we take the longer LCS of $(i, j-1)$ and $(i-1, j)$.

$$dp[i][j] = \begin{cases} dp[i-1][j-1]+1, & s1[i-1] = s2[j-1] \\ \max(dp[i-1][j], dp[i][j-1]), & s1[i-1] \neq s2[j-1] \end{cases}$$

Time complexity: $O(nm)$.