Longest Common Subsequence

> 最长公共子序列

dp[i][j] 为 S1[0] ~S1[i -1] 与 S2[0] ~ S2[j - 1]的最长公共子序列

对于S1[i - 1]与S2[j - 1]：

1. S1[i - 1] == S2[j - 1]，则 dp[i][j] = dp[i - 1][j - 1] + 1 （问题转换为求 dp[i -1][j - 1] ，即 S1[0 ~ i - 2]，S2[0 ~ j - 2]的最长公共子序列
2. S1[i - 1] ≠ S2[j - 1]，则 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

```
#include <bits/stdc++.h>

using namespace std;

int LCS(string s1, string s2, int len1, int len2) {
    vector<vector<int>> dp(len1 + 1, vector<int>(len2 + 1));
    for (int i = 0; i <= len1; ++i) {
        for (int j = 0; j <= len2; ++j) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }
    return dp[len1][len2];
}

int main() {
    int T;
    scanf("%d\n", &T);
    while (T--) {
        int len1, len2;
        scanf("%d %d", &len1, &len2);
        string s1, s2;
        cin >> s1 >> s2;
        printf("%d\n", LCS(s1, s2, len1, len2));
    }
    return 0;
}
```

空间优化（二维数组）

通过观察发现，dp[i][j]的取值仅仅与 dp[i- 1][j - 1] 、dp[i - 1][j]或者dp[i][j - 1]相关

也就是说当前的dp[i][j]只与当前行或者前一行有关，所以只需要dp[2][max(len1, len2) + 1]就可以实现计算LCS

最开始 cur = 1, pre = 0，然后不断交换 pre与cur即可

```
#include <bits/stdc++.h>

using namespace std;
int LCS(string s1, string s2, int len1, int len2) {
    vector<vector<int>> dp(2, vector<int>(max(len1, len2) + 1));
    int cur = 1, pre = 0;
    for (int i = 1; i <= len1; ++i) {
        for (int j = 1; j <= len2; ++j) {
            if (s1[i - 1] == s2[j - 1])
                dp[cur][j] = dp[pre][j - 1] + 1;
            else
                dp[cur][j] = max(dp[pre][j], dp[cur][j - 1]);
        }
        swap(cur, pre);
    }
    return dp[pre][len2];
}

int main() {
    int T;
    scanf("%d\n", &T);
    while (T--) {
        int len1, len2;
        scanf("%d %d", &len1, &len2);
        string s1, s2;
        cin >> s1 >> s2;
        printf("%d\n", LCS(s1, s2, len1, len2));
    }
    return 0;
}
```

空间优化（一维数组）

如果只需要求LCS的长度，实际上只需要dp[n]就行了，应用滚动数组。因为dp[i][j]由dp[i-1][j-1],dp[i-1][j],dp[i][j-1]，用dp[j]表示dp[i][j]，则更新dp[j]时用pre存储dp[i-1][j-1]，此时的dp[j-1]表示dp[i][j-1]，此时的dp[j]表示dp[i-1][j]，

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
```

```
char s1[505],s2[505];
int len1,len2,pre,tmp;
int dp[505];

int main(){
    while(~scanf("%s%s",s1,s2)){
        len1=strlen(s1),len2=strlen(s2);
        memset(dp,0,sizeof(dp));
        for(int i=1;i<=len1;++i){
            pre=0;
            for(int j=1;j<=len2;++j){
                tmp=dp[j];
                if(s1[i-1]==s2[j-1])
                    dp[j]=pre+1;
                else
                    dp[j]=max(dp[j-1],dp[j]);
                pre=tmp;
            }
        }
        printf("%d\n",dp[len2]);
    }
    return 0;
}
```