动态规划时间复杂度的例子：

dp[i][j] = max(dp[i][k], dp[k+1][j]) + cost // 区间型动态规划

时间复杂度：O(i的范围\*j的范围\*k的范围)

dp[i] = or{dp[j] and j+1到i是一个单词} // 前缀型动态规划

时间复杂度：O(i的范围\*j的范围)=O(n\*maxWordLen)

滚动数组对动态规划的状态转移方程优化：如果状态依赖关系只存在于相邻的几层之间，则可以使用滚动数组进行空间优化。

例如：数字三角形，lintcode 109，状态转移方程为：

dp[i][j] = max(dp[i-1][j], dp[i-1][j-1]) + A[i][j]

第i层的结果只和第i-1层相关，而和i-2，i-3..等无关。利用滚动数组进行优化：

dp[i%2][j]=max(dp[(i-1)%2][j], dp[(i-1)%2][(j-1)%2]) + A[i][j]

第i层的计算结果和第i-2层，第i-4层都用的是同一个存储空间。

滚动数组不能同时滚动i和j，因为在整个循环中i是单调递增（或单调递减）的，但j并不是单调递增的，这就意味着，第j层不能覆盖第j-2层，因为之后还会用到第j-2层的内容。

写动态规划时，可以先写不滚动的数组，然后再优化。

滚动数组滚动的是第一重循环的变量，因为第一重循环的变量是单调的（上升或下降），不滚动第二重或第三重，并且不能两个维度一起滚动，只滚动一个维度。

对于一个有向图的问题，如果图中不存在环，可以使用拓扑排序，就可以使用动态规划。

接龙型动态规划：

单词接龙：