给定字符串，找出其中最长的重复出现并且非重叠的子字符串。

例子：

输入：str = "geeksforgeeks"

输出： geeks

输入：str = "aabaabaaba"

输出： aaba

1. 实现思想

算法的实现思想是：**使用一个二维数据dp[i][j]表示字符串中以第i个位置为子字符串结尾和以第j个位置为子字符串结尾重复但不重叠的字符串最长长度（j>i），**分析可知：若str[i]和str[j]字符不一样的话dp[i][j]=0；若str[i]和str[j]字符一致的话考虑i-1与j-1的子问题即可，dp[i][j]=dp[i-1][j-1]+1。根据上面的分析可以明确得出dp[i][j]的递推关系为：

**dp[i][j]=dp[i-1][j-1]+1, str[i]=str[j];**

**dp[i][j]=0, str[i]!=str[j];**

设置边界条件都为0即可，然后根据递推关系维护dp数组，再根据dp数组中的最大值即可知道重复但不重叠的最长字符串长度和具体字符串。算法的时间复杂度为两层循环，即O（n^2）。

（蛮力算法大致思路为从1-n循环，根据子字符串的长度进行遍历，复杂度为O（n^3））

算法步骤为：

1. 创建一个二维的动态规划表dp，其中dp[i][j]表示字符str中str[i-1]和str[j-1]结尾的两个子字符串的最长公共后缀的长度。

2. 初始化result为空字符串，用于存储最长的重复但不重叠子字符串。

3. 使用两层循环来遍历字符串，外层循循环i从1到n，表示以第i个字符结尾的子字符串，内层循环j从i+1到n+1，表示与外层循环i对应的子字符串的可能匹配位置。

4. 在内层循环中，首先检查str[i-1]和str[j-1]是否相等，如果相等，说明在两个子字符串中找到了一个字符匹配。

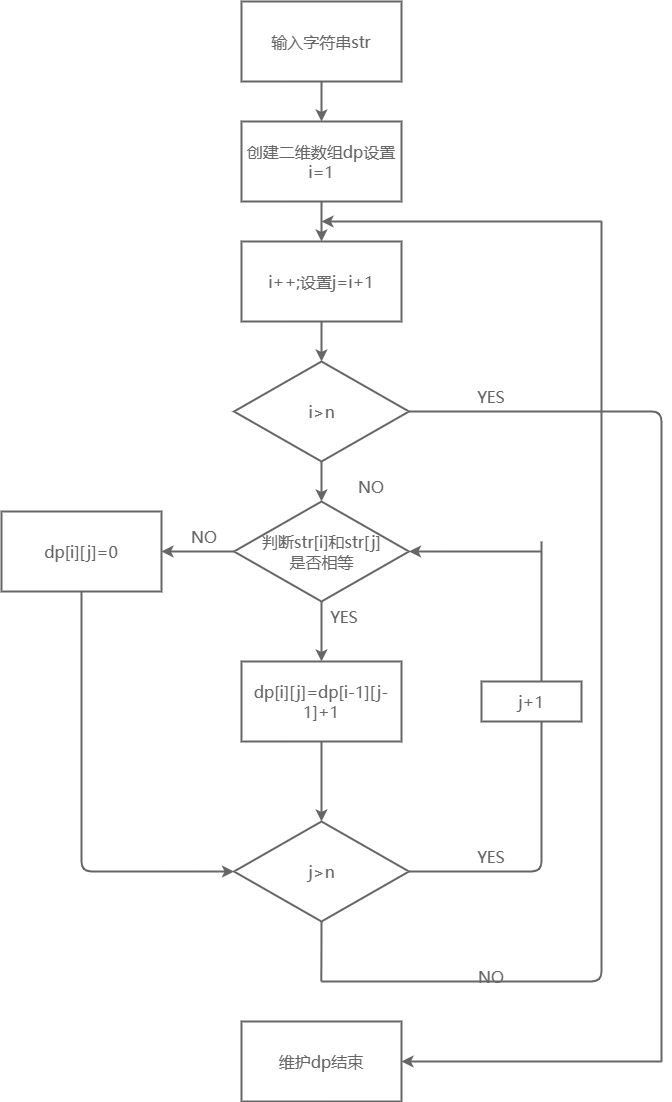
5. 根据匹配情况，通过以下逻辑来更新dp[i][j]：

如果当前字符匹配，检查是否能构成更长的重复但不重叠子字符串，即dp[i-1][j-1] + 1 > j - i。如果是，将dp[i][j]设置为j - i，否则，将dp[i][j]设置为dp[i-1][j-1] + 1。

6. 如果dp[i][j]大于max\_length，则更新max\_length为dp[i][j]，同时更新result为以第i个字符为结尾的重复但不重叠子字符串。

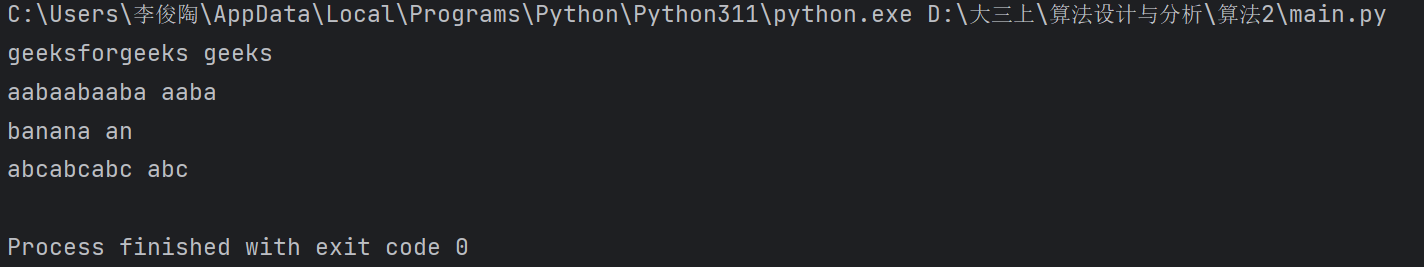
7. 最终，result中将包含最长的重复但不重叠子字符串。

算法流程图如下：



1. 输出截图

除了所给例子之外还自行设置了几个测试数据，检验算法的正确性，均给出了最长的重复但不重叠的子字符串：



分析所给例子中的测试数据，给出关键的dp数组最终情况：

geeksforgeeks

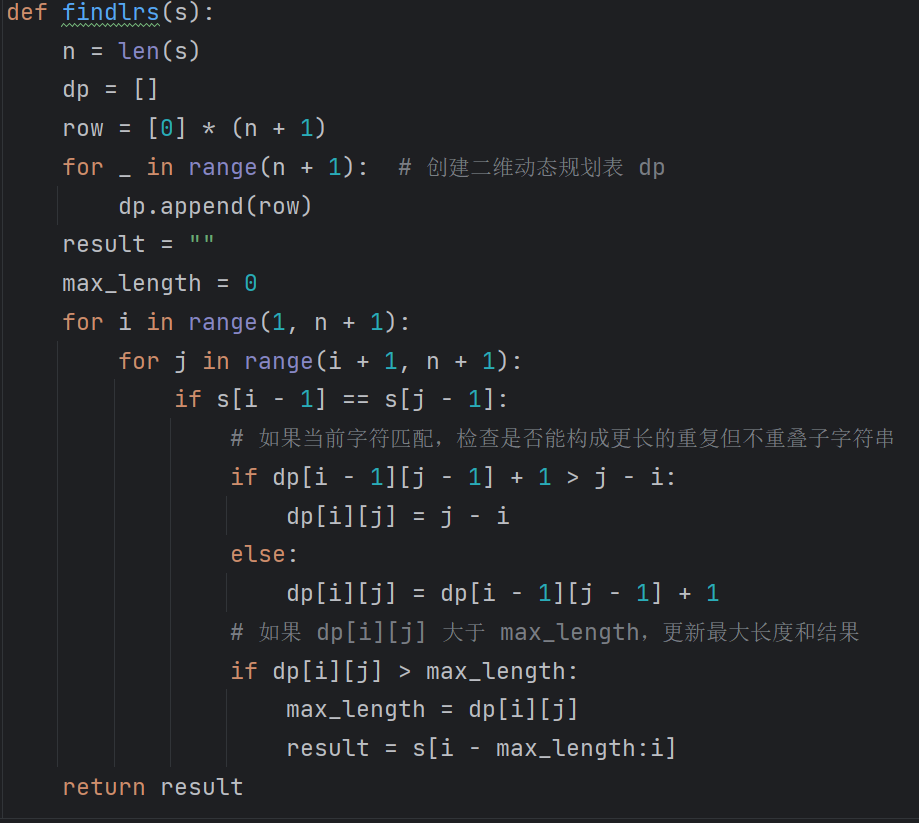
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | g | e | e | k | s | f | o | r | g | e | e | k | s |
| g |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| e |  |  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 |
| e |  |  |  | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 0 | 0 |
| k |  |  |  |  | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| s |  |  |  |  |  | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| f |  |  |  |  |  |  | 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| o |  |  |  |  |  |  |  | 0 | 1 | 2 | 3 | 4 | 5 |
| r |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |
| g |  |  |  |  |  |  |  |  |  | 2 | 3 | 4 | 5 |
| e |  |  |  |  |  |  |  |  |  |  | 3 | 4 | 5 |
| e |  |  |  |  |  |  |  |  |  |  |  | 4 | 5 |
| k |  |  |  |  |  |  |  |  |  |  |  |  | 5 |
| s |  |  |  |  |  |  |  |  |  |  |  |  |  |

aabaabaaba

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | a | b | a | a | b | a | a | b | a |
| a |  | 1 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 |
| a |  |  | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 |
| b |  |  |  | 1 | 2 | 3 | 1 | 2 | 3 | 1 |
| a |  |  |  |  | 1 | 3 | 3 | 4 | 3 | 4 |
| a |  |  |  |  |  | 3 | 2 | 3 | 3 | 4 |
| b |  |  |  |  |  |  | 2 | 3 | 3 | 4 |
| a |  |  |  |  |  |  |  | 1 | 3 | 3 |
| a |  |  |  |  |  |  |  |  | 3 | 2 |
| b |  |  |  |  |  |  |  |  |  | 2 |
| a |  |  |  |  |  |  |  |  |  |  |

1. 源代码

寻找最长重复但不重叠的子字符串的函数：



测试样例及调用函数：

