第二次上机作业

在这里我使用算法实现的思想是动态规划。以下是算法的具体实现步骤和思想：

算法思想：

1. **引入参数来界定子问题的边界.注意子问题的重叠程度**

   在这段代码中，子问题被定义为找出长度为 dp[i] [j] 的重复子字符串，其中 i 和 j 分别是子字符串在原字符串中的起始和结束索引。dp[i] [j] 表示从索引 i 到索引 j 的子字符串与从索引 i-dp[i] [j] 到 j-dp[i] [j] 的子字符串相同。

2. **给出带边界参数的优化函数定义与优化函数的递推关系，找到递推关系的初值**

   - 优化函数 dp[i] [j] 定义为从索引 i 到 j 的子字符串与从索引 i-dp[i] [j] 到 j-dp[i] [j] 的子字符串相同的最大长度。
   - 递推关系的初值是 dp[0] [0] = 0，表示空字符串与空字符串相同。

3. **判断该优化问题是否满足优化原则**

   这个问题满足优化原则，我们可以分解问题，找出更小的重复子字符串，然后使用这些子问题的解来构建更大的问题的解。

4. **考虑是否需要标记函数**

   在这段代码中，不需要额外的标记函数，因为最长重复子字符串的结束索引和长度已经通过 endIndex 和 maxLength 变量来跟踪。

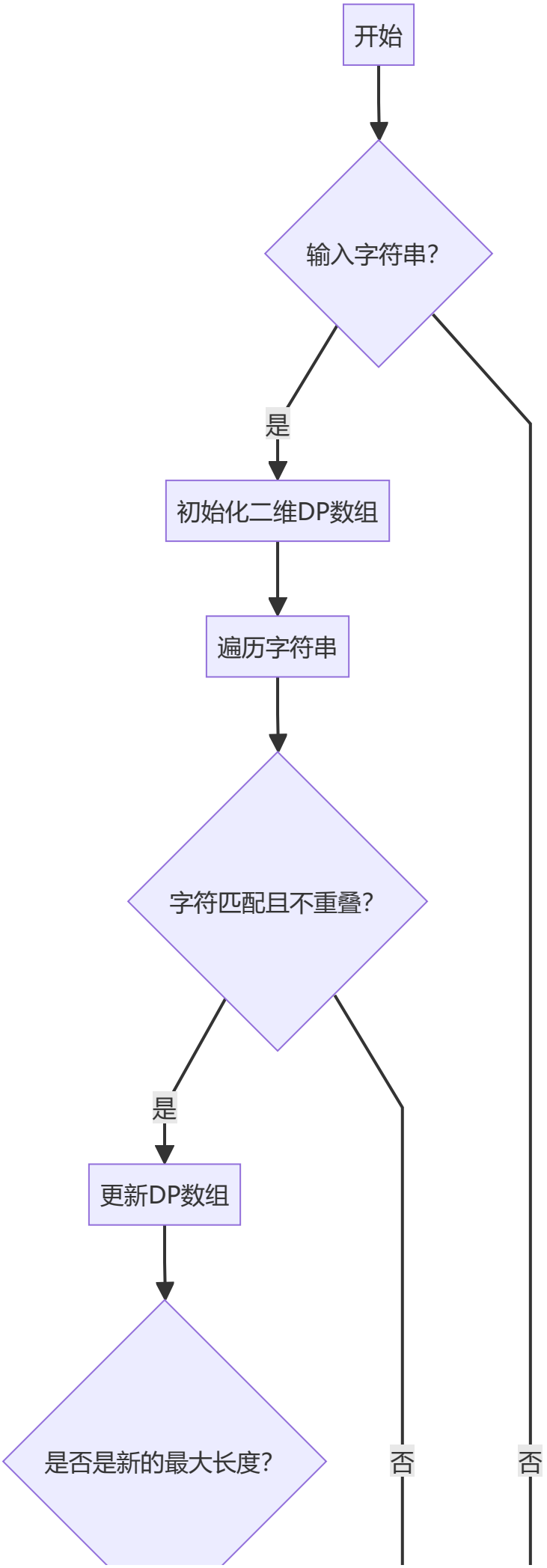
5. **采用自底向上的实现技术，从最小的子问题开始迭代计算，计算中用备忘录保留优化函数和标记函数的值**

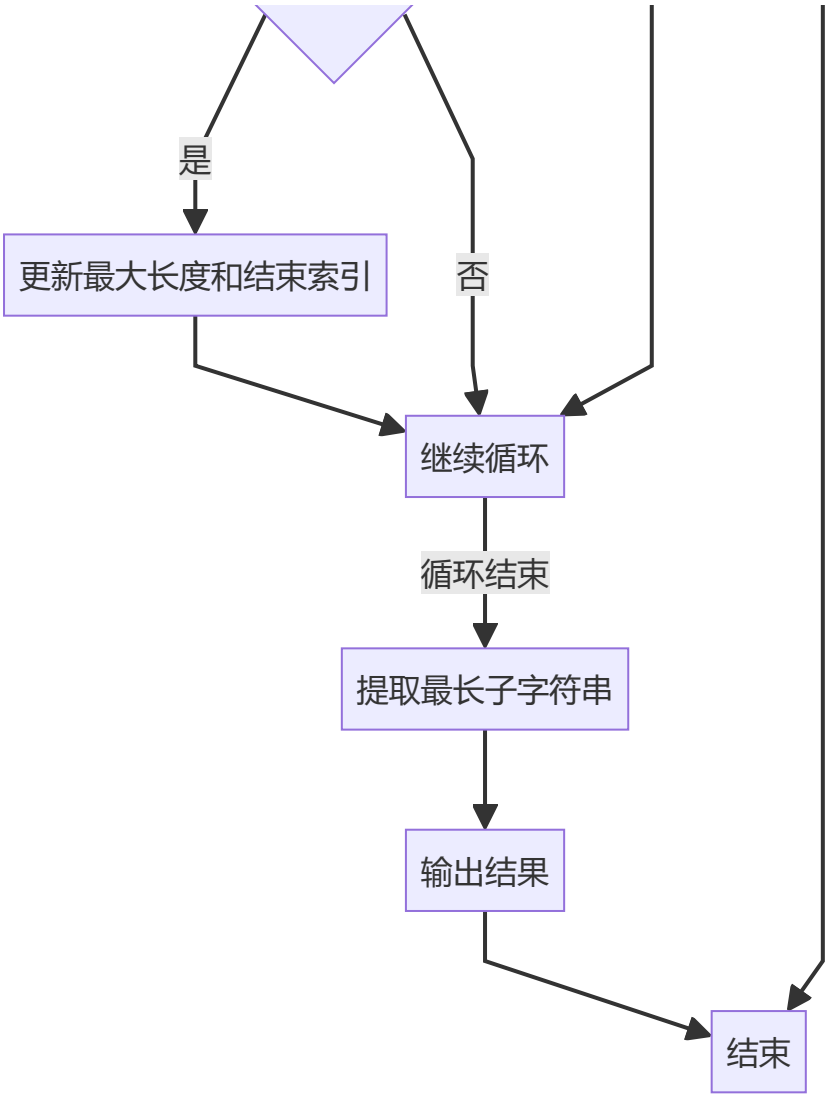
   代码使用了一个二维动态规划表 dp 来保存子问题的解。它从最小的子问题（长度为1的子字符串）开始，逐步构建到更大的子问题。

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

string findLongest(const string &str) {
    int n = str.length();
    vector<vector<int>> dp(n + 1, vector<int>(n + 1, 0));
    int maxLength = 0; // 最长重复子字符串的长度
    int endIndex = 0; // 最长重复子字符串的结束索引

    // 构建dp数组
    for (int i = 1; i <= n; ++i) {
        for (int j = i + 1; j <= n; ++j) {
            // 检查字符是否相同且子字符串不重叠
            if (str[i - 1] == str[j - 1] && dp[i - 1][j - 1] < (j - i)) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
                // 更新最长长度和结束索引
                if (dp[i][j] > maxLength) {
                    maxLength = dp[i][j];
                    endIndex = i - 1;
```

```
                }
            }
        }
    }

    // 提取最长重复非重叠子字符串
    if (maxLength > 0) {
        return str.substr(endIndex - maxLength + 1, maxLength);
    }
    return ""; // 如果没有重复子字符串，返回空字符串
}

int main() {
    string str1 ;
    while(cin >> str1){
        cout << "Longest repeating non-overlapping substring: " <<
findLongest(str1) << endl;
    }
    return 0;
}
```

流程图：

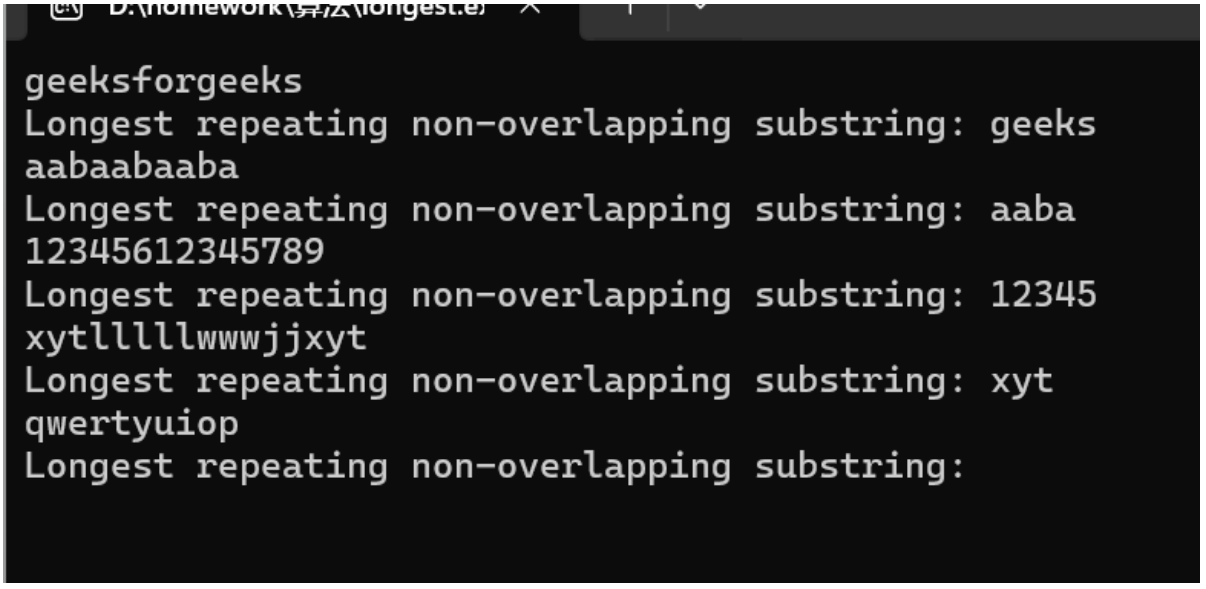

开始
输入字符串?
是
初始化二维DP数组
遍历字符串
字符匹配且不重叠?
是
更新DP数组
是否是新的最大长度?
否
否



代码实现：

输入字符串，回车即可找出最大字符串。若无最大重复字符串则返回空

结果截图：



可得出算法正确。

时间复杂度：

这段代码的时间复杂度是 O(n^3)，其中 n 是字符串的长度。这是因为外层循环遍历了 n 个可能的起始位置，内层循环遍历了 n 个可能的结束位置，而每次比较操作需要 O(n) 的时间来更新 dp 表。