2022.11.11

每日一题 1704 判断字符串的两半是否相似

简单题，将元音存进一个 set 里然后遍历判断即可。

```
class Solution {
private:
    set<char> idx;
public:
    bool halvesAreAlike(string s) {
        init();
        int len = s.size();
        string sub = s.substr(0,len>>1);
        int pre = 0, tot = 0;
        for(auto ch : sub) pre += idx.count(ch);
        for(auto ch : s) tot += idx.count(ch);
        return pre * 2 == tot;
    }

    void init() {
        string str = "aeiou";
        for(auto ch : str)
            { idx.insert(ch);   idx.insert(toupper(ch)); }
    }
};
```

或者换一种科技写法：

```
class Solution {
public:
    bool halvesAreAlike(string s) {
        string a = s.substr(0, s.size() / 2);
        string b = s.substr(s.size() / 2);
        string h = "aeiouAEIOU";
        int sum1 = 0, sum2 = 0;
        for (int i = 0; i < a.size(); i++) {
            if (h.find_first_of(a[i]) != string::npos) {
                sum1++;
            }
        }
        for (int i = 0; i < b.size(); i++) {
            if (h.find_first_of(b[i]) != string::npos) {
                sum2++;
            }
        }
        return sum1 == sum2;
    }
```

如果是 $python$ 的话可以更短：

```python
class Solution:
    def halvesAreAlike(self, s: str) -> bool:
        VOWELS = "aeiouAEIOU"
        a, b = s[:len(s) // 2], s[len(s) // 2:]
        return sum(c in VOWELS for c in a) == sum(c in VOWELS for c in b)
```

1092 最短公共超序列

线性DP，与 LCS 非常相似。

设 $dp[i][j]$ 表示 $str1$ 取 $[0,1,\dots,i]$ 而 $str2$ 取 $[0,1,\dots,j]$ 时的最短公共超序列长度，则有转移方程：

$$dp[i+1][j+1]=min\begin{cases}dp[i+1][j]+1\\dp[i][j+1]+1\\dp[i][j]+1+(str1[i+1]\neq str2[j+1])\end{cases}$$

当找到最短的公共超序列长度时，根据状态回溯即可找到该超序列的具体字符。

```cpp
class Solution {
public:
    int dp[1010][1010];
    string shortestCommonSupersequence(string str1, string str2) {
        for(int i = 0; i < str1.size(); i ++ )
            dp[i + 1][0] = i + 1;
        for(int j = 0; j < str2.size(); j ++ )
            dp[0][j + 1] = j + 1;
        for(int i = 0; i < str1.size(); i ++ ) {
            for(int j = 0; j < str2.size(); j ++ ) {
                dp[i + 1][j + 1] = min(dp[i][j + 1] + 1, dp[i + 1][j] + 1});
                dp[i + 1][j + 1] = min(dp[i + 1][j + 1], dp[i][j] + 1 + (str1[i]
!= str2[j]));
            }
        }
        int len = dp[str1.size()][str2.size()];
        string ans;
        int i = str1.size(), j = str2.size();
        while(i > 0 && j > 0) {
            if(len == dp[i - 1][j] + 1) {
                ans = str1[i - 1] + ans;
                len -= 1;
                i --;
            } else if(len == dp[i][j - 1] + 1) {
                ans = str2[j - 1] + ans;
                len -= 1;
                j --;
            } else {
                if(str1[i - 1] == str2[j - 1]) {
                    ans = str1[i - 1] + ans;
                    len -= 1;
                } else {
                    ans = str1[i - 1] + (str2[j - 1] + ans);
                    len -= 2;
                }
                i --;
```

```
                j --;
            }
        }
        while(j > 0) ans = str2[--j] + ans;
        while(i > 0) ans = str1[--i] + ans;
        return ans;
    }
};
```