阶乘字符串

DoremySwee

2020 年 11 月 4 日

原题链接

给定一个由前 n 个小写字母组成的串 S。串 S 是阶乘字符串当且仅当前 n 个小写字母的全排列（共 $n!$ 种）都作为的子序列（可以不连续）出现。

由这个定义出发，可以得到一个简单的枚举法去验证，但是它实在太慢了。所以现在请你设计一个算法，在 1 秒内判断出给定的串是否是阶乘字符串。

数据规模	内存限制	运行时间
$\|S\| \leq 450, n \leq 26$	125 MB	1.0 s

题解. 暴力解法自然是枚举全排列并检验，30% 的分数到手，不过这个程序没有其它太大意义，甚至由于阶乘级的时间复杂度难以对拍。不多作讨论。

考虑 70% 的分数。$n \leq 20$，可以考虑 $\mathcal{O}(2^n)$ 的算法。既然不能进行全排列的枚举，那么完全可以考虑状态压缩动态规划，枚举一个字母是否出现在状态中。由于要求解的是整个字符串中是否存在前 n 个字母的全排列，则考虑 $f(x)$ 表示 x 对应的几个字母的全排列最早出现在字符串种的何处。状态转移方程举例：

$$\begin{aligned} f(a,b,c,e) = \max(&\text{从}f(a,b,c)\text{开始找第一个}\texttt{'e'},\\ &\text{从}f(a,b,e)\text{开始找第一个}\texttt{'c'},\\ &\text{从}f(a,c,e)\text{开始找第一个}\texttt{'b'},\\ &\text{从}f(b,c,e)\text{开始找第一个}\texttt{'a'}) \end{aligned}$$

但是如果对于每一个寻找都是通过循环寻找的话，时间会有点长，只能拿 50 分，需要进行一定的预处理。可以在预处理之后用二分查找，但是方便起见建议添加一个数组标记每一个位置下一个目标字母的位置，所需时间不长，450×26 而已。

最后，考虑剩下的分数点。可以发现，对于过大的 n，可以直接输出"No"，至于这个值如何确定？按照 $n=3 \implies S=$ abcbab，$n=4 \implies S=$ abcdcbabcdcba 这种构造方式 S 的长度为 n^2-n+1，$n \le 21$，不过证明过于困难。为保险起见，因为 $n=22$ 时用时是可以接受的，取 $n \ge 23$ 时直接输出"No"。

代码：

```cpp
#include <iostream>

using namespace std;

int main(){
   int t;
   cin >> t;
   while (t--) {
      int n;
      cin >> n;
      string s;
      cin >> s;
      if (n >= 23) {
         cout << "NO" << endl;
      } else {
         short NEXT[26][450];
         for (int i = s.length() - 1; i >= 0; i--) {
            if (i == s.length() - 1) {
               for (int j = 0; j < 26; j++)
                  NEXT[j][i] = -1;
            } else {
               for (int j = 0; j < 26; j++)
                  NEXT[j][i] = NEXT[j][i + 1];
            }
            NEXT[s[i] - 'a'][i] = i;
         }
         short *f = new short[1 << n];
         for (int i = 0; i < (1 << n); i++) {
            f[i] = 0;
            for (int j = 0; (1 << j) <= i; j++) {
               if (i & (1 << j)) {
                  if (NEXT[j][f[i - (1 << j)]] == -1) {
                     cout << "NO" << endl;
                     return 0;
                  }
```

```
36                      f[i] = max(f[i], NEXT[j][f[i - (1 << j)]]);
37                  }
38              }
39          }
40          cout << "YES" << endl;
41      }
42  }
43  }
```