



LeetCode题目：Interleaving String，二维动态规划

By uniEagle | 2012 年 9 月 29 日 - 23:22 | C++, Develop, 算法

分析

题目给定3个字符串(s1,s2,s3)，看s3是否有s1和s2通过交织可以得到。

可以这么来看这个问题，每次从s3开头拿出来一个字符，如果s1的开头或者s2的开头有这个字符的话，就消掉相应的字符，把这个操作记为del。

这样看待这个问题的话，好像挺简单的，很直观。

但是当遇到case：s1=aa,s2=ab,s3=abaa，的时候可以看出当s1和s2同时可以进行del操作的时候，选择就成了一个必须考虑的问题。

假如最开始那个a，消掉了s1中的第一个a，那么就进行不下去了。

所以最后这个问题其实并不那么简单，假如函数

```
bool isInterleaving(string &s1, int len1, string &s2, int len2, string &s3, int len3);
```

表示子问题：si取前leni个字符的话，那么实际上可以得到这样的一个公式：

```
isInterleaving(s1,len1,s2,len2,s3,len3)
= (s3.lastChar == s1.lastChar) && isInterleaving(s1,len1 - 1,s2,len2,s3,len3 - 1)
|| (s3.lastChar == s2.lastChar) && isInterleaving(s1,len1,s2,len2 - 1,s3,len3 - 1)
```

由于len3 === len1 + len2，所以这个问题里面实际上存在着两个变量，是一个二维动态规划题目。

从矩阵的角度来看的话，每一个元素的值，依赖于它的上边和左边两个值。

最后写出程序，用24ms过大测试集合。

题目描述：Interleaving String

Given s1, s2, s3, find whether s3 is formed by the interleaving of s1 and s2.

For example,

Given:

s1 = "aabcc",

s2 = "dbbca",

When s3 = "aadbcbcbac", return true.

When s3 = "aadbbaacc", return false.

正解就是二维动态规划

```
class Solution {
public:
    bool isInterleave(string s1, string s2, string s3) {
```

```

if(s3.size() != s1.size() + s2.size())
    return false;
//create indicator
vector<vector<bool>> > match(s1.size() + 1, vector<bool>(s2.size() + 1, false));
//initialization the first row and the first column
match[0][0] = true;
for( int l1 = 1; l1 <= s1.size(); ++ l1 ) {
    char c1 = s1[l1 - 1];
    char c3 = s3[l1 - 1];
    if (c1 == c3) {
        match[l1][0] = true;
    } else
        break;
}
for( int l2 = 1; l2 <= s2.size(); ++ l2 ) {
    char c2 = s2[l2 - 1];
    char c3 = s3[l2 - 1];
    if (c2 == c3) {
        match[0][l2] = true;
    } else
        break;
}
//work through the rest of matrix using the formula
for( int l1 = 1; l1 <= s1.size(); ++ l1 ) {
    char c1 = s1[l1 - 1];
    for( int l2 = 1 ; l2 <= s2.size() ; ++ l2 ) {
        char c2 = s2[l2 - 1];
        int l3 = l1 + l2;
        char c3 = s3[l3 - 1];
        if (c1 == c3) {
            match[l1][l2] = match[l1 - 1][l2] || match[l1][l2];
        }
        if (c2 == c3) {
            match[l1][l2] = match[l1][l2 - 1] || match[l1][l2];
        }
    }
}
//the last element is the result
return match[s1.size()][s2.size()];
}
};

```

几个失败的想法

一开始的想法，错在当s1,s2同时匹配的时候，这里存在一个策略问题，比如测试用例：aa,ab,abaa，就过不了。

```

class Solution {
public:
    bool isInterleave(string s1, string s2, string s3) {
        if(s3.size() != s1.size() + s2.size())
            return false;
        int i1 = 0;
        int i2 = 0;

```

```

for(int i3 = 0 ; i3 < s3.size(); ++i3) {
    if(i1 < s1.size() && s1[i1] == s3[i3]) {
        ++i1;
    } else if (i2 < s2.size() && s2[i2] == s3[i3]) {
        ++i2;
    } else
        return false;
}
return true;
}
};

```

第二个错误想法，先从s3中标记掉s1中出现的内容，然后在看剩下的部分是不是s2，也不对，一样的存在“标记掉s1”的策略问题

```

class Solution {
public:
    bool isInterleave(string s1, string s2, string s3) {
        if(s3.size() != s1.size() + s2.size())
            return false;
        vector<bool> visited(s3.size(),false);
        int i3 = 0;
        for( int i1 = 0; i1 < s1.size() ; ++i1 ) {
            bool match = false;
            while(i3 < s3.size()) {
                if(s3[i3] == s1[i1]) {
                    visited[i3 ++] = true;
                    match = true;
                    break;
                }
                ++i3;
            }
            if(!match)
                return false;
        }
        i3 = 0;
        for( int i2 = 0; i2 < s2.size() ; ++i2 ) {
            bool match = false;
            while(i3 < s3.size()) {
                if (visited[i3]) {
                    ++i3;
                    continue;
                }
                if(s3[i3 ++] == s2[i2]) {
                    match = true;
                    break;
                }
            }
            if(!match)
                return false;
        }
        return true;
    }
};

```


所以，绕不开策略问题，就得在做出决策的时候让程序分支。下面是一个正确但大集合会超时的递归算法：

```
class Solution {
public:
    bool isInterleave(string &s1,int i1, string &s2, int i2, string &s3, int i3) {
        if(i1 == s1.size() && i2 == s2.size() && i3 == s3.size())
            return true;
        bool match = false;
        if(i1 < s1.size() && s1[i1] == s3[i3]) {
            match = isInterleave(s1, i1 + 1, s2, i2, s3, i3 + 1);
        }
        if (!match && i2 < s2.size() && s2[i2] == s3[i3]) {
            match = isInterleave(s1, i1, s2, i2 + 1, s3, i3 + 1);
        }
        return match;
    }
    bool isInterleave(string s1, string s2, string s3) {
        if(s3.size() != s1.size() + s2.size())
            return false;
        return isInterleave(s1,0,s2,0,s3,0);
    }
};
```

Tagged [algorithm](#), [c++](#), [dp](#), [Interleaving String](#), [leetcode](#), [动态规划](#), [算法](#). Bookmark the [permalink](#).

7 Responses to *LeetCode*题目： *Interleaving String*，二维动态规划

Pingback: [LeetCode: Interleaving String | Hongze Zhao's Blog](#)



Zhanrui

2013 年 5 月 26 日 at 20:07

好像没那么复杂吧...我在找有没有不是 $O(n^2)$ 的做法.

```
class Solution {
public:
    bool isInterleave(string s1, string s2, string s3) {
        // Start typing your C/C++ solution below
        // DO NOT write int main() function
        int i, j;
        if(s1.size() + s2.size() != s3.size()) return false;
        vector<vector > f(s1.size() + 1, vector(s2.size() + 1));
        for(i = 0; i <= s1.size(); i++){
            for(j = 0; j 0 and s2[j-1] == s3[i+j-1] and f[i][j-1]
            or i>0 and s1[i-1] == s3[i+j-1] and f[i-1][j]);
        }
    }
    return f[s1.size()][s2.size()];
}
```

回复



还要一起用啊...

```
class Solution {
public:
bool isInterleave(string s1, string s2, string s3) {
// Start typing your C/C++ solution below
// DO NOT write int main() function
int i, j;
if(s1.size() + s2.size() != s3.size()) return false;
vector<vector > f(s1.size() + 1, vector(s2.size() + 1));
for(i = 0; i <= s1.size(); i++){
for(j = 0; j 0 and s2[j-1] == s3[i+j-1] and f[i][j-1]
or i>0 and s1[i-1] == s3[i+j-1] and f[i-1][j]);
}
}
return f[s1.size()][s2.size()];
}
};
```

回复



antiagainst 2013 年 1 月 30 日 at 02:13

递归是可以过这个题的。楼主只注意了字符串头的比较，没有注意字符串尾的比较。同时考虑字符串尾可以剪掉很多不必要的分支。附我写的200ms左右过大数据的程序：

```
class Solution {
public:
    string str1, str2, str3;
    bool isInterleave(string s1, string s2, string s3) {
        // Start typing your C/C++ solution below
        // DO NOT write int main() function
        if (s3.size() != s1.size() + s2.size()) return false;
        str1 = s1;
        str2 = s2;
        str3 = s3;
        return check(0, s1.size() - 1, 0, s2.size() - 1, 0, s3.size() - 1);
    }
    bool check(int p1, int q1, int p2, int q2, int p3, int q3) {
        if (p3 > q3) return true;
        if (p1 > q1) {
            while (p3 <= q3 && str3[p3] == str2[p2]) p2++, p3++;
            if (p3 <= q3) {
                while (p3 <= q3 && str3[p3] == str1[p1]) p1++, p3++;
                if (p3 <= q3) return false;
                return true;
            }
        }

        if (str3[p3] == str1[p1]) {
            if (str3[q3] == str1[q1] && check(p1 + 1, q1 - 1, p2, q2, p3 + 1, q3 - 1)) return true;
            if (str3[q3] == str2[q2] && check(p1 + 1, q1, p2, q2 - 1, p3 + 1, q3 - 1)) return true;
        }
        if (str3[p3] == str2[p2]) {
            if (str3[q3] == str1[q1] && check(p1, q1 - 1, p2 + 1, q2, p3 + 1, q3 - 1)) return true;
            if (str3[q3] == str2[q2] && check(p1, q1, p2 + 1, q2 - 1, p3 + 1, q3 - 1)) return true;
        }
    }
};
```

```
    }  
    return false;  
}  
};
```

回复



antiagainst 2013 年 1 月 30 日 at 02:18

Oops, 贴出来的代码被吃掉了一部分.....

回复



uniEagle 2013 年 1 月 30 日 at 10:00

把代码包含在<pre></pre>中就可以保留格式和各种符号了。

回复



Monamona 2013 年 10 月 20 日 at 04:26

这个剪枝神了。。。

回复

发表评论

电子邮件地址不会被公开。 必填项已用*标注

姓名 *

电子邮件

*

站点

评论

您可以使用这些HTML标签和属性： <abbr title=""> <acronym title=""> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <strike>

发表评论

☐ 通过邮件通知我后续评论

☐ 通过邮件通知我有新文章

Search

OK

功能

- 登录
- 文章RSS
- 评论RSS
- WordPress.org

Tags