Implement strStr()

Implement strStr().

Returns the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Solution 1

```
int strStr(char *haystack, char *needle) {
        if (!haystack || !needle) return -1;
        for (int i = 0; ; ++i) {
            for (int j = 0; ; ++j) {
                if (needle[j] == 0) return i;
                if (haystack[i + j] == 0) return -1;
                if (haystack[i + j] != needle[j]) break;
            }
        }
    }
```

written by shichaotan original link here

Solution 2

Well, the problem does not aim for an advanced algorithm like KMP but only a clean brute-force algorithm. So we can traverse all the possible starting points of `haystack` (from `0` to `haystack.length() - needle.length()`) and see if the following characters in `haystack` match those of `needle`.

The code is as follows.

```
class Solution {
public:
    int strStr(string haystack, string needle) {
        int m = haystack.length(), n = needle.length();
        if (!n) return 0;
        for (int i = 0; i < m - n + 1; i++) {
            int j = 0;
            for (; j < n; j++)
                if (haystack[i + j] != needle[j])
                    break;
            if (j == n) return i;
        }
        return -1;
    }
};
```

Of course, you may challenge yourself implementing the KMP algorithm for this problem.

KMP is a classic and yet notoriously hard-to-understand algorithm. However, I think the following two links give nice explanations. You may refer to them.

1. KMP on jBoxer's blog;
2. KMP on geeksforgeeks, with a well-commented C code.

I am sorry that I am still unable to give a personal explanation of the algorithm. I only read it from the two links above and mimic the code in the second link.

My accepted C++ code using KMP is as follows. Well, it also takes 4ms -_-

```
class Solution {
public:
    int strStr(string haystack, string needle) {
        int m = haystack.length(), n = needle.length();
        if (!n) return 0;
        vector<int> lps = kmpProcess(needle);
        for (int i = 0, j = 0; i < m; ) {
            if (haystack[i] == needle[j]) {
                i++;
                j++;
            }
            if (j == n) return i - j;
            if (i < m && haystack[i] != needle[j]) {
                if (j) j = lps[j - 1];
                else i++;
            }
        }
        return -1;
    }
private:
    vector<int> kmpProcess(string& needle) {
        int n = needle.length();
        vector<int> lps(n, 0);
        for (int i = 1, len = 0; i < n; ) {
            if (needle[i] == needle[len])
                lps[i++] = ++len;
            else if (len) len = lps[len - 1];
            else lps[i++] = 0;
        }
        return lps;
    }
};
```

written by jianchao.li.fighter original link here

Solution 3

```
public class Solution {
    public int strStr(String haystack, String needle) {
        int l1 = haystack.length(), l2 = needle.length();
        if (l1 < l2) {
            return -1;
        } else if (l2 == 0) {
            return 0;
        }
        int threshold = l1 - l2;
        for (int i = 0; i <= threshold; ++i) {
            if (haystack.substring(i,i+l2).equals(needle)) {
                return i;
            }
        }
        return -1;
    }
}
```

written by iziang original link here