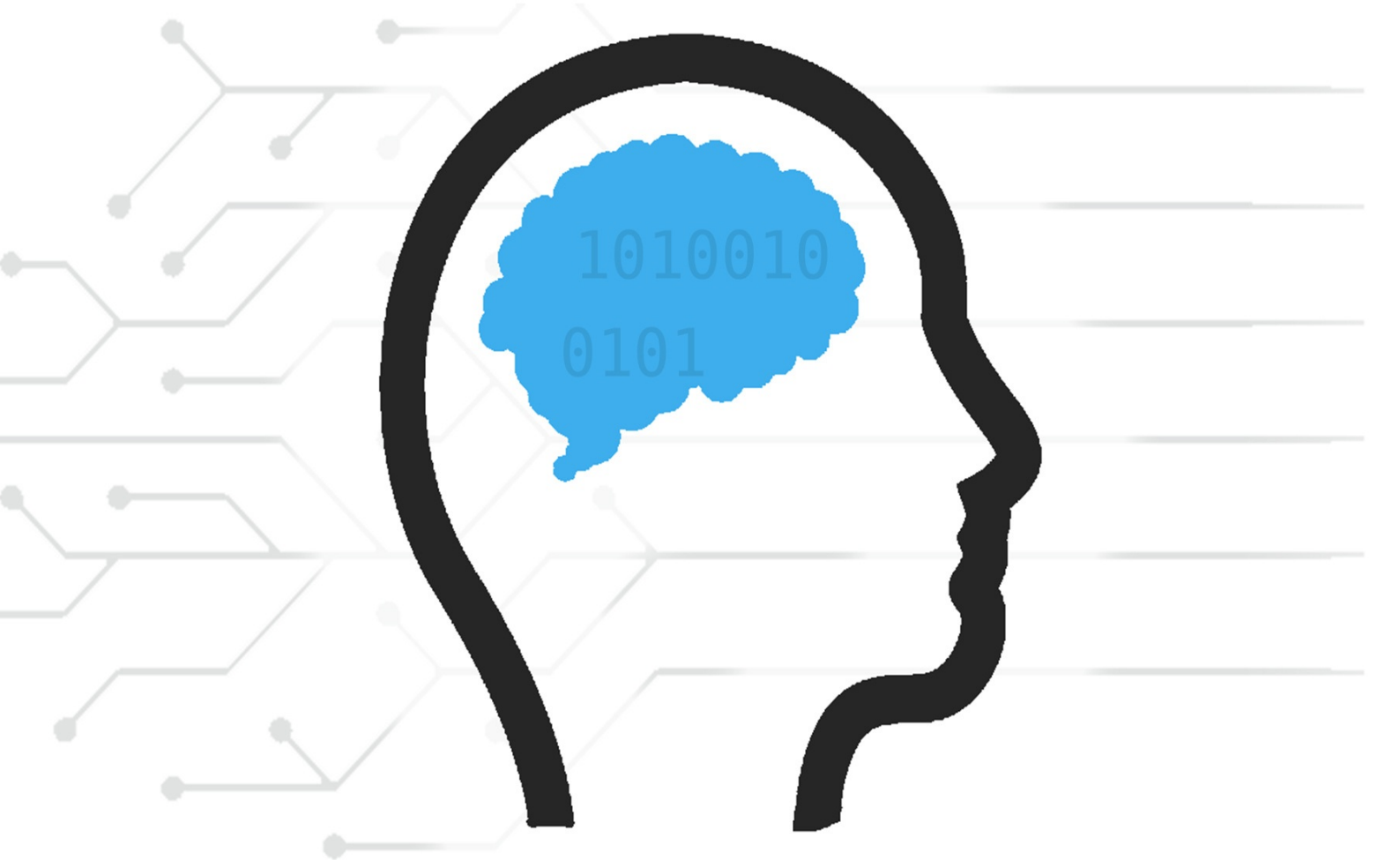

Algorithm

http://algorithm.yuanbin.me

Table of Contents

Data Structure and Algorithm/leetcode/lintcode

- English via Data Structure and Algorithm notes
- 简体中文请戳 数据结构与算法/leetcode/lintcode题解
- 繁體中文請瀏覽 資料結構與演算法/leetcode/lintcode題解

Introduction

This work is some notes of learning and practicing data structures and algorithm.

1. Part I is some brief introduction of basic data structures and algorithm, such as, linked lists, stack, queues, trees, sorting and etc.
2. Part II is the analysis and summary of programming problems, and most of the programming problems come from https://leetcode.com/, http://www.lintcode.com/, http://www.geeksforgeeks.org/, http://hihocoder.com/, https://www.topcoder.com/.
3. Part III is the appendix of resume and other supplements.

This project is hosted on https://github.com/billryan/algorithm-exercise and rendered by Gitbook. You can star the repository on the GitHub to keep track of updates. Another choice is to subscribe channel `#github_commit` via Slack https://ds-algo.slack.com/messages/github_commit/. ~~RSS feed is under development.~~

Feel free to access http://slackin4ds-algo.herokuapp.com for Slack invite automation.

You can view/search this document online or offline, feel free to read it. :)

- Online(Rendered by Gitbook): http://algorithm.yuanbin.me
- Offline(Compiled by Gitbook and Travis-CI):
  1. EPUB: GitHub, Gitbook, GitCafe(mainland China) - Recommended for iPhone/iPad/MAC
  2. PDF: GitHub, Gitbook, GitCafe(mainland China) - Recommended for Desktop
  3. MOBI: GitHub, Gitbook, GitCafe(mainland China) - Recommended for Kindle
- Site Search via Google: `keywords site:algorithm.yuanbin.me`
- Site Search via Swiftype: Click `Search this site` on the right bottom of webpages

License

To Do

- [ ] add multiple languages support, currently 繁體中文, 简体中文 are available
- [x] explore nice writing style
- [x] add implementations of `Python`, `C++`, `Java` code
- [x] add time and space complexity analysis
- [x] summary of basic data structure and algorithm
- [x] add CSS for online website http://algorithm.yuanbin.me
- [x] add proper Chinese fonts for PDF output

FAQ - Frequently Asked Question

Some guidelines for contributing and other questions are listed here.

How to Contribute?

- Access Guidelines for Contributing for details.

Guidelines for Contributing

- Access English via Guidelines for Contributing
- 繁體中文請移步 貢獻指南
- 简体中文请移步 贡献指南

Contributors

Contributors are listed here.

Maintainer

- English is maintained by @billryan, @niangaotuantuan
- 简体中文 is maintained by @billryan, @Shaunwei, @niangaotuantuan
- 繁體中文 is maintained by @CrossLuna

You can find other contributors in Contributors to billryan/algorithm-exercise.

Donation

For privacy protection, some personal information are omitted.

- `taoli**@gmail.com`, 支付宝转账
- `张亚*`, 支付宝转账
- `俞卓*`, 支付宝转账
- `季*`, 支付宝转账

Part I - Basics

Data Structure

String

String related topics are discussed in this chapter.

> In order to re-use most of the memory of an existing data structure, internal implementation of string is immutable in most programming languages(Java, Python). Take care if you want to modify character in place.

strStr

Question

- leetcode: Implement strStr() | LeetCode OJ
- lintcode: lintcode - (13) strstr

Problem Statement

For a given source string and a target string, you should output the **first** index(from 0) of target string in source string.

If target does not exist in source, just return `-1`.

Example

If source = `"source"` and target = `"target"`, return `-1`.

If source = `"abcdabcdefg"` and target = `"bcd"`, return `1`.

Challenge

O(n2) is acceptable. Can you implement an O(n) algorithm? (hint: *KMP*)

Clarification

Do I need to implement KMP Algorithm in a real interview?

- Not necessary. When you meet this problem in a real interview, the interviewer may just want to test your basic implementation ability. But make sure your confirm with the interviewer first.

Problem Analysis

It's very straightforward to solve string match problem with nested for loops. Since we must iterate the target string, we can optimize the iteration of source string. It's unnecessary to iterate the source string if the length of remaining part does not exceed the length of target string. We can only iterate the valid part of source string. Apart from this naive algorithm, you can use a more effective algorithm such as KMP.

Python

```python
class Solution:
    def strStr(self, source, target):
        if source is None or target is None:
            return -1

        for i in range(len(source) - len(target) + 1):
            for j in range(len(target)):
                if source[i + j] != target[j]:
                    break
            else:  # no break
                return i
        return -1
```

C

```c
int strStr(char* haystack, char* needle) {
    if (haystack == NULL || needle == NULL) return -1;

    const int len_h = strlen(haystack);
    const int len_n = strlen(needle);
    for (int i = 0; i < len_h - len_n + 1; i++) {
        int j = 0;
        for (; j < len_n; j++) {
            if (haystack[i+j] != needle[j]) {
                break;
            }
        }
        if (j == len_n) return i;
    }

    return -1;
}
```

C++

```cpp
class Solution {
public:
    int strStr(string haystack, string needle) {
        if (haystack.empty() && needle.empty()) return 0;
        if (haystack.empty()) return -1;
        if (needle.empty()) return 0;
        // in case of overflow for negative
        if (haystack.size() < needle.size()) return -1;

        for (int i = 0; i < haystack.size() - needle.size() + 1; i++) {
            string::size_type j = 0;
            for (; j < needle.size(); j++) {
                if (haystack[i + j] != needle[j]) break;
            }
            if (j == needle.size()) return i;
        }

        return -1;
    }
};
```

Java

```java
public class Solution {
    public int strStr(String haystack, String needle) {
        if (haystack == null && needle == null) return 0;
        if (haystack == null) return -1;
        if (needle == null) return 0;

        for (int i = 0; i < haystack.length() - needle.length() + 1; i++) {
            int j = 0;
            for (; j < needle.length(); j++) {
                if (haystack.charAt(i+j) != needle.charAt(j)) break;
            }
            if (j == needle.length()) return i;
        }

        return -1;
    }
}
```

Source Code Analysis

1. corner case: `haystack(source)` and `needle(target)` may be empty string.

2. code convention:
    - space is needed for `==`
    - use meaningful variable names
    - put a blank line before declaration `int i, j;`
3. declare j outside for loop if and only if you want to use it outside.

Some Pythonic notes: 4. More Control Flow Tools section 4.4 and if statement - Why does python use 'else' after for and while loops?

Complexity Analysis

nested for loop, $O((n-m)m)$ for worst case.