28. Implement strstr() 2 Dec. 9, 2019 | 40.3K views

Average Rating: 3.67 (30 votes)

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

Example 1:

Implement strStr().

Input: haystack = "hello", needle = "ll" Output: 2

Example 2: Input: haystack = "aaaaa", needle = "bba"

Output: -1

Clarification: What should we return when needle is an empty string? This is a great question to ask during an interview.

Solution

The problem is to find needle of length L in the haystack of length N. Let's discuss three different ideas how to proceed. They are all based on sliding window idea. The key point is how to implement a window slice.

Overview

Linear time window slice $\mathcal{O}(L)$ is quite easy, move the window of length L along the haystack and compare substring in the window with the needle. Overall that would result in $\mathcal{O}((N-L)L)$ time complexity.

comparison happens not for all substrings, and that improves the best time complexity up to $\mathcal{O}(N)$. The worst time complexity is still $\mathcal{O}((N-L)L)$ though.

Approach 1: Substring: Linear Time Slice

Python

class Solution:

Java

• Rabin-Karp = constant-time slice using rolling hash algorithm.

Could that be improved to $\mathcal{O}(N)$? Yes, but one has to implement constant time slice $\mathcal{O}(1)$. There are two ways to do it:

Could that be improved? Yes. Two pointers approach is still the case of linear time slice, though the

• Bit manipulation = constant-time slice using bitmasks. Bit Manipulation approach in Java is more suitable for the short strings or strings with very limited number of

no such a problem). Here we deal with quite long strings and it's more simple to implement the basic version of Rabin Karp algorithm.

there is a match --> return 1 needle Implementation

L, n = len(needle), len(haystack) 3 for start in range(n - L + 1): if haystack[start: start + L] == needle: 6 7 return start return -1 8 **Complexity Analysis**

haystack

Algorithm

Implementation

Java

9

16 17

18

19

20 21

22

23

24

25 26

27

There are two technical problems:

String slice in a constant time

1. How to implement a string slice in a constant time?

2. How to generate substring hash in a constant time?

string to another data structure, for example, to integer array of ascii-values.

Python

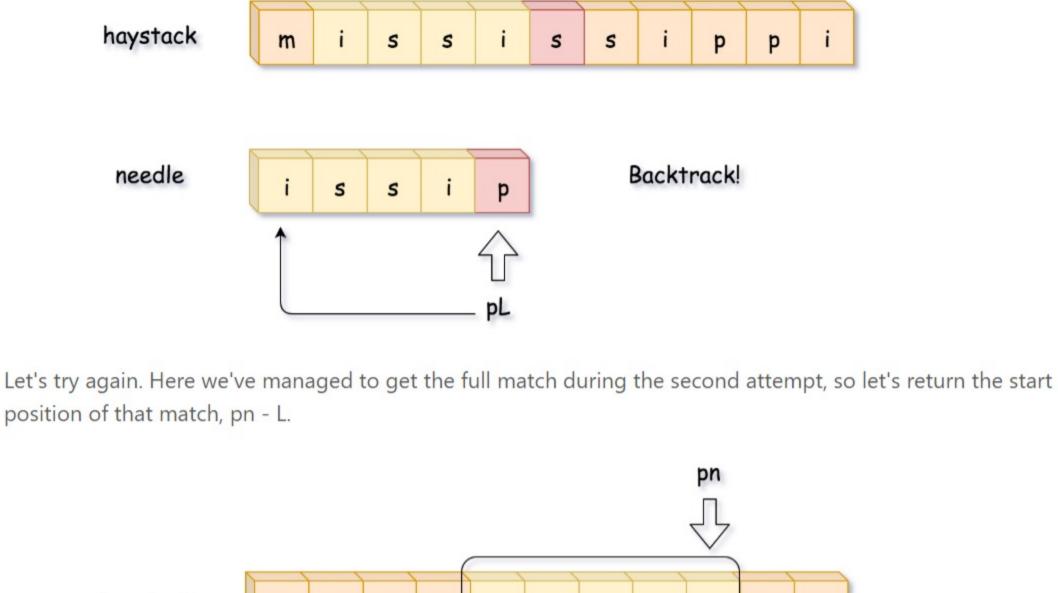
m

- needle
- needle

Here it was impossible to manage the full match up to the length of needle string, which is L = 5. Let's

backtrack then. Note, that we move pn pointer back to the position pn = pn - curr_len + 1, and not to the

pn

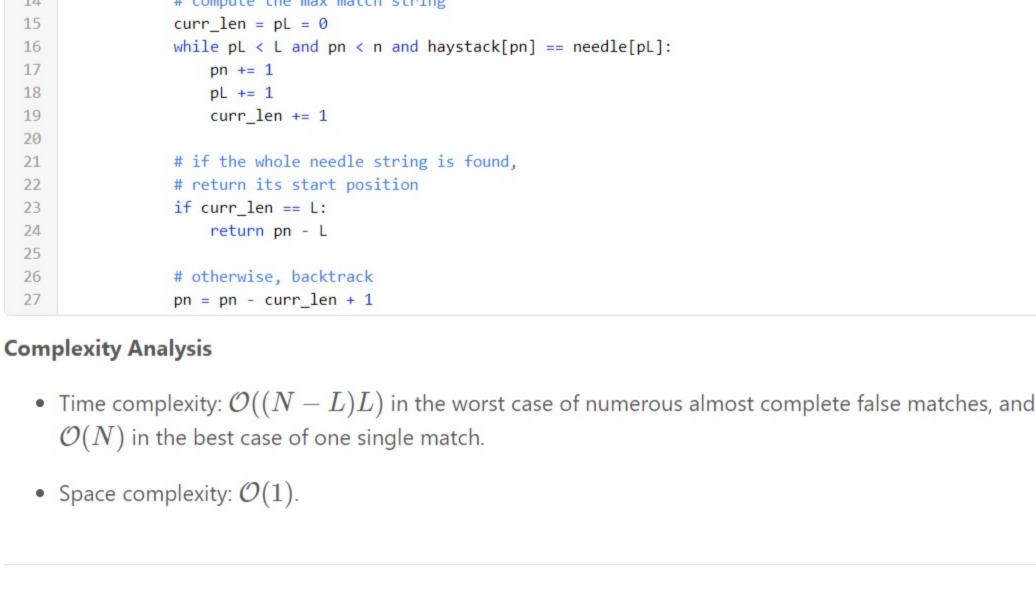


class Solution: def strStr(self, haystack: str, needle: str) -> int: L, n = len(needle), len(haystack) 3 4 if L == 0:

Сору

• Compute the max string match by increasing pn, pL and curr_len in the case of equal characters.

• If you managed to get the full match, curr_len == L, return the start position of that match, pn - L.



 a^L could be a large number and hence the idea is to set limits to avoid the overflow. To set limits means to limit a hash by a given number called modulus and use everywhere not hash itself but h % modulus. It's quite obvious that modulus should be large enough, but how large? Here one could read more about the

topic, for the problem here 2^{31} is enough.

needle.substring(0, L).

return 0

aL = pow(a, L, modulus)

for start in range(1, n - L + 1):

22 23

24 25

26 27

Complexity Analysis

O Previous

Comments: 26

• Space complexity: $\mathcal{O}(1)$.

Rate this article: * * * * *

Preview

to remove number 0 and to add number 4.

In a generalised way

overflow problem.

Algorithm

How to avoid overflow

• Return start position if the hash is equal to the reference one. Return -1, that means that needle is not found. Implementation

• Iterate over the start position of possible match: from 1 to N - L.

Compute rolling hash based on the previous hash value.

15 16 # compute the hash of strings haystack[:L], needle[:L] $h = ref_h = 0$ 17 for i in range(L): 18 $h = (h * a + h_{to_int(i)}) % modulus$ 19 20 ref_h = (ref_h * a + needle_to_int(i)) % modulus if h == ref_h: 21

const value to be used often : a**L % modulus

then runs a loop of (N-L) steps with constant time operations in it.

compute rolling hash in O(1) time

seandot 🛊 33 🕗 February 6, 2020 12:28 AM my solution: return haystack.indexOf(needle)

Solution 2 with a more elegant implementation in Python 3 if you need: def strStr(self, haystack: str, needle: str) -> int: if not needle: return 0 Read More 3 A V C Share Reply anaken 🛊 2 🗿 June 22, 2020 12:23 AM Abdul Bari's Explanation on Rabin Karp https://youtu.be/qQ8vS2btsxI

what about collisions in approach 3?

1 A V C Share Reply

SHOW 2 REPLIES Isheng_mel ★ 167 ② March 7, 2020 8:34 AM @andvary can you explain why the hash of the needle is calculated below:

(123)

For the purpose of this problem, we will return 0 when needle is an empty string. This is consistent to C's strstr() and Java's indexOf().

Quite straightforward approach - move sliding window along the string and compare substring in the window with the needle. haystack † d e e C 0 e

ullet Time complexity: $\mathcal{O}((N-L)L)$, where N is a length of haystack and L is a length of needle. We compute a substring of length L in a loop, which is executed (N - L) times. • Space complexity: $\mathcal{O}(1)$. Approach 2: Two Pointers: Linear Time Slice Drawback of the previous algorithm is that one compares absolutely all substrings of length L with the needle. There is no need to that. First, let's compare only substrings which starts from the first character in the needle substring.

Second, let's compare the characters one by one and stop immediately in the case of mismatch. pn

haystack S S S p p

5 return 0 6 7 pn = 08 while pn < n - L + 1:

Rolling hash: hash generation in a constant time To generate hash of array of length L, one needs $\mathcal{O}(L)$ time. How to have constant time of hash generation? Use the advantage of slice: only one integer in, and only one - out. That's the idea of rolling hash. Here we'll implement the simplest one, polynomial rolling hash. Beware that's polynomial rolling hash is NOT the Rabin fingerprint. Since one deals here with lowercase English letters, all values in the integer array are between 0 and 25: arr[i] = (int)S.charAt(i) - (int)'a'. So one could consider string abcd -> [0, 1, 2, 3] as a number in a numeral system with the base 26. Hence abcd -> [0, 1, 2, 3] could be hashed as $h_0 = 0 \times 26^3 + 1 \times 26^2 + 2 \times 26^1 + 3 \times 26^0$ Let's write the same formula in a generalised way, where c_i is an integer array element and a=26 is a system base. $h_0 = c_0 a^{L-1} + c_1 a^{L-2} + ... + c_i a^{L-1-i} + ... + c_{L-1} a^1 + c_L a^0$ $h_0 = \sum_{i=0}^{L-1} c_i a^{L-1-i}$

Now let's consider the slice abcd -> bcde . For int arrays that means [0, 1, 2, 3] -> [1, 2, 3, 4],

 $h_1 = (h_0 - 0 \times 26^3) \times 26 + 4 \times 26^0$

 $h_1 = (h_0 a - c_0 a^L) + c_{L+1}$

Now hash regeneration is perfect and fits in a constant time. There is one more issue to address: possible

• Compute the hash of substring haystack.substring(0, L) and reference hash of

Strings are immutable in Java and Python, and to move sliding window in a constant time one has to convert

Сору Python Java class Solution: 1 2 def strStr(self, haystack: str, needle: str) -> int: L, n = len(needle), len(haystack) 3 4 if L > n: 5 return -1 6 # base value for the rolling hash function 7 8 # modulus value for the rolling hash function to avoid overflow 9 modulus = 2**3110

They managed to make solution 2 look like rocket science in this article. 68 ∧ ∨ ☑ Share ¬ Reply SHOW 2 REPLIES

Type comment here... (Markdown is supported)

d_darric ★ 78 ② January 16, 2020 5:46 PM

gfmacode ★ 30 ② February 25, 2020 10:47 PM

Rialakshmi ★ 94 ② February 23, 2020 4:07 AM

SHOW 2 REPLIES

hash collision that would give us a false positive?

you write will be maintained and supported by some other people. They will have hard time understanding your best solution. Read More wangr 🛊 3 🗿 May 4, 2020 8:58 PM One example of hash collision for Solution 3: "gytisyz" "aaaaaab" 3 A V C Share Reply wil30303 🛊 2 🗿 March 1, 2020 4:40 AM i can't pass the test case "a", "a". it should return zero but it is not. I followed solution 1 2 A V C Share Reply

Read More 1 A V C Share Reply

> ref h = (ref h * a + charToInt(i. needle)) % modulus: Read More

h = (h * a + charToInt(i, haystack)) % modulus;

characters, ex. Repeated DNA Sequences. That's a consequence of overflow issues in Java (in Python there is

Сору

2 def strStr(self, haystack: str, needle: str) -> int: 4 5

haystack

curr_len = 4 i S S p

S

S

S

S

position pn = pn - curr_len, since this last one was already investigated.

needle needle is found

• Move pn till you'll find the first character of the needle string in the haystack.

• If you didn't, backtrack: pn = pn - curr_len + 1, pL = 0, curr_len = 0.

find the position of the first needle character

10 # in the haystack string 11 while pn < n - L + 1 and haystack[pn] != needle[0]:</pre> 12 pn += 113 # compute the max match string 14 15

Approach 3: Rabin Karp: Constant Time Slice Let's now design the algorithm with $\mathcal{O}(N)$ time complexity even in the worst case. The idea is simple: move along the string, generate hash of substring in the sliding window and compare it with the reference hash of the needle string.

11 12 # lambda-function to convert character to integer h_to_int = lambda i : ord(haystack[i]) - ord('a') 13 needle_to_int = lambda i : ord(needle[i]) - ord('a') 14

• Time complexity: $\mathcal{O}(N)$, one computes the reference hash of the needle string in $\mathcal{O}(L)$ time, and

Next 👀

Sort By -

Post

SHOW 4 REPLIES

@andvary when computing hash for haystack.substring(0, L) how do we guarantee there is no

@liaison and @andvary The solution is very good except the base could be 256 to cover all character

set. Please avoid using a,b,c,d for variable names, give some meaningful name. In real projects the code

SHOW 2 REPLIES nelapsi 🖈 11 🗿 February 25, 2020 4:45 AM

walrussgt ★7 ② February 18, 2020 12:41 PM

1 A V C Share Reply **SHOW 2 REPLIES**

for (int i = 0; i < L; ++i) {