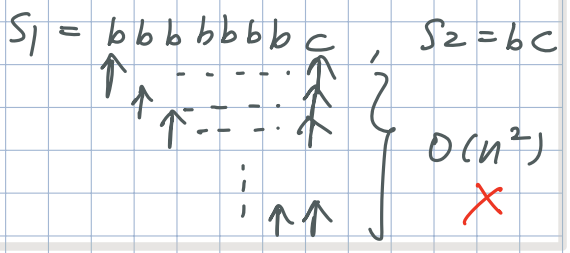
output

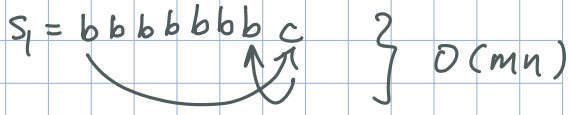
s1 = abcdebdde

s2 = bde

Intuition : Brute-force sort



⇒ Alternative :



Def :

- S_1 (string) : big string
- S_2 (string) : substring

char array (list)

⇒ S_1 = ["a", "b", ---- "e"]

⇒ S_2 = ["b", "d", "e"]

- i
- j

(arrows for tracking current indices) : for S_1 / for S_2

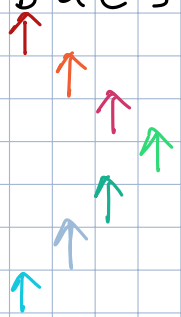
- start [default=-1], end
- min [default = len(S_1)]

return { empty str if start=-1; substring otw

while $i < len(S_1)$,

$S_1 = [a\ b\ c\ d\ e\ b\ d\ d\ e]$, $len\_s_1 = 9$

$S_2 = [b\ d\ e]$, $len\_S_2 = 3$



$i += 1$

$i = 0$ 第3到起点

$i = 1$, $S_1[1] = S_2[0]$

$i = 2$, $j = 1$

$i = 3$, $S_1[3] = S_2[1]$

$i = 4$, $S_1[4] = S_2[2]$

$end = i+1 = 4+1 = 5$

$i = 4 \Rightarrow j -= 1$

$i = 3 \Rightarrow j -= 1$

$i = 2$ ∵ j negative

$i = 1 \Rightarrow$ stop turning

$end - i = 5 - 1 = 4$

$< min$,

set $min = end - i$,

start $= i$

$j = 0$

$j = 1 \neq len\_S_2$

$j = 2 \neq len\_S_2$

$j = 3 = len\_S_2$

j 往回走, j=2 定位终点

$j = 1$

$j = 0$

return $\begin{cases} "" & \text{if start} == -1 \text{ (没有匹配到)} \\ S[start, start+min] & \text{otherwise} \end{cases}$

Minimum Window Subsequence (/problems/minimum-window-subsequence/)

Submission Detail

66 / 67 test cases passed.

Status: Wrong Answer

Submitted: **0 minutes ago**

Input: **"aa"**
"aa"

Output: ""

Expected: "aa"

Submitted Code: 0 minutes ago

Language: python

Edit Code

```python
class Solution(object):
    def minWindow(self, s1, s2):
        """
        :type s1: str
        :type s2: str
        :rtype: str
        """
        start = -1
        min_len = len(s1) + 1

        i, j = 0, 0
        while(i < len(s1)):
            # aligning beginning
            if(s1[i] == s2[j]): # start is found
                j += 1
                if(j == len(s2)): # end is found
                    end = i + 1 # mark the end
                    j -= 1 # start turning back
                    while(j >= 0):
                        if(s1[i] == s2[j]):
                            j -= 1 # once matched an element, turn further back
                        i -= 1
                    # now, i is pointing to one-char front of s1 and j is negative
                    i += 1 # to point to the start of s1
                    j += 1 # to point to the start of s2
                    if(end - i < min_len):
                        min_len = end - i
                        start = i
            # iterate through s1
            i += 1

        returned_str = ""
        if(start != -1):
            returned_str = s1[start: start+min_len]

        return returned_str
```

← 沒有成功進來，因為 end-i = 2-0 = min_len

Back to problem (/problems/minimum-window-subsequence/)

727. Minimum Window Subsequence

Hard | 1159 | 69 | Add to List | Share

Given strings `s1` and `s2`, return *the minimum contiguous substring part of* `s1`, *so that* `s2` *is a subsequence of the part*.

If there is no such window in `s1` that covers all characters in `s2`, return the empty string `""`. If there are multiple such minimum-length windows, return the one with the **left-most starting index**.

Example 1:

```
Input: s1 = "abcdebdde", s2 = "bde"
Output: "bcde"
Explanation:
"bcde" is the answer because it occurs before "bdde" which has the same length.
"deb" is not a smaller window because the elements of s2 in the window must occur in order.
```

Example 2:

```
Input: s1 = "jmeqksfrsdcmsiwvaovztaqenprpvnbstl", s2 = "u"
Output: ""
```

Constraints:

- `1 <= s1.length <= 2 * 10^4`
- `1 <= s2.length <= 100`
- `s1` and `s2` consist of lowercase English letters.

Accepted 72,481 Submissions 169,053

Seen this question in a real interview before?

Yes No

Companies

Related Topics

Similar Questions

Show Hint 1

```python
class Solution(object):
    def minWindow(self, s1, s2):
        """
        :type s1: str
        :type s2: str
        :rtype: str
        """
        start = -1
        min_len = len(s1)+1

        i, j = 0, 0
        while(i < len(s1)):
            # aligning beginning
            if(s1[i] == s2[j]): # start is found
                j += 1
                if(j == len(s2)): # end is found
                    end = i + 1 # mark the end
                    j -= 1 # start turning back
                    while(j >= 0):
                        if(s1[i] == s2[j]):
                            j -= 1 # once matched an element, turn further back
                        i -= 1
                    # now, i is pointing to one-char front of s1 and j is negative
                    i += 1 # to point to the start of s1
                    j += 1 # to point to the start of s2
                    if(end - i < min_len):
                        min_len = end - i
                        start = i
            # iterate through s1
            i += 1

        returned_str = ""
        if(start != -1):
            returned_str = s1[start: start+min_len]
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 33 ms

Your input
"abcdebdde"
"bde"

Output "bcde" Diff

Expected "bcde"

Console | Use Example Testcases | Run Code | Submit

Minimum Window Subsequence (/problems/minimum-window-subsequence/)

Submission Detail

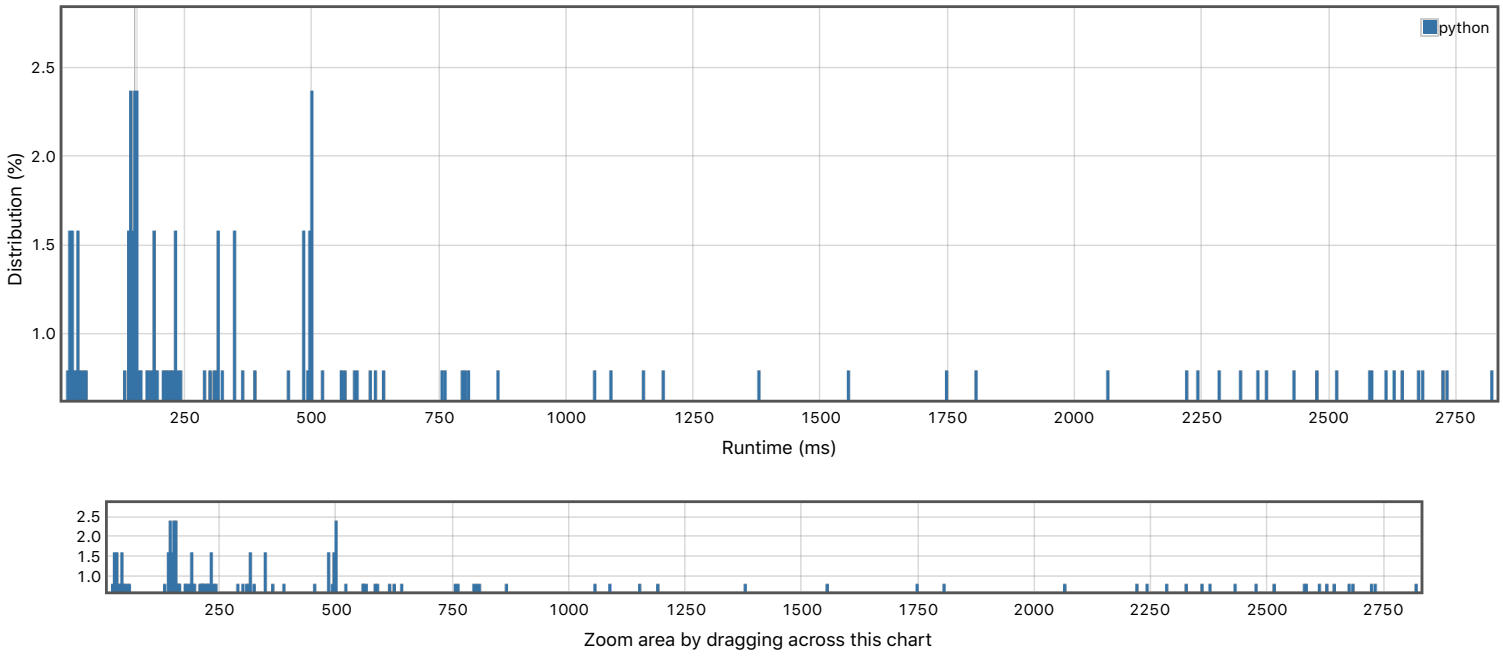
67 / 67 test cases passed.

Status: Accepted

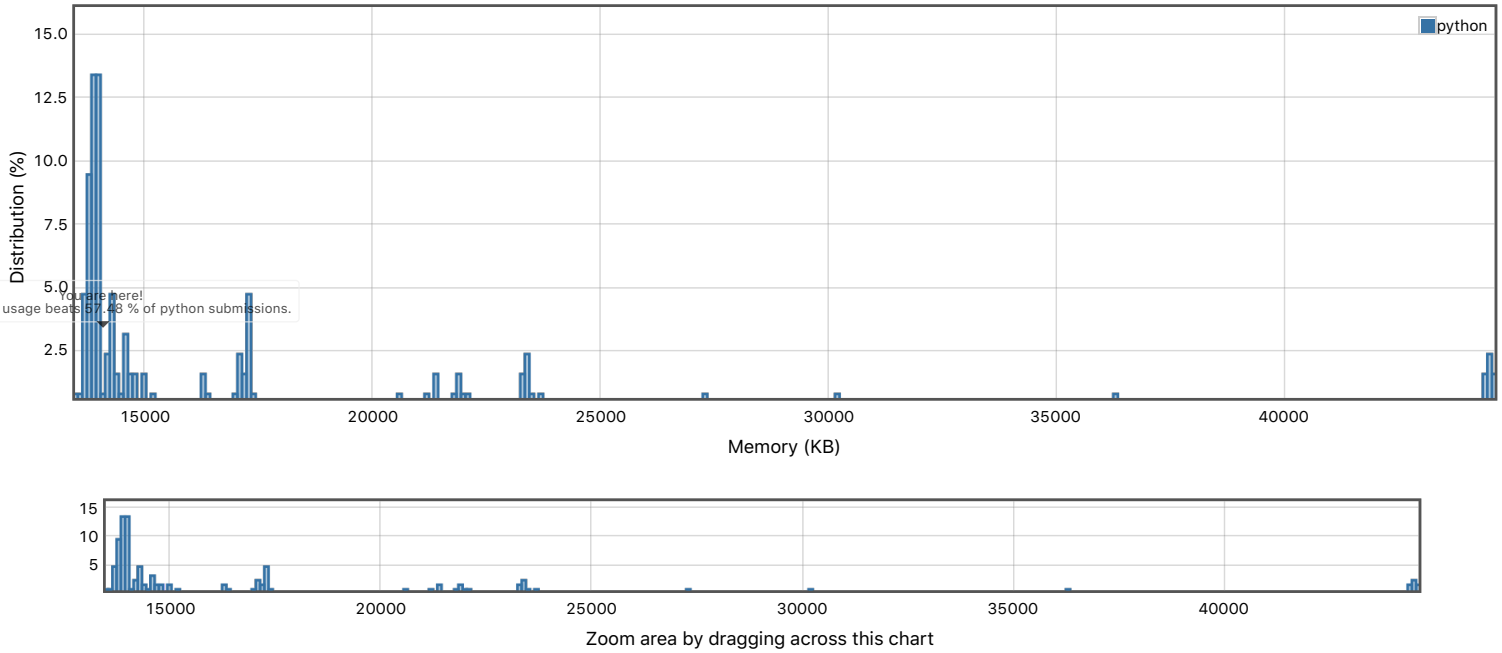
Runtime: **156 ms**
Memory Usage: **14 MB**

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Minimum Window Subsequence**

Submitted Code: 0 minutes ago

Language: python

Edit Code

```python
class Solution(object):
    def minWindow(self, s1, s2):
        """
        :type s1: str
        :type s2: str
        :rtype: str
        """
        start = -1
        min_len = len(s1)+1

        i, j = 0, 0
        while(i < len(s1)):
            # aligning beginning
            if(s1[i] == s2[j]): # start is found
                j += 1
                if(j == len(s2)): # end is found
                    end = i + 1 # mark the end
                    j -= 1 # start turning back
                    while(j >= 0):
                        if(s1[i] == s2[j]):
                            j -= 1 # once matched an element, turn further back
                        i -= 1
                    # now, i is pointing to one-char front of s1 and j is negative
                    i += 1 # to point to the start of s1
                    j += 1 # to point to the start of s2
                    if(end - i < min_len):
                        min_len = end - i
                        start = i
            # iterate through s1
            i += 1

        returned_str = ""
        if(start != -1):
            returned_str = s1[start: start+min_len]

        return returned_str
```

Back to problem (/problems/minimum-window-subsequence/)