Back

Python O(M + N*logM) using inverted index + binary search (Similar to LC 792)

Iwahu ★ 386 Last Edit: June 30, 2019 12:53 AM 4.5K VIEWS

78

This is a google phone screen question. It's the same idea as 792. Number of Matching Subsequences. I recommend solving that question first.

The idea is to create an inverted index that saves the offsets of where each character occurs in `source`. The index data structure is represented as a hashmap, where the Key is the character, and the Value is the (sorted) list of offsets where this character appears. To run the algorithm, for each character in `target`, use the index to get the list of possible offsets for this character. Then search this list for next offset which appears after the offset of the previous character. We can use binary search to efficiently search for the next offset in our index.

Example with `source = "abcab", target = "aabbaac"`
The inverted index data structure for this example would be:
inverted_index = {
a: [0, 3] # 'a' appears at index 0, 3 in source
b: [1, 4], # 'b' appears at index 1, 4 in source
c: [2], # 'c' appears at index 2 in source
}
Initialize `i = -1` (i represents the smallest valid next offset) and `loop_cnt = 1` (number of passes through source).
Iterate through the target string "aabbaac"
a => get the offsets of character 'a' which is [0, 3]. Set `i` to 1.
a => get the offsets of character 'a' which is [0, 3]. Set `i` to 4.
b => get the offsets of character 'b' which is [1, 4]. Set `i` to 5.
b => get the offsets of character 'b' which is [1, 4]. Increment `loop_cnt` to 2, and Set `i` to 2.
a => get the offsets of character 'a' which is [0, 3]. Set `i` to 4.
a => get the offsets of character 'a' which is [0, 3]. Increment `loop_cnt` to 3, and Set `i` to 1.
c => get the offsets of character 'c' which is [2]. Set `i` to 3.
We're done iterating through target so return the number of loops (3).

The runtime is O(M) to build the index, and O(logM) for each query. There are N queries, so the total runtime is O(M + N*logM). M is the length of source and N is the length of target. The space complexity is O(M), which is the space needed to store the index.

```
::python
def shortestWay(self, source: str, target: str) -> int:
    inverted_index = collections.defaultdict(list)
    for i, ch in enumerate(source):
        inverted_index[ch].append(i)

    loop_cnt = 1
    i = -1
    for ch in target:
        if ch not in inverted_index:
            return -1
        offset_list_for_ch = inverted_index[ch]
        # bisect_left(A, x) returns the smallest index j s.t. A[j] >= x. If no such index j exists, it returns len(A).
        j = bisect.bisect_left(offset_list_for_ch, i)
        if j == len(offset_list_for_ch):
            loop_cnt += 1
            i = offset_list_for_ch[0] + 1
        else:
            i = offset_list_for_ch[j] + 1

    return loop_cnt
```

Closing thoughts: As usual, there are multiple solutions to this problem. One way other way to solve is the naive O(M*N) solution using two pointers. This solution won't be enough to pass a phone screen. There is also a dynamic programming O(|Σ|*M + N) solution (where |Σ| is the size of the alphabet, or 26 in this question). xiangmo posted an example solution here, which I encourage you to check out.

Comments: 9 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)

Post

RadishBu ★ 21 June 4, 2019 12:19 PM

Java Version:

```
public int shortestWay(String source, String target) {
    Map<Character, List<Integer>> map = new HashMap<>();
    for(int i = 0; i<source.length(); i++){
        char cs = source.charAt(i);
        if(!map.containsKey(cs)){
            map.put(cs, new ArrayList<Integer>());
        }
        map.get(cs).add(i);
    }
```

Read More

6 Reply

kevinhynes ★ 213 June 15, 2019 12:23 AM

Great post, thank you for this. Clear, concise explanation with quality code and analysis. Have an internet high five!

3 Reply

Dirk41 ★ 175 March 19, 2020 5:48 PM

Your explanation is very clear, but if you use index instead of offset, it would be more clear. Thank you very much for this work !!!!!

2 Reply

nix_on ★ 43 November 25, 2019 2:38 AM

Why do you use the word "offset" instead of "index" in this case? I mean the inverted index is a dictionary of lists of indexes, don't understand why you would call them offsets.

2 Reply

tohrah ★ 807 June 26, 2019 2:55 AM

```
class Solution {
    public int shortestWay(String source, String target) {
        if(source==null || target == null) {
            return -1;
        }
        Map<Character, List<Integer>> store = new HashMap();
        int idx = 0;
        for(char ch : source.toCharArray()){
            List<Integer> lis = store.getOrDefault(ch, new ArrayList());
            lis.add(idx);
```

Read More

1 Reply

xiangmo ★ 104 June 11, 2019 10:51 AM

Java O(M + N) solution: https://leetcode.com/problems/shortest-way-to-form-string/discuss/309632/Java-O(M-+-N)-solution

1 Reply