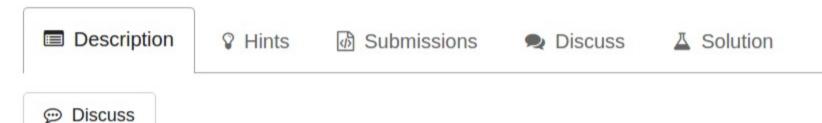
Pick One



Design a search autocomplete system for a search engine. Users may input a sentence (at least one word and end with a special character "#"). For **each character** they type **except** '#', you need to return the **top 3** historical hot sentences that have prefix the same as the part of sentence already typed. Here are the specific rules:

- 1. The hot degree for a sentence is defined as the number of times a user typed the exactly same sentence before.
- 2. The returned top 3 hot sentences should be sorted by hot degree (The first is the hottest one). If several sentences have the same degree of hot, you need to use ASCII-code order (smaller one appears first).
- 3. If less than 3 hot sentences exist, then just return as many as you can.
- 4. When the input is a special character, it means the sentence ends, and in this case, you need to return an empty list.

Your job is to implement the following functions:

The constructor function:

AutocompleteSystem(String[] sentences, int[] times): This is the constructor. The input is **historical data**. Sentences is a string array consists of previously typed sentences. Times is the corresponding times a sentence has been typed. Your system should record these historical data.

Now, the user wants to input a new sentence. The following function will provide the next character the user types:

List<String> input(char c): The input c is the next character typed by the user. The character will only be lower-case letters ('a' to 'z'), blank space ('') or a special character ('#'). Also, the previously typed sentence should be recorded in your system. The output will be the **top 3** historical hot sentences that have prefix the same as the part of sentence already typed.

## Example:

Operation: AutocompleteSystem(["i love you", "island", "ironman", "i love leetcode"], [5,3,2,2])

The system have already tracked down the following sentences and their corresponding times:

"i love you" : 5 times
"island" : 3 times
"ironman" : 2 times
"i love leetcode" : 2 times

Now, the user begins another search:

Operation: input('i')

Output: ["i love you", "island","i love leetcode"]

**Explanation:** 

There are four sentences that have prefix "i". Among them, "ironman" and "i love leetcode" have same hot degree. Since '' has ASCII code 32 and 'r' has ASCII code 114, "i love leetcode" should be in front of "ironman". Also we only need to output top 3 hot sentences, so "ironman" will be ignored.

Operation: input(' ')

Output: ["i love you","i love leetcode"]

Explanation:

There are only two sentences that have prefix "i".

Operation: input('a')

Output: [] Explanation:

There are no sentences that have prefix "i a".

Operation: input('#')

Output: [] Explanation:

The user finished the input, the sentence "i a" should be saved as a historical sentence in system. And the following input will be counted as a new search.

## Note:

- 1. The input sentence will always start with a letter and end with '#', and only one blank space will exist between two words.
- 2. The number of **complete sentences** that to be searched won't exceed 100. The length of each sentence including those in the historical data won't exceed 100.
- 3. Please use double-quote instead of single-quote when you write test cases even for a character input.
- 4. Please remember to RESET your class variables declared in class AutocompleteSystem, as static/class variables are persisted across multiple test cases. Please see here for more details.

Difficulty:

Total Accepted:

256

Total Submissions:

1.1K

Contributor:

fallcreek

**Subscribe** to see which companies asked this question.

Related Topics ▼

Similar Questions ▼