Bonus Question

June 12, 2023

Important Notes

- Doubt Document
- Plagiarism Policy. Please read this before you start the assignment
- Usage of C++ is **NOT** allowed for this assignment.

1 Target Word Retrieval in Pictionary Games

1.1 Introduction to Pictionary

Pictionary is a classic drawing and guessing game played with two or more people. One player chooses a word from a list and then tries to draw it without using any words or letters. The other players try to guess the word by looking at the drawing. The first player to guess the word correctly wins the round. We call the chosen word a target word. And the guesses as guess words.

A typical game session records the following data:

Data	**Description**
Target Word	The word that is drawn to get the guesser to guess.
Guess Words	The words that the guesser guessed during the game.

Example: The following is an example of a game session:

- Target word: `bird`
- Guess words: [`crow` , `fly` , `bird`]

1.2 Problem Context

Guess Retrieval for Pictionary is an auto-complete-like feature on top of the game that helps retrieve the target words for the guesses made and vice-versa.

For this problem, we would like you to design a feature similar to this, which returns target words given the guess word or prefix of guess words.

The below URL contains videos of some sessions which illustrate the game and guess. - http://drawm0n.github.io/

1.3 Question

You will be given a set of target words and corresponding guesses for various game sessions you played. Your task is to design a data structure that stores all guess words and target words in a way such that target words can be listed given the guess word or prefix of guess words efficiently.

1.4 Input Format

Input will consist of multiple lines.

- The first line will contain the number of sessions, **S**.
- The subsequent **S** lines will contain
  - The Number of Guesses n_i
  - T_i The Target Word (no spaces within the word), and
  - $G_{(i,1)}$,$G_{(i,2)}$ ... $G_{(i,n_i)}$ (n_i space-separated list of guess words)
  - **Example:** `2 hello hi Hello`
- The 2^{nd} last line will contain the number of queries M for which target words have to be printed for. (a query is a guess word or prefix of guess words)
- The last line will contain a list of queries (a query is a guess word or prefix of guess words) for which target words have to be predicted.
- All letters in target and guess words are in lowercase.
- A word doesn't contain space in it.
- **A word is a prefix of itself**.
- **So you can consider all queries as a prefix of guess words and print all target words associated with guess words with the given prefix**.

1.5 Output Format

For each query, the program should print the following in a new line:

- The number of target words associated with each query (a query is a guess word or prefix of guess words).
- The associated list of target words in **sorted order**

1.6 Constraints

$0 \leq S \leq 100$

$0 \leq length(G_{j,i}) + \ length(T_i) \leq 2000000$. (Sum of the length of all guess words and target words across all sessions is less than 2000000).

$0 \leq \Sigma n_i \leq 100000$

$0 \leq M \leq 10000$

The sum of the length of all queries is less than $4 * 10^5$. (query is a guess word or prefix of guess words).

Example 1

Input

```
3
2 cat animal cat
3 dog animal cat dog
5 house box hut village home house
3
animal cat village
```

Output

```
2 cat dog
2 cat dog
1 house
```

Example 2

Input

```
3
2 cat animal cat
3 dog animal cat dog
3 bird crow fly bird
3
an c dog
```

Output

```
2 cat dog
2 bird cat dog
1 dog
```

1.7 Explanation

1.7.1 Example 1

Input:

- In this example, we have three sessions.
- In the first session, the target word is `cat`, and the guess words are `animal` and `cat`.
- In the second session, the target word is `dog` and the guess words are `animal`, `cat`, and `dog`.
- In the third session, the target word is `house` and the guess words are `box`, `hut`, `village`, and `home`.
- The next line `3` represents the number of guess words or prefixes of guess words we are interested in:
- `animal`, `cat`, and `village` are the queries for which we need to retrieve all target words associated with these guess words or prefixes of the guess word.

Output:

- For the query `animal`:
  - The associated target words are `cat` and `dog`.
  - so the output is `2 cat dog`.
- For the query `cat`:
  - The associated target words are `cat` and `dog`.
  - So the output is `2 cat dog`.
- For the query `village`:
  - The only associated target word is `house`.
  - so the output is `1 house`

1.7.2 Example 2

Input:

- In this example, we have three sessions.
- In the first session, the target word is `cat` and the guess words are `animal` and `cat`.
- In the second session, the target word is `hapdogpy` and the guess words are `animal`, `cat`, and `dog`.
- In the third session, the target word is `bird` and the guess words are `crow`, `fly` and `bird`
- The next line `3` represents the number of guess words or prefixes of guess words we are interested
- `an`, `c`, and `dog` are the words for which we need to retrieve all target words associated with these guess words or prefixes of the guess word.
- Here notice that `c` and `an` are prefixes of guess words and are not guess words on their own.

Output:

- For the query `an`:
  - The only guess word with prefix `an` is `animal`
  - for the guess word `animal` associated target words are `dog` and `cat`
  - so the output is `2 cat dog`.
- For the prefix of guess words `c`:
  - There are two guess word with prefix `c`, they are `cat` and `crow`.
  - The associated target words for guess word `cat` is `cat` and `dog`.
  - The associated target words for guess word `crow` is `bird`.
  - So possible target words for prefix of guess word `c` are `cat`, `dog`, and `bird`.
  - In sorted order: `bird`, `cat`, `dog`.
  - So the output is `3 bird cat dog`
- For the guess word `dog`
  - The only associated target word is `dog`
  - so the output is `1 dog`.