HW #3. Search (Binary Input)

(Due: 2017/04/23)

Objective

- Practices on way to handle approximate string matching.

Introduction

In this assignment you are asked to implement the search feature of a phonebook. The idea is that for each pattern given, your program should be able to selects all possible contacts where their phone number matches the given pattern.

Due date

This assignment due on **2017-04-23 (Sun), 23:59:00**.

In this assignment, time takes to analyze your program on CodeSensor might increase significantly compared to the previous assignment. You are encouraged to start this assignment earlier so you can have more time to comes out with idea of implementing the program that has higher efficiency.

Discussion Board

If you have any problem regard this homework, please post your question in our discussion board, or email your question to me at cyng93+algo@gmail.com. You are encouraged to post the question on discussion board as others might have the similar question as you does.

Input

Please noted that the input is given in binary format, you are expected to DECODE THE BINARY FORMAT OF INPUT USING MSGPACK. The **decoded** input's format looks like what are being shown below:

N	$line_1$
x1 $Name_1$ $PhoneNum_1$ $Name_2$ $PhoneNum_2$ … $Name_{x1}$ $PhoneNum_{x1}$	$line_2$
n1 $Pattern_1$ $Pattern_2$ … $Pattern_{n1}$	$line_3$
x2 $Name_1$ $PhoneNum_1$ $Name_2$ $PhoneNum_2$ … $Name_{x2}$ $PhoneNum_{x2}$	$line_4$
n2 $Pattern_1$ $Pattern_2$ … $Pattern_{n2}$	$line_5$
…	$line_{...}$
xN $Name_1$ $PhoneNum_1$ $Name_2$ $PhoneNum_2$ … $Name_{xN}$ $PhoneNum_{xN}$	$line_{2N}$
nN $Pattern_1$ $Pattern_2$ … $Pattern_{nN}$	$line_{2N+1}$

Fig.1 - Input format

- You are encouraged to decode the msgpack into vector/list of strings. Each element in vector/list represented a **complete line of input**. We use *whitespace* to separate each word in lines. Keep in mind that each line is represented in string, you might need to cast them into appropriate type while implementing your program.
- The first line of the input --- N indicates that there are total N ($1 \leq N \leq 5$) runs.
- For each run, there are two lines of inputs. The first line holds the dataset, while the second line contains the patterns we wanted to query for.
  - ✓ First line(dataset) -- Line starts with X ($1 \leq X \leq 10,000$), follow by X contacts. Each contact is formed by a Name-PhoneNum pair, where *whitespace* is used to separate two contacts, as well as separate Name from PhoneNum, i.e,

    ```
    3 albert 0987785671 ken 0998987621 cedric 0912978798
    ```

  - ✓ Second line(patterns to query for) -- Line start with I ($1 \leq I \leq 1,000,000$), followed by I patterns. Patterns are made up by various digits, where each patterns are separated by *whitespace*, i.e,

    ```
    5 0972 781 9378 2378 6723
    ```

Output

Your program should ENCODE THE OUTPUT TO BINARY FORMAT USING MSGPACK.
Belows are the format of the output **before encoding**:

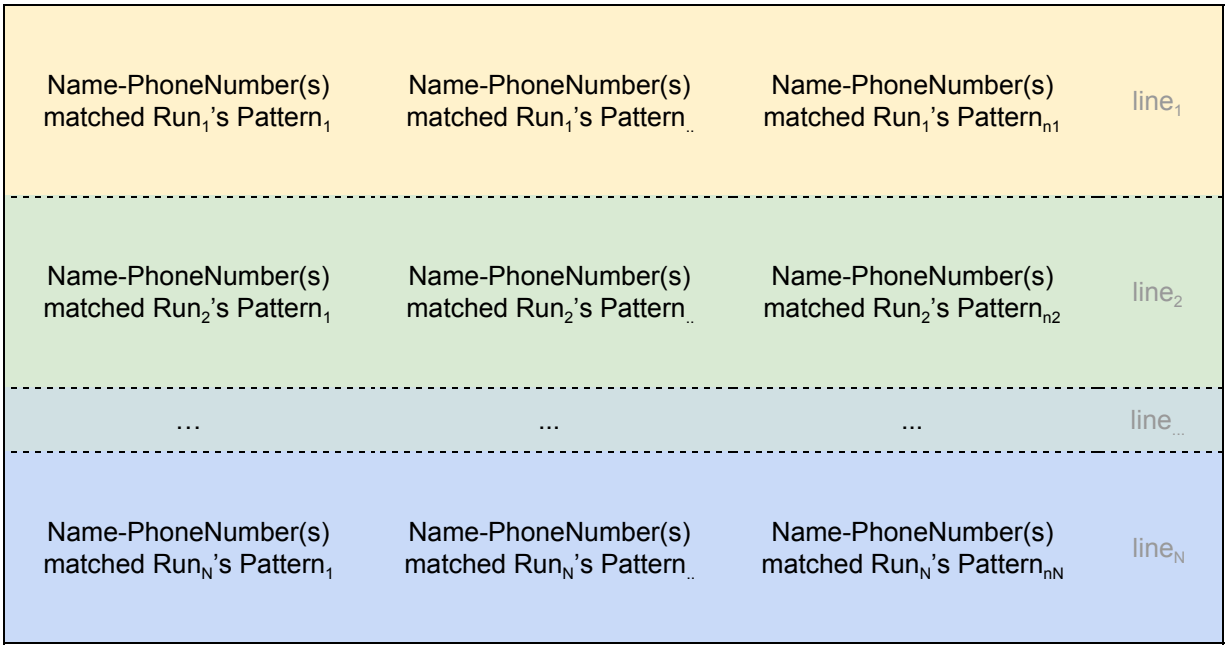

Fig.2 - Output format

- All query results from the same run should be placed in the same line, separated by whitespace.
- For each pattern mentioned in input, your program should be able to return those contacts where the pattern matches part of their PhoneNum (pattern is the substring of contact's PhoneNum). Each pattern is guaranteed to have AT LEAST ONE Name-PhoneNum whose PhoneNum will match the pattern. If there are more than one matched contact, you program should **sort the contact** with their PhoneNum ascendingly, where *whitespace* should be used to separate two contacts.
- You earn score for a particular run only when you have correct answers for all queries in that run.

Example

Input (after decode):

2	$line_1$
3 aa 0932804251 bb 0932804266 cc 0932804310	$line_2$
3 9328042 3280431 328042	$line_3$
6 dd 0932804384 ee 0932804326 ff 0932804402 gg 0932804410 hh 0932804383 ii 0932804354	$line_4$
7 932804 3280435 32804 80438 328044 3280432 3280435	$line_5$

Output (before encode):

aa 0932804251 bb 0932804266 cc 0932804310 aa 0932804251 bb 0932804266	$line_1$
ee 0932804326 ii 0932804354 hh 0932804383 dd 0932804384 ff 0932804402 gg 0932804410 ii 0932804354 ee 0932804326 ii 0932804354 hh 0932804383 dd 0932804384 ff 0932804402 gg 0932804410 hh 0932804383 dd 0932804384 ff 0932804402 gg 0932804410 ee 0932804326 ii 0932804354	$line_2$

Hints

The easiest way to complete this assignment is to burst-forcely go through all the contacts and select out those whose PhoneNum matches a given pattern.

Using Suffix Tree & Generalized Suffix Tree in your program might help your program run faster, but will also consume more memory. For those who are interested in Suffix Tree, you are encouraged to understand suffix trees before start looking into generalized suffix trees.

Any other implementation that could help you to come out with a program that has higher efficiency are also welcomed!

References

- MsgPack: http://msgpack.org/index.html
- SuffixTree:
  - https://www.cs.helsinki.fi/u/tpkarkka/teach/16-17/SPA/lecture08.pdf
  - http://www.geeksforgeeks.org/ukkonens-suffix-tree-construction-part-1/