Due: Dec. 4th (12/04), 13:00

Overview

This assignment consists of two parts: implementing a substring pattern matching data structure and a query-answering data structure.

General Notes

- *Read this homework guideline carefully.* If you do not follow the guidelines, you may receive a 0 regardless of whether your code works or not.
- Do not use any IDEs (Eclipse, IntelliJ IDEA, etc.)
  - We recommend Sublime Text (Linux/Mac/Windows), Notepad++ (Windows), or TextWrangler (Mac).
  - IDEs often create a "package" of your code, which breaks the autograder.
  - **If you know how to fix the package problem**, you can use any IDE you want. However, we will not answer any questions related to this problem since we have already recommended a solution.
- Do not change any method or class signatures. If your code changes any class or method names or signatures, you will receive an automatic 0.
- Make sure your code compiles. Non-compiling code will automatically receive a 0. If your code does not compile, it may be better to just comment out the incorrect code and return a dummy value (something like null or -1) so the rest can compile.
- To ensure that your code will be accepted by the autograder, you should submit your code on LearnUS, download it again, recompile it and check the provided test suite. This way, you know the file you are submitting is correct.
- If you use any code from the textbook, make a reference to the textbook through comments. If your code is flagged as copied and does not have a legitimate reference, you may get a 0 for plagiarism.

Substring Search

You will implement `YonseiSubstringSearch`, a data structure that *efficiently* counts the number of pattern occurrences over the text `t` of the class `YonseiString`. The class `YonseiString` is a specific type of string where each character can be any integer value. Let $|\texttt{w}|$ denote the number of characters in the `YonseiString` instance `w`. An instance of the `YonseiString` class is equipped with the following methods.

- `int charAt(int pos)`: returns the character at position `pos`. The indices start from 0. The method's behavior for input positions outside the range $[0, |\texttt{this}| - 1]$ is undefined. This method runs in $O(1)$ time.
- `int length()`: returns $|\texttt{this}|$, the number of characters in the `YonseiString` instance. This method runs in $O(1)$ time.

Your job is to implement the following methods.

- `YonseiSubstringSearch(YonseiString t)`: The constructor. Any `countPattern(p)` calls to this instance should use the given `YonseiString` `t` as the text. The constructor should run in $O(1)$ time.
- `int countPattern(YonseiString p)`: Counts and returns the number of occurrences of the pattern `p` from the text `t` given in the constructor. The `countPattern(YonseiString p)` method should run in $O(|\texttt{p}| + |\texttt{t}|)$ time.

For instance, given the text `t`="mississippi", the results of `countPattern` calls according to each pattern are:

Pattern	Result	Note
i	4	mississippi
issi	2	mississippi (They are overlapped.)
ip	1	mississippi
sss	0	

Prefix-Min

The data structure `PrefixMin` stores string-integer pairs as entries. For each entry, the string is the key and the integer is the value. The data structure can add or delete entries in time linear in the length of the key. Also, given a string, the data structure can compute the minimum value among all strings that have the given string as a proper prefix in time linear in the length of the given string. All strings are given as an instance of the `YonseiString` class. Two `YonseiString` instances are equal if they have the same length and have the same characters in the same order. Your job is to implement the following methods.

- `PrefixMin()`: the constructor.
- `int getValue(YonseiString key)`: if there exists an entry with the given `YonseiString` object as the key, return its value. Otherwise returns `Integer.MIN_VALUE`.
- `void insert(YonseiString key, int value)`: if there does not exist an entry with the key `key`, insert the entry (`key`, `value`) in the `PrefixMin` instance. Otherwise, do nothing.
- `void remove(YonseiString key)`: if there exists an entry with the key `key`, remove that entry. Otherwise, do nothing.
- `void clear()`: removes all entries from the `PrefixMin` instance.
- `int getMin(YonseiString query)`: returns the minimum value over all entries which have the `query` as a proper prefix of the entry's key. If there are no such entries, then return `Integer.MIN_VALUE`.

The methods `getValue()`, `insert()`, and `remove()` should all run in $O(|\texttt{key}|)$ time, and the method `getMin()` should run in $O(|\texttt{query}|)$ time. Also, the method `clear()` should run in $O(1)$ time.

General Directions

- Write your name and student ID number in the comment at the top of the files in src/main directory.
- Implement all of the required methods.
- You should not import anything that is not already included in the file.
- Pay careful attention to the required return types and edge cases.
- All the codes we provided can be found in src/base directory. If you are unsure what a class/method exactly does, please refer to the code.

Submission Procedure

Submit the files in the src/main directory, excluding Main.java, as a zip file. You *must* make a zip file for submission using the Gradle build tool (refer to Compiling section).

For this assignment, you should submit only the following three files:

- YonseiSubstringSearch.java
- PrefixMin.java
- your_student_id_number.txt

You must rename 2023xxxxxx.txt to your actual student ID number. Inside that text file, you must include the following text. Please be sure to write all the following text including the last period.

In completing this assignment, I pledge that I have not given nor received any unauthorized assistance.

If this file is missing, you will get a 0 on the assignment. It should be named *exactly* your student id, with no other text. For example, *2023123456.txt* is correct while something like *2023123456_pa5.txt* will receive 0.

Compiling

You can test your Java code using the following command:

```
% ./gradlew -q runTestRunner
```

You can also make a zip file for submission using the following command. The zip file named with your student id (the name of the .txt file) will lie in the “build” directory:

```
% ./gradlew -q zipSubmission
```

We provide an empty Main class for testing using standard input/output:

```
% ./gradlew -q --console=plain runMain
```

Since this file (src/main/Main.java) is not for submission, you can use any package in the file.

On Windows, use `gradlew.bat` instead of `./gradlew`.

Testing

We have provided a small test suite (src/test) for you to check your code. You can test your code by compiling and running the tester program.

Note that the test suite we will use to grade your code will be much more rigorous than the one provided here (and not necessarily a superset of the provided tests). You should consider making your own test cases to check your code more thoroughly.