CS 245 - Lab Assignment 3 Fall, 2020

Interview Coding Problem: Delete Columns to Sort

Consider the following scenario:

- You are given an array A of N lowercase letter strings, all of the same length.
- You may choose any set of deletion indices, and for each string, you delete all the characters in those indices.
- For example, if you have array `A = ["abcdef","uvwxyz"]` and deletion indices `{0, 2, 3}`, then the final array after deletions is `["bef", "vyz"]`, and the remaining columns of `A` are `["b","v"]`, `["e","y"]`, and `["f","z"]`. (Formally, the c-th column is `[A[0][c], A[1][c], ..., A[A.length-1][c]]`).
- Suppose you chose a set of deletion indices `D` such that after deletions, each remaining column in `A` is in non-decreasing sorted order.

Task (with examples provided below):

- Return the indices of the columns in `A` that are not in sorted order.
- Return an empty set of indices (i.e. `{}`) if all columns are in order.
- Return `{-1}` if there are any errors.

In other words: which columns (`D`) do you need to delete so that the remaining array's columns are in non-decreasing (sorted) order? Note that your implementation is not allowed to change the values in the array, `A`.

Examples

Here are four examples:

Example 1:

- Input: `A = ["cba","daf","ghi"]`
- Output: `{1}`

Explanation:

- After choosing `D = {1}`, each column `["c","d","g"]` and `["a","f","i"]` are in non-decreasing sorted order.
- If we chose `D = {}`, then a column `["b","a","h"]` would not be in non-decreasing sorted order.

Example 2:

- Input: `A = ["a","b"]`
- Output: `{}`

Explanation:

- "a" and "b" are separate individual columns and are therefore already sorted so we would output 0

Example 3:

- Input: `A = ["zyx","wvu","tsr"]`
- Output: `{0, 1, 2}`

Explanation:

- `D = {0, 1, 2} or D = 3`
- `Column 0: "z" "w" "t" is not sorted`
- `Column 1: "y" "v" "u" is not sorted`
- `Column 2: "x" "u" "r" is not sorted`

Example 4:

- Input: `A = ["Captain","Marvel","saved", "the", "Avengers"]`
- Output: `{-1}`

Explanation:

- The rows contain columns of unequal length, which is not allowed according to the description above.
- Output `{-1}` is required for errors such as this

Requirements

You must submit a single Java file containing a class named SortColumns. This class must contain a function "minDeletionSize" with the following header:

```
List<Integer> minDeletionSize(String [] A)
```

… or alternatively:

```
int [] minDeletionSize(String [] A)
```

The `minDeletionSize` function must return the indices of the columns in the argument, `A`, that are not in sorted order. It must return an empty List or empty array when all columns in the argument, `A`, are in sorted order.

Your class may contain a main function for testing, along with other helper functions.

Recommendations

Use a List<Integer> in the `minDeletionSize(...)` function. A List is a variable-sized container (i.e. can hold 0, 1 or more objects). We will discuss details of the List class and some of its concrete implementations later in the semester. Until then, a List instance for containing Integer instances, called "myList" can be declared and used as follows:

```
// Import necessary packages to use the List
import java.util.ArrayList;
import java.util.List;
```

```java
// Declare an instance of List.
// Since a List is an interface, we use ArrayList also.
List<Integer> myList = new ArrayList<Integer>();

// Use a List's important functions: add, get.
// The "add" function adds the content to the end.
myList.add(12);
myList.add(15);

// The "get" function retrieves the data at the index provided
// ... so this loop prints the data in order.
for (int i = 0; i < myList.size(); i++) {
     System.out.println(myList.get(i));
}
```

Submission

In a unique repository on github, check all source code. If you are unsure about how to create a (free) Github account, speak with the instructor or either of the TAs. The source code must be in Java. If your github repository is not public, you must allow access for the following user IDs: dbrizan, dennisdang17, huntermrocha. Place the URL for this repository on Canvas. You must also submit an README file (text, PDF or Word document) which explains the asymptotic runtime and space complexity of your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 70% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
- 15% = Decomposition: in the eyes of the grader, your solution must follow the suggestions above or otherwise must represent a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Readme: Explain the runtime and space complexity of your implementation
- 5% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.