Challenge: Reassemble Text Fragments

Background

Imagine you have 5 copies of the same page of text. You value this text and have no hard or soft copies of it. Your two year old nephew visits and, while you are not looking, rips each page up into fragments and gleefully plays in the "snow" he has just created.

You need at least one copy of that page of text back ASAP. As punishment to your niece, who should have been supervising your nephew at the time of the incident, you set her the painstaking task of keying in all the paper text fragments to a text file on your shiny MacBook Pro. Now the task is yours. Can you reassemble a soft copy of the original document?

The Challenge

Write a program to reassemble a given set of text fragments into their original sequence. For this challenge your program should have a main method accepting one argument – the path to a well-formed UTF-8 encoded text file. Each line in the file represents a test case of the main functionality of your program: read it, process it and println to the console the corresponding defragmented output.

Each line contains text fragments separated by a semicolon, ';'. You can assume that every fragment has length at least 2.

Example input 1:

```
O draconia;conian devil! Oh la;h lame sa;saint!
```

Example output 1:

```
O draconian devil! Oh lame saint!
```

Example input 2:

```
m quaerat voluptatem.;pora incidunt ut labore et d;, consectetur, adipisci
velit;olore magnam aliqua;idunt ut labore et dolore magn;uptatem.;i dolorem
ipsum qu;iquam quaerat vol;psum quia dolor sit amet, consectetur, a;ia
dolor sit amet, conse;squam est, qui do;Neque porro quisquam est, qu;aerat
voluptatem.;m eius modi tem;Neque porro qui;, sed quia non numquam ei;lorem
ipsum quia dolor sit amet;ctetur, adipisci velit, sed quia non numq;unt ut
labore et dolore magnam aliquam qu;dipisci velit, sed quia non numqua;us
modi tempora incid;Neque porro quisquam est, qui dolorem i;uam eius modi
tem;pora inc;am al
```

Example output 2:

```
Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit, sed quia non numquam eius modi tempora
incidunt ut labore et dolore magnam aliquam quaerat voluptatem.
```

You may also assume:

1. Your niece's attention to detail is impeccable; she has not misread any fragments or made any typos.
2. All test cases / input lines will reduce to one unambiguous, final, reassembled document.

Implementation

For each input line, search the collection of fragments to locate the pair with the maximal overlap match then merge those two fragments. This operation will decrease the total number of fragments by one. Repeat until there is only one fragment remaining in the collection. This is the defragmented line / reassembled document.

If there is more than one pair of maximally overlapping fragments in any iteration then just merge one of them. So long as you merge *one maximally* overlapping pair per iteration the test inputs are guaranteed to result in good and deterministic output.

When comparing for overlaps, compare case sensitively.

Examples:

- "ABCDEF" and "DEFG" overlap with overlap length 3
- "ABCDEF" and "XYZABC" overlap with overlap length 3
- "ABCDEF" and "BCDE" overlap with overlap length 4
- "ABCDEF" and "XCDEZ" do *not* overlap (they have matching characters in the middle, but the overlap does not extend to the end of either string).

Your submission will be processed automatically and will not be seen by our technical team unless it passes so please follow the instructions below carefully.

Instructions:

- Your code must compile and run against **Java 8**.
- Your code must have a main method receiving one argument - the absolute path to the input file.

- Your code must read the input file treating each line of the file as a separate "test case" / separate instance of the problem. Each line is a semicolon-separated collection of text fragments.
- The challenge is to defragment each line and print the solution to each problem as a single line to the console.
- The console output of your code will be evaluated by an automatic process; do not print anything other than the solutions to each problem to the console or it will fail.
- Consider the edge cases carefully. Your solution will be tested against a wide variety of valid fragment problems.