EN 600.425 Declarative Methods - Term Project Writeup
Better `diff`

Guoye Zhang, Qiang Zhang

May 6, 2017

1 Introduction

The utility `diff` is commonly used in all scenarios, whether for programming purpose or in general text editing circumstances. The website Github integrated `diff` into their version control system, facilitating ubiquitous knowledge of the utility. This legacy utility is based on the renowned *Edit Distance* algorithm, which in essence is a Dynamic Programming algorithm.

Aside from the application domain of version control, `diff`'s algorithm in itself is interesting and instructive enough for most computer science students to delve into. In general sense, the concepts of finding similar text and displaying minimum edit distance can be embodied in various domains and categories of applications. Our project aims at extending `diff` to provide firstly the feature of multi-file editting source referencing, accompanied with visualization, that can display the chain of editting among more than two files placed in the same folder, together with a better algorithm that fixes a certain scenario where the normal `diff`'s Edit Distance algorithm would perform suboptimally, which is a case we already described in the proposal.

1.1 Algorithm Optimization

As a reminder, recall the scenario where normal `diff` would perform suboptimally. In this code snippet:

```
for(int i=0; i<10; i++) {
    System.out.println("First␣line");
}
for(int i=0; i<10; i++) {
    System.out.println("Second␣line");
}
for(int i=0; i<10; i++) {
    System.out.println("Third␣line");
}
```

from which we will delete the second `for` block. And if we use the traditional `diff` on the old and new versions of this snippet, we would get:

```
  for(int i=0; i<10; i++) {
      System.out.println("First␣line");
  }
  for(int i=0; i<10; i++) {
-     System.out.println("Second␣line");
- }
- for(int i=0; i<10; i++) {
      System.out.println("Third␣line");
  }
```

Instead of what would be expecting as follows:

```
  for(int i=0; i<10; i++) {
      System.out.println("First␣line");
  }
- for(int i=0; i<10; i++) {
-     System.out.println("Second␣line");
- }
  for(int i=0; i<10; i++) {
      System.out.println("Third␣line");
  }
```

And this case is fixd with our improved version of `diff` algorithm, which we shall call by `MDiff` in the rest of this document. In fact, there are much more scenarios where the traditional `diff` algorithm would disappoint you. For example, consider the instance of comparison below (results are hand-generalized from Github's webpage highlighting):

```
<a><b>
 ----

<b><a>
 ++++
```

The notations used here should be self-explanatory. When comparing the two strings of `<a><b>` and `<b><a>`, traditional `diff` would return the results as shown above, which means, in a 0-based indexing convention, deleting the 1-4 of the first string, and adding the 4 characters at index 1-4 of the second string. But this answer of editting distance is clearly suboptimal. The optimal solution we expect would be:

```
<a><b>
 -  -

<b><a>
 +  +
```

which is the answer that `MDiff` would return.

1.2 Multi-file Referencing

In this part, we extended `diff` to another domain of applicaiton of interest. To most eyes, `diff` is generally used to comparing the editting distance between two files. It is a utility used widely in use. Indeed, it is a utility, in that it does a job well, but it is not typically tailored for any domain. We implemented an application that embodies `MDiff` in its core, but meanwhile provides far more convenient and appleasing functionalities.

For oft, we are interested not only to know the difference between two text files. Rather, we want to be able to look at some paragraph of this text file, and find out from which other paragraph in other file this paragraph derives. Text reusing is so common nowadays that this is a feature of no trivial productivity significance. In fact, to fullfill such a task, of finding the ancestor of a paragraph in this text file, most people would resort to nothing better than a brute-force file-by-file *Open&Search* routine.