Theory: Edit distance in Java

29 minutes 0 / 5 problems solved

Skip this topic Start practicing

Edit distance is a metric that allows estimating the similarity between two arbitrary strings. The metric is defined as the minimum number of insertions, deletions or substitutions required to transform one string into the other.

An algorithm for calculation of edit distance can be implemented using the dynamic programming technique. We can start calculation of edit distance from smaller prefixes of both strings (divide the problem into smaller subproblems in term of dynamic programming), then use the answers for these subproblems to find edit distance for bigger prefixes and finally get the answer for the initial strings.

§1. An implementation in Java

Consider how the algorithm can be implemented in Java:

```
public static int match(char a, char b) {
    return (a == b) ? 0 : 1;
}

public static int editDistance(String s, String t) {
    int[][] d = new int[s.length() + 1][t.length() + 1];

    for (int i = 0; i < s.length() + 1; i++) {
        d[i][0] = i;

    }



    for (int j = 0; j < t.length() + 1; j++) {

        d[0][j] = j;

    }



    for (int i = 1; i < s.length() + 1; i++) {

        for (int j = 1; j < t.length() + 1; j++) {

            int insCost = d[i][j - 1] + 1;

            int delCost = d[i - 1][j] + 1;

            int subCost = d[i - 1]
[j - 1] + match(s.charAt(i - 1), t.charAt(j - 1));

            d[i][j] = Math.min(Math.min(insCost, delCost), subCost);

        }

    }



    return d[s.length()][t.length()];

 }
```

The `editDistance` method takes two strings `s` and `t` as arguments and returns the edit distance for the strings.

119 users completed this topic.
Latest completion was
13 days ago.

Current topic:

Edit distance in Java

Topic depends on

- Edit distance
- Algorithms in Java

Topic is required for

Edit distance alignment in Java

Table of contents:

Initially, we create an array `d` to store intermediate answers. Then, we fill in the first row and the first column of the array (recall that they correspond to the transformation of an empty string into `s` or `t`). Next, we start successively calculating the edit distance for each pair of prefixes of `s` and `t`. When all prefixes are processed, the lower-right cell of the array contains the edit distance for the initial strings and we return it as a final result.

The method can be used like this:

```
1  editDistance("editing", "distance"); // 5
```