Edit Distance in Java

Category: Algorithms December 10, 2013

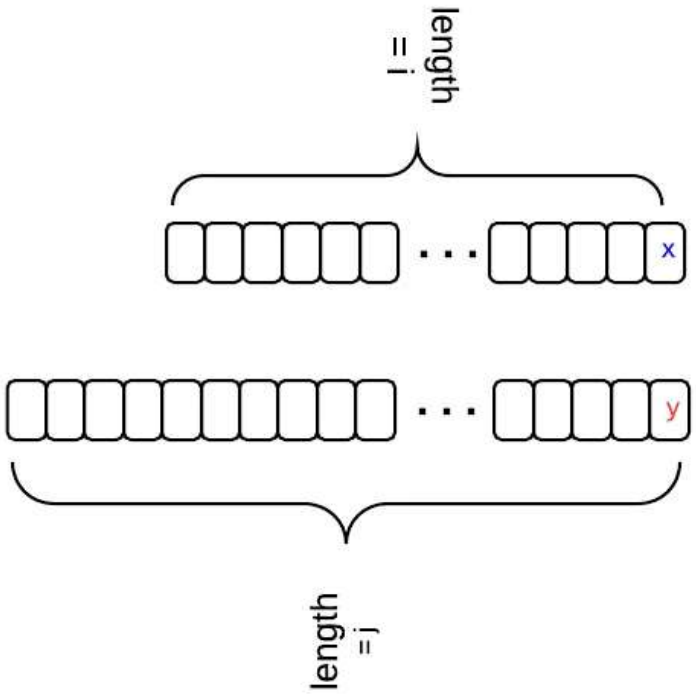
From Wiki:

> In computer science, edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other.

There are three operations permitted on a word: replace, delete, insert. For example, the edit distance between "a" and "b" is 1, the edit distance between "abc" and "def" is 3. This post analyzes how to calculate edit distance by using dynamic programming.

Key Analysis

Let dp[i][j] stands for the edit distance between two strings with length *i* and *j*, i.e., word1[0,...,i-1] and word2[0,...,j-1].
There is a relation between dp[i][j] and dp[i-1][j-1]. Let's say we transform from one string to another. The first string has length *i* and it's last character is "x"; the second string has length *j* and its last character is "y". The following diagram shows the relation.



1. if x == y, then dp[i][j] == dp[i-1][j-1]
2. if x != y, and we insert y for word1, then dp[i][j] = dp[i][j-1] + 1
3. if x != y, and we delete x for word1, then dp[i][j] = dp[i-1][j] + 1

4. if x != y, and we replace x with y for word1, then dp[i][j] = dp[i-1][j-1] + 1
5. When x!=y, dp[i][j] is the min of the three situations.

Initial condition:
dp[i][0] = i, dp[0][j] = j

Java Solution 1 - Iteration

After the analysis above, the code is just a representation of it.

```
public static int minDistance(String word1, String word2) {
        int len1 = word1.length();
        int len2 = word2.length();

        // len1+1, len2+1, because finally return dp[len1][len2]
        int[][] dp = new int[len1 + 1][len2 + 1];

        for (int i = 0; i <= len1; i++) {
                dp[i][0] = i;
        }

        for (int j = 0; j <= len2; j++) {
                dp[0][j] = j;
        }

        //iterate though, and check last char
        for (int i = 0; i < len1; i++) {
                char c1 = word1.charAt(i);
                for (int j = 0; j < len2; j++) {
                        char c2 = word2.charAt(j);

                        //if last two chars equal
                        if (c1 == c2) {
                                //update dp value for +1 length
                                dp[i + 1][j + 1] = dp[i][j];
                        } else {
                                int replace = dp[i][j] + 1;
                                int insert = dp[i][j + 1] + 1;
                                int delete = dp[i + 1][j] + 1;

                                int min = replace > insert ? insert : replace;
                                min = delete > min ? min : delete;
                                dp[i + 1][j + 1] = min;
                        }
                }
        }

        return dp[len1][len2];
}
```

Java Solution 2 - Recursion

We can write the solution in recursion.

```
public int minDistance(String word1, String word2) {
    int m=word1.length();
    int n=word2.length();
    int[][] mem = new int[m][n];
    for(int[] arr: mem){
        Arrays.fill(arr, -1);
    }
    return calDistance(word1, word2, mem, m-1, n-1);
```

```
}

private int calDistance(String word1, String word2, int[][] mem, int i, int j){
    if(i<0){
        return j+1;
    }else if(j<0){
        return i+1;
    }

    if(mem[i][j]!=-1){
        return mem[i][j];
    }

    if(word1.charAt(i)==word2.charAt(j)){
        mem[i][j]=calDistance(word1, word2, mem, i-1, j-1);
    }else{
        int prevMin = Math.min(calDistance(word1, word2, mem, i, j-1), calDistance(word
        prevMin = Math.min(prevMin, calDistance(word1, word2, mem, i-1, j-1));
        mem[i][j]=1+prevMin;
    }

    return mem[i][j];
}
```

Related Posts:

1. **LeetCode – Shortest Word Distance III (Java)**
2. **LeetCode – Shortest Word Distance (Java)**
3. **LeetCode – Shortest Word Distance II (Java)**
4. **LeetCode – One Edit Distance (Java)**

Category >> Algorithms

If you want someone to read your code, please put the code inside <pre><code> and </code></pre> tags. For example:

```
<pre><code>
String foo = "bar";
</code></pre>
```

Arthur • 3 years ago

This might be the BEST explain on the internet. Most other websites' explains are bullshit in my opinion.

mg • 5 years ago

```
public int minDistance(String word1, String word2) {

if((word1 == null || word1.length() == 0) && (word2 == null || word2.length() == 0)){
return 0;
}

if( word1 == null || word1.length() == 0 ){
return word2.length();
}

if(word2 == null || word2.length() == 0){
return word1.length();
}

int rows = word1.length() + 1;
int cols = word2.length() + 1;
```


Internet Hero • 7 years ago

int replace = dp[i][j] + 1;
int insert = dp[i][j + 1] + 1;
int delete = dp[i + 1][j] + 1;

Shouldn't this be reverse, ie

insert = dp[i+1][j]+1
delete = dp[i][j+1] + 1

Bhudev ➚ Internet Hero • 7 years ago

Yes it should, works either way

Vimukthi Weerasiri • 7 years ago

```
for (int i = 1; i < dp.length; i++) {
for (int j = 1; j < dp[0].length; j++) {
dp[i][j] = word2.charAt(i - 1) == word1.charAt(j - 1) ? dp[i - 1][j - 1] : Math.min(dp[i - 1][j - 1],
Math.min(dp[i - 1][j], dp[i][j - 1])) + 1;
}
}
```

A simpler loop which gives the solution

^ | v • Reply • Share ›

AThomasTran • 8 years ago

Appreciated! This example was *exactly* what I was looking for to solidify my understanding of Min Edit Distance DP problem

^ | v • Reply • Share ›

RoganDawes • 8 years ago

Note that there is no need to keep the full 2 dimensional array of results. This obviously grows with the product of the length of the two strings. All you really need to keep is 2 rows, and you simply switch them as you move from row to row.

Unfortunately, that doesn't change the number of operations required to calculate the edit distance, which remains O(m*n), where m and n are the lengths of the strings.

^ | v • Reply • Share ›

Dmitry Petrov ↗ RoganDawes • 6 years ago

1 row and 1 value are enough :)

^ | v • Reply • Share ›

RoganDawes ↗ Dmitry Petrov • 6 years ago

True enough. Good point. Simply update the single row in place as you work through it.

^ | v • Reply • Share ›

PrasAnth Mamakar Komaragiri • 6 years ago

Hey can you please tell me how to print the matrix at every step in this example

^ | v 1 • Reply • Share ›