

Here are two different versions of the edit distance recurrence relation. Let

- d denote the cost of deleting a letter,
- a denote the cost of adding a letter,
- r denote the cost of replacing a letter.

1. Version 1.

$$ED(s, t, i, j) = \begin{cases} a * j & \text{if } i = 0 \\ d * i & \text{if } j = 0 \\ \min \begin{cases} \begin{cases} ED(s, t, i-1, j-1) & \text{if } s_i = t_j \\ r + ED(s, t, i-1, j-1) & \text{if } s_i \neq t_j \end{cases} \\ a + ED(s, t, i, j-1) \\ d + ED(s, t, i-1, j) \end{cases} \end{cases}$$

2. Version 2.

$$ED(s, t, i, j) = \begin{cases} a * j & \text{if } i = 0 \\ d * i & \text{if } j = 0 \\ ED(s, t, i-1, j-1) & \text{if } s_i = t_j \\ \min \begin{cases} r + ED(s, t, i-1, j-1) \\ a + ED(s, t, i, j-1) \\ d + ED(s, t, i-1, j) \end{cases} & \text{if } s_i \neq t_j \end{cases}$$

3. We write

$$d + ED(s, t, i-1, j)$$

to denote that we first delete a character from the end of s and then (recursively) edit the resulting string of length $i-1$ to match the substring of t of length j . We write

$$a + ED(s, t, i, j-1)$$

to denote that we first (recursively) edit the string s to match the substring of t of length $j-1$ and then add a character to the end of the resulting string. We write

$$r + ED(s, t, i-1, j-1)$$

to denote that we first (recursively) edit the substring s of length $i-1$ to match the substring of t of length $j-1$ and then add a character to the end of the resulting string.

4. Notice that these two recurrence relations do not describe the same algorithm (though hopefully they describe equivalent algorithms). For example, if the source and target strings are

$$s = \text{"abbb"} \quad \text{and} \quad t = \text{"abb"},$$

then the first algorithm will delete the last "b" from s . The second algorithm will first match the b's at the end of s and t , then it will match the second to last b's in s and t , and then it will delete the third from last "b" in s .