Minimum Edit Distance

Minimum Edit distance is a measure used to compare the distance between two strings. It is calculated by the number of simple operations – insertion, deletion and substitution – required to convert one string to another. Insertion and deletion cost 1 operation. But a substitution is considered as two operations. You are required to write a procedure which calculate edit distance between pairs of words. The words will contain only lowercase alphabet characters. The distance will be measured in integers.

Mathematically, the minimum edit distance between two strings $w_1, w_2$ is given by $editDist_{w_1,w_2}(\text{len}(w_1), \text{len}(w_2))$ where,

$$editDist_{w_1,w_2}(i,j) = \begin{cases} \max(i,j) & \text{if min}(i,j) = 0, \\ \min \left\{ \begin{array}{ll} editDist_{w_1,w_2}(i-1,j) + 1(w_1^i, w_2^j, \text{ins}) \\ editDist_{w_1,w_2}(i,j-1) + 1(w_1^i, w_2^j, \text{del}) \\ editDist_{w_1,w_2}(i-1,j-1) + 1(w_1^i, w_2^j, \text{sub}) \end{array} \right. & \text{otherwise.} \end{cases}$$

$$1(w_1^i, w_2^j, \text{op}) = \begin{cases} 2 & w_1^i \neq w_2^j \land \text{op} = \text{sub} \\ 0 & w_1^i = w_2^j \land \text{op} = \text{sub} \\ 1 & \text{otherwise} \end{cases}$$
**Example**

kitten, sitting : 5

kitten → sitten (substitution of k for s, +2)

sitten → sittin (substitution of e for i, +2)

sittin → sitting (add for g, +1)

**Requirement**

1. Implement a class which meets following requirements.

   class signature : public class EditDistance

   method signature : public static int calculateDistanceNoMemoize(String w1, String w2)

   returns minimum edit distance of w1 and w2 with no memoize.

   method signature : public static int calculateDistanceMemoize(String w1, String w2)

   returns minimum edit distance of w1 and w2 with memoize.

2. Analyze the time complexity of two methods, and show it graphically.

1Please use following package name → kr.ac.postech.csed233.hw4._ < YOUR_STUDENT_NO >

2Please hand in the report compressed with with the source code.