Algorithm 1: RecursiveED

Input: S1 , S2

```
/* Parameters :                                                    */
/*    s1,s2 :  strings.                                           */
/*    operations :  list.                                         */
/* Return :                                                       */
/*    ED, operations                                              */
1  begin
       /* if s1 is empty,we insert all characters of s2 in s1     */
2      if len[s1] == 0 then
3          return ED ←ED+len(s1), operations ←operations + ['insert'+x] for x in s2 ]
       /* if s2 is empty,we insert all characters of s1 s2        */
4      if len[s2] == 0 then
5          return ED ←ED+len(s2), operations ←operations + ['insert'+x] for x in s1 ]
       /* If last characters of both strings are the same,we set k=0 because we ignore them and we
          compute Edit Distance for these strings without last characters */
6      if (s1[-1]== s2[-1]) then
7          k ←0;
8          w1 = RecursiveED (s1[:-1],s2[:-1],operations+['skip':s1[-1]],ED+k)
       /* if last characters are different, we set k=1 and we consider all three operations on last
          character of s1, compute cost for all three operations */
9      else
10         k ←1
11         w1=RecursiveED(s1[:-1],s2[:-1],operations+['replace':s1[-1]],ED+k)
12     w2=RecursiveED(s1[:-1],s2,operations+['delete':s1[-1]],ED+1)
13     w3=RecursiveED(s1,s2[:-1],operations+['insert':s2[-1]],ED+1)
       /* compare all costs and choose the minimal one of them    */
14     return min(w1,w2,w3)
```