Algorithm 1: Greedy_Approach_Edit_Distance

Input: X, Y

```
/* Parameters :                                                    */
/*    X , Y : strings, we assume here that len(X) <= len(Y).       */
/* Return :                                                       */
/*    ed :  integer, optimal edit distance between X and Y.       */
/*    alignment :  array of instructions, to go from Y to X.      */
/*                                                                */
```

```
1  n ←len(X);
2  m ←len(Y);
3  ed ←∞;
4  for all possible combinaison 0≤j≤i≤min(n-m,⌈m/2⌉) s.t. i = j + (n-m) do
5      str1 ←i*"-" + X// e.g.  X= H E L L O , str1 = - H E L L O
6      str2 ←Y + j*"-"// e.g.  Y= H O L A,    str2 = H O L A - -
7
8      OR    // When all the possibilities are explored this way do it the other way.
9
10     str1 ←X + i*"-"// e.g.  X= H E L L O , str1 = H E L L O - -
11     str2 ←j*"-" + Y// e.g.  Y= H O L A,    str2 = - - - H O L A
12
13     ed_tmp,alignment_t ←greedy_ed(str1,str2);
14     if ed > ed_tmp then
15         ed ←ed_tmp;
16         alignment ←alignment_tmp;
17 for 0≤i≤n-m do
18     str1 ←X // e.g.  X= B O N J O U R , str1 = B O N J O U R
19     str2 ←(n-m-i)*"-" + Y + (i)*"-" // e.g.  Y= H O L A,        str2 = - H O L A - -
20     ed_tmp,alignment_t ←greedy_ed(str1,str2);
21     if ed > ed_tmp then
22         ed ←ed_tmp;
23         alignment ←alignment_tmp;
24 return {ed : ed , alignment : alignment}
```

Algorithm 2: greedy_ed

Input: str1 , str2

```
/* Check from left to right, 1 by 1 the letters of str1 and str2 to compare them.          */
/* Parameters :                                                                            */
/*    str1,str2 :  srings of same length.                                                  */
/* Return :                                                                                */
/*    not optimal edit_distance and alignment, computed without modifying str1 or str2.    */
```

```
ed ←0;
alignment ←[];
for i in range(len(str1)) do
    if str1[i] == str2[i] then
        alignment.append(["skip" , str1[i]]);
    else if str1[i] == "-" then
        alignment.append(["del" , str2[i]]);
    else if str2[i] == "-" then
        alignment.append(["add" , str1[i]]);
    else
        // str1[i] != str2[i]
        alignment.append(["sub" , str1[i]]);
    return {ed : ed , alignment : alignment}
```