UVa 843

From Algorithmist

The approach to solving this problem is backtracking. However, the backtrack tree will be pruned to such a state it will take a very short time to search.

First, let us define Ω as the set of all strings. Then, let us put up an equivalence class given by the relation

$R := \{(w, z) \in \Omega^2 : |w| = |z| \wedge match(w, z)\}$, where

match is a routine which takes as arguments two equal length words and return whether the characters in them match. So, for instance, match("abc", "xyz") will return true and match("iff", "abc") would not because the 'f' assumes the place of both the 'b' and 'c'.

match can be defined as

```
injective(A):
    let F the the array full of 0 with |A| elements
    foreach key k in A
        if F[k] != 0 then return false
        else F[k]:= 1
    return true

match(w, z):
    let A be the array full of '*' with 26 elements
    foreach keys (k, v) in (w, z)
        if A[k] != '*' && A[k] != A[v] then return false;
        else A[k]:= v
    return injective(A)
```

Then let us sort the words by an heuristic. Here it will be "the longer, the nearest the beginning". This is because matching those huge words is harder and should be done first. That must be done in such a way to leave the initial configuration stored somewhere else.

Consider D the dictionary. The let us make a list of sets L such that

$$L_i := \{w \in D : (words_i, w) \in R\}$$

Then just backtrack the matchings:

```
backtrack(A, i):
    if i = |words| then return true
    else do
        foreach word w in L[i]:
            B:= A
            if match(w, words[i], B) then do
                answer[i]:= w
                b:= backtrack(B, i+1)
                if b then return true
        return false
```

This match function is a little adaptation of the match function above, in which the array is passed by reference instead of being set to a bunch of '*'s. Call backtrack(the array full of '*'s with 26 elements, 0) to receive whether a solution is possible and have this solution stored in answer.

--Schultz 21:09, 26 May 2007 (EDT)