Estructura de Datos y Algoritmos

Grados en Ingeniería Informática, de Computadores y del Software

Examen Final Junio - 1 de junio de 2018 - Grupos C y F

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

1. **(3.5 puntos)** Sean dos cadenas de caracteres $V[0..N), W[0..M)$ con N, M arbitrariamente grandes. Se quiere desarrollar un programa que compruebe si la primera cadena es prefijo de la segunda sin tener en cuenta los espacios blancos que pueda haber en ambas. Para ello se pide:

   - (0.5 puntos) Especifica una función que dadas dos cadenas de caracteres $V[0..N), W[0..M)$ tales que $N <= M$ y ninguna de ellas tiene ningún carácter blanco compruebe que la primera cadena es prefijo de la segunda.
   - Implementa la función anterior, escribe el invariante (0.5 puntos) y la función cota (0.2 puntos) del bucle utilizado y comprueba el paso 2 de la verificación $\{I \wedge B\}S\{I\}$ (0.3 puntos)
   - (2 puntos)Modifica la implementación de la función anterior para que resuelva el problema completo de comprobar que la primera cadena es prefijo de la segunda sin tener en cuenta los espacios en blancos que pueda haber en ambas y pudiendo ser la primera cadena más larga o más corta que la segunda. La implementación debe realizarse con complejidad $\mathcal{O}(N + M)$. No pueden utilizarse cadenas auxiliares, ni otras estructuras auxiliares, ni realizar copia de cadenas, ni modificar las cadenas dadas.

   Ejemplos. Los blancos se indican con el carácter subrayado. ϵ es la cadena vacía. (Fichero Problema1Entrada.txt)

V	W	prefTrimSpaces
ada_cccb_b	a_dacc_cbb	TRUE
ada_ccc___	a_dacc_cbb	TRUE
aaa_cccb_b	a_dacc_cbb	FALSE
ada_ccbb_b	a_dacc__	FALSE
__________	a_dacc_cbb	TRUE
ϵ	ϵ	TRUE
a_da	ϵ	FALSE
ϵ	a_d	TRUE
a_d___	ad	TRUE

   La entrada consta de una serie de casos de prueba. Cada caso de prueba tiene dos líneas, en la primera se indica la cadena de caracteres que debe ser prefijo de la que aparece en la segunda línea.

   Para cada caso de prueba se escribirá en una línea `TRUE` si la primera cadena es prefijo de la segunda y `FALSE` en caso contrario.

   Nota: La especificación y verificación la puedes realizar en papel o sobre el fichero de código.

```
/*

Easy Spec part:
---------
P : 0 <= N <= M, V[0..N), W[0..M) , white(V,N)=white(V,W)=0
Q : \forall i : 0 <= i < N : V[i]=W[i]

    (alternative)
    N = min i : : 0 <= i < N and i<N -> V[i]!=W[i] : i

where white(V,N): #i : 0 <= i < N :V[i]==' '

Easy Implement:
--------------
   n = 0
   while (n<N and V[n]==W[n])
      n= n + 1
   return n=N

I : Q[N/n] and 0 <= n <= N
Cuote (N,n) = N-n <= 0

I and B => I[n/n+1]

Proof
----
0 <= n <= N and n<N
=>
 0 <= n+1 <= N

\forall i : 0 <= i < n : V[i]=W[i] and V[n]==W[n]
=>
\forall i : 0 <= i < n+1 : V[i]=W[i]



Hard Version (including spaces)
-------------------------------

(See tests below the code )


As a decission, we turn it into a optimization problem.

Informally: "The minimum non-white space position in V such
             that no _corresponding_ matching
             pos exists in W. (N if no such position in V exist)"

Formally: (Not required)

P: N >= 0
Q: N = min i : 0 <= i <= N and ( i < N ) -> R(V,i) : i

where
   R(V,i) : V[i]!=' ' and (
                           (nonb(W,0,M) < nonb(V,0,i)
                           or
                           W[pos(W,nonb(V,0,i))]!=V[i]
                           )

   pos(W,n)= min j : 0 <= j <= M and
                        (j<M) -> n= nonb(W,0,j)
                   : j
   nonb(W,0,n) = # i : 0 <= i <= n : W[i]!=' '



   var n,m
```

(Hard)

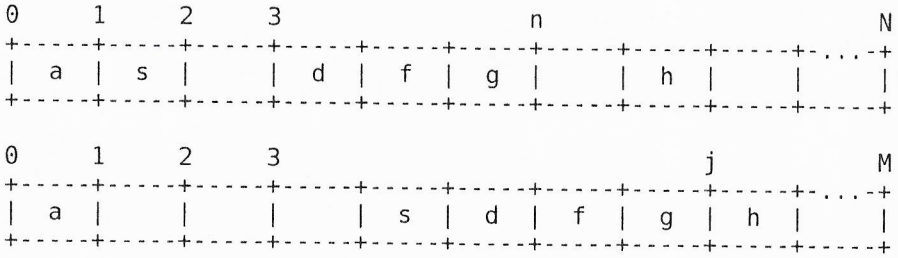
!B : n == N || (V[n]!=' ' && (m==M) || ((W[m]!=V[n]) && (W[m]!=' '))))

Informally
* we reach the end a V
* we have a non white at V and
    * no remaining chars in W
    * neither white nor matching char in W

B: (n < N) && (V[n]==' ' || ((m<M) and ((W[m]=V[n]) || (W[m]==' '))))

I : Q[N/n] and 0<=n<=N, 0 <=m<=M

Snapshot invariant



Quote: (N+M-(n+m)) >= 0
O(N+M) since body and B is O(1)

I and B => Q (By Lemma)
Lemma:

R \sqsubset R'

n = min i : R : i -> n = min i : R' : i

Init:

n,m = 0, 0

Step:

n,m = n + \chi(V[n]=' ' or (V[n]==W[m])),
m + \chi(m<M and ((V[n]==W[m]) or W[m]=' '))

or (conditional assignment)

case
(V[n]=' ') -> n = n + 1
(m < M && V[m]==V[n]) -> n,m = n+1,m+1
(m < M && V[m]==' ') -> m = m + 1
esac

Restore:

skip

Pseudocode:

```
n,m = 0, 0
while (n < N) && (V[n]==' ' || ((m<M) and ((W[m]=V[n]) || (W[m]==' '))))
  case
```

```
        (V[n]=' ') -> n = n + 1
        (m < M && V[m]==V[n]) -> n,m = n+1,m+1
        (m < M && V[m]==' ') -> m = m + 1
       esac
     done
     return (n==N)



*/

#include <iostream>
#include <string>

using namespace std;


int prefixUpToSpaces(const char V[],  const int N, const char W[], const int M)

{
  int n,m;
  for(n=m=0; // init
      (n<N) && (V[n]==' ' || ((m<M) && (W[m]==' ' || V[n]==W[m]))); )// B
    {
      if  (V[n]==' ') n++;
      else if  ((m<M) && (V[n]==W[m])) {n++; m++;}
      else if  ((m<M) && (W[m]==' ')) m++;
    }
  return (n==N);
}



int main()
{
  string str1,str2;
  for (  ; getline(cin, str1) && getline(cin, str2);  )
      cout <<
      (prefixUpToSpaces(str1.c_str(), str1.length(),
                        str2.c_str(), str2.length())?"TRUE":"FALSE") << endl;
  return 0;
}


/*

g++ sol.rafa.cc -o main && ./main
asdf              f
asdff
1
asdf
a              sdf
1
    (intro)
    (intro)
1
ada cccb b
a dacc cbb
1
ada ccc
a dacc cbb
1
aaa ccc b b
a dacc cbb
0
ada ccbb b
a dacc cbb
0
```

```
a dacc cbb
1


1

a dacc cbb
1
asdfasdf
a                           dsfadsf
0
asdfasdf
a                              sdfasdf
1
*/
```