ADF 2*x* & PRO 2*x* **Übungen zu Fortgeschrittenen Algorithmen & Datenstrukturen und OOP** **SS 21, Übung 2**

Abgabetermin: Mi, 07. 04. 2021

☐ **Gr. 1,** Dr. S. Wagner **Name** Angelos Angelis **Aufwand in h** 9

☒ **Gr. 2,** Dr. D. Auer

☐ **Gr. 3,** Dr. G. Kronberger **Punkte** _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

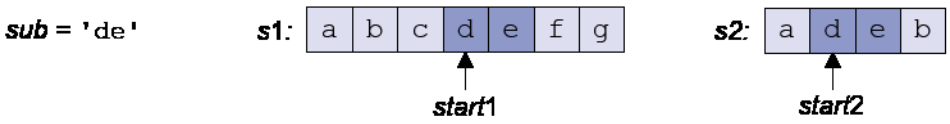
1. Längste gemeinsame Teilkette (12 Punkte)

Entwickeln Sie eine Pascal-Prozedur

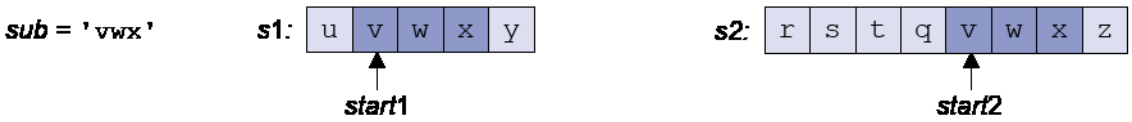
```
PROCEDURE FindLongestMatch(s1, s2: STRING; VAR sub: STRING;
                           VAR start1, start2: INTEGER);
```

die für zwei Zeichenketten *s*1 und *s*2 jene längste Teilkette (*substring*, *sub*) findet, die sowohl in *s*1 als auch in *s*2 vorkommt. Die Anfangsposition der gefundenen längsten Teilkette muss für *s*1 in *start*1 und für *s*2 in *start2* zurückgegeben werden.

Beispiel 1*:*



Beispiel 2*:*



Gibt es keine gemeinsame Teilkette, also nicht einmal ein gemeinsames Zeichen, liefert *sub* eine leere Zeichenkette und *start*1 sowie *start*2 den Wert 0.

Gibt es mehrere längste Teilketten, dann ist eine davon frei zu wählen und es sind für diese die entsprechenden Positionen in *start*1 und *start*2 zu liefern.

Für die Lösung von Teilaufgaben in *FindLongestMatch* können natürlich bekannte Verfahren zur Zeichenkettensuche eingesetzt werden, versuchen Sie aber, eine möglichst effiziente Gesamtlösung zu finden, und geben Sie eine Abschätzung der asymptotischen Laufzeitkomplexität dieser Gesamtlösung an.

1)Lösungsidee:

Es soll der größt-mögliche Substring in zwei Strings gefunden werden. Ich hab es mir ein wenig einfach gemacht und wechsle am Anfang die Strings falls s2 größer als s1 ist. Ich habe die Aufgabe mit Hilfe von 3 Schleifen gelöst. Die eine durchläuft s1 und die anderen zwei durchlaufen s2. Man vergleicht dann jeweils an der Stelle von der dritten Schleife und an i + k – j. Mit Hilfe einer BOOLEAN Variable und einer count Variable wird die Anzahl von den aufeinanderfolgenden Matches gespeichert. Letzten Endes muss der String ausgegeben werden.

Laufzeit hierbei wäre O(n*m^2)

(Quellcode kommt am Ende ich habe Aufgabe 1 und 2 in einer Pascal Datei programmiert)

Testfälle:

1)Kein Match

```
FindLongestMatch wird getestet
s1: uvwxy s2:rstqz
pos in s1: 0 pos in s2: 0
Longest String:
```

2)Bsp1

```
FindLongestMatch wird getestet
s1: abcdefg s2:adeb
pos in s1: 4 pos in s2: 2
Longest String: de
```

3)Bsp2

```
FindLongestMatch wird getestet
s1: uvwxy s2:rstqvwxz
pos in s1: 2 pos in s2: 5
Longest String: vwx
```

2)Lösungsidee:

Hier muss der schon besprochene BruteForce Algorithmus erweitert werden. Hierbei muss man einfach nur ein paar If Anweisungen hinzufügen. Einmal muss man überprüfen ob der aktuelle char im String s gleich „_“ ist. Falls ja soll i und count erhöht werden. Count soll dann am Ende beim herausfinden der Stelle des Matches subtrahiert werden. Ebenfalls muss man dann auch überprüfen ob p(j) = „?“ ist. Falls ja gilt dies direkt als match.

Testfälle:

1)

```
BruteForcePatternSearchingExtended wird getestet
s: Land der Berge p: Berg
Match gefunden an Stelle: 10
```

2)

```
BruteForcePatternSearchingExtended wird getestet
s: Land der Berge p: B??g
Match gefunden an Stelle: 10
```

3)

```
BruteForcePatternSearchingExtended wird getestet
s: Land der Berge p: B??t
Match gefunden an Stelle: 0
```

4)

```
BruteForcePatternSearchingExtended wird getestet
s: ab_cde p: ab?de
Match gefunden an Stelle: 1
```

5)

```
BruteForcePatternSearchingExtended wird getestet
s: ab____c___de p: ab?de
Match gefunden an Stelle: 1
```

Quellcode:

```
PROGRAM commonSubstring;

  PROCEDURE Swap(VAR s1, s2 : STRING);
    VAR
      h : STRING;
  BEGIN
    h := s1;
    s1 := s2;
    s2 := h;
  END;

  PROCEDURE FindLongestMatchv2(s1, s2 : STRING; VAR sub : STRING; VAR start1, sta
rt2 : INTEGER);
    VAR
      sLen, pLen, i, j, count, longest, z, k : INTEGER;
      consec,swapped : BOOLEAN;
  BEGIN
    count := 0;
    longest := 0;
    swapped := FALSE;
    IF Length(s1) < Length(s2) THEN BEGIN
      Swap(s1, s2);
      swapped := TRUE;
    END;
    sLen := Length(s1);
    pLen := Length(s2);
    i := 1;
    WHILE (i <= sLen) AND (i + pLen - 1 <= sLen) DO BEGIN
      j := 1;
      WHILE (j <= pLen) DO BEGIN
        k := j;
        count := 0;
        WHILE (k <= plen) DO BEGIN
          IF (s1[i + k - j] = s2[k]) THEN BEGIN
            Inc(count);
            consec := TRUE;
          END ELSE BEGIN
            consec := FALSE;
          END;
          IF (count > longest) AND (consec = FALSE) THEN BEGIN
              longest := count;
              start1 := i+k-j-longest;
              start2 := k-longest;
```

```
            count := 0;
          END; (* IF *)
          Inc(k);
        END; (* WHILE *)
        Inc(j);
      END;
      Inc(i);
    END;
    IF (swapped = TRUE) THEN BEGIN
      count := start1;
      start1 := start2;
      start2 := count;
      FOR z := start1 TO start1+longest-1 DO BEGIN
        sub := Concat(sub,s2[z])
      END; (* FOR *)
    END ELSE BEGIN
      FOR z := start1 TO start1+longest-1 DO BEGIN
        sub := Concat(sub,s1[z])
      END; (* FOR *)
    END;
  END;

  FUNCTION BruteForceLR2(s, p : STRING) : INTEGER;
  VAR
    i, j, n, m, count : INTEGER;
  BEGIN
    m := Length(p);
    n := Length(s);
    i := 1; j := 1;
    count := 0;
    WHILE (i <= n) AND (j <= m) DO BEGIN
      IF (s[i] = '_') THEN BEGIN
        Inc(i);
        Inc(count);
      END ELSE IF (s[i] = p[j]) OR (p[j] = '?') THEN BEGIN
        Inc(i);
        Inc(j);
      END ELSE BEGIN
        i := i - j + 2;
        j := 1;
      END;
    END;
    IF j > m THEN
      BruteForceLR2 := i - j + 1 - count
    ELSE
```

```
      BruteForceLR2 := 0;
  END;

VAR
  start1,start2 : INTEGER;
  sub,s1,s2 : STRING;
BEGIN (* commonSubstring *)
  start1 := 0;
  start2 := 0;
  s1 := 'uvwxy';
  s2 := 'rstqvwxz';
  WriteLn('FindLongestMatch wird getestet');
  WriteLn('s1: ',s1, ' s2:', s2);
  FindLongestMatchv2(s1,s2,sub,start1,start2);
  WriteLn('pos in s1: ',start1,' pos in s2: ',start2);
  WriteLn('Longest String: ',sub);

  WriteLn;
  WriteLn('BruteForcePatternSearchingExtended wird getestet');
  s1 := 'ab____c___de';
  s2 := 'ab?de';
  WriteLn('s: ',s1,' p: ',s2);
  WriteLn('Match gefunden an Stelle: ',BruteForceLR2(s1,s2));
END. (* commonSubstring *)
```