Freie Universität Berlin

Sortieralgorithmen

Teil II (Beispiel)

2012

Prof. Dr. Margarita Esponda
Freie Universität Berlin

Freie Universität Berlin

Welches ist die längste Zeichenfolge, die sich wiederholt?

a a b c f d e s a b a b c d f e r f c s d e a e d a b c f d e s a b e d a

a a b c f d e s a b a b c d f e r f c s d e a e d a b c f d e s a b e d a

Freie Universität Berlin

Anwendungen

- Linguistik
- Bioinformatik (DNA-Analyse)
- Datenkompression
- Untersuchung von Plagiaten
- Antiviren-Software
- Musik-Analyse
- usw.

Freie Universität Berlin

Lösung mit Brute-Force

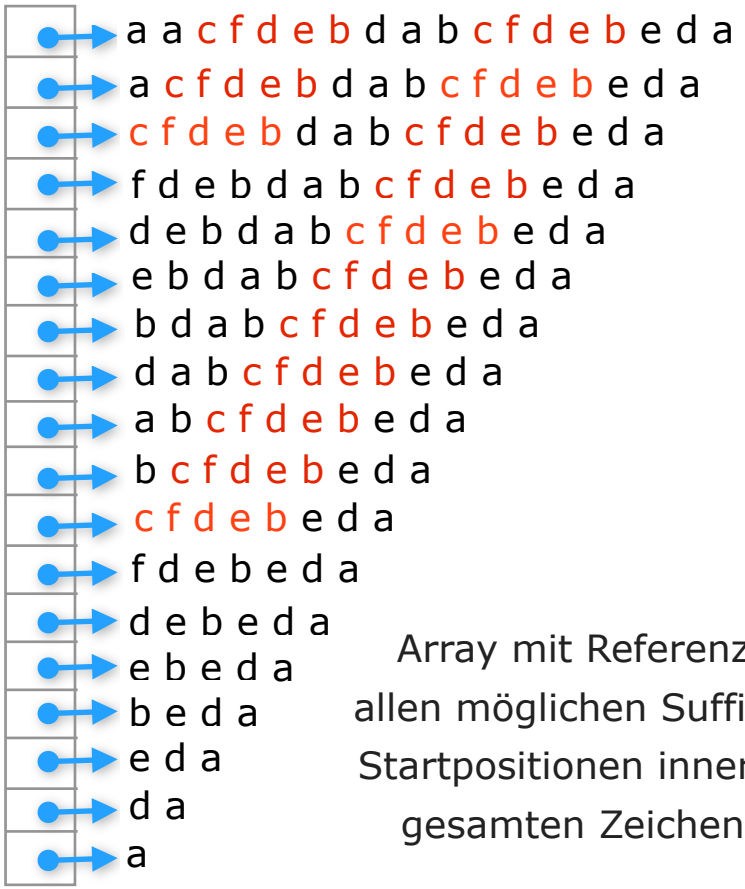
Für alle Startpositionen (i, j) müssen wir das längste Präfix finden.

Anzahl der i, j Kombinationen = (n-1) + (n-2) + … + 2 + 1 = $O(n^2)$

Freie Universität Berlin

Lösung mit Sortieralgorithmen

0	a a c f d e b d a b c f d e b e d a
1	a c f d e b d a b c f d e b e d a
2	c f d e b d a b c f d e b e d a
3	f d e b d a b c f d e b e d a
4	d e b d a b c f d e b e d a
5	e b d a b c f d e b e d a
6	b d a b c f d e b e d a
7	d a b c f d e b e d a
8	a b c f d e b e d a
9	b c f d e b e d a
10	c f d e b e d a
11	f d e b e d a
12	d e b e d a
13	e b e d a
14	b e d a
15	e d a
16	d a
17	a



Array mit Referenzen auf allen möglichen Suffixen oder Startpositionen innerhalb der gesamten Zeichenketten

Freie Universität Berlin

Lösung mit Sortieralgorithmen

17	a
0	a a c f d e b d a b c f d e b e d a
8	a b c f d e b e d a
1	a c f d e b d a b c f d e b e d a
9	b c f d e b e d a
6	b d a b c f d e b e d a
14	b e d a
10	c f d e b e d a
2	c f d e b d a b c f d e b e d a
16	d a
7	d a b c f d e b e d a
4	d e b d a b c f d e b e d a
12	d e b e d a
5	e b d a b c f d e b e d a
13	e b e d a
15	e d a
11	f d e b e d a
3	f d e b d a b c f d e b e d a

Die Suffixe werden sortiert

Freie Universität Berlin

Lösung mit Sortieralgorithmen

17	a
0	a a c f d e b d a b c f d e b e d a
8	a b c f d e b e d a
1	a c f d e b d a b c f d e b e d a
9	b c f d e b e d a
6	b d a b c f d e b e d a
14	b e d a
10	c f d e b e d a
2	c f d e b d a b c f d e b e d a
16	d a
7	d a b c f d e b e d a
4	d e b d a b c f d e b e d a
12	d e b e d a
5	e b d a b c f d e b e d a
13	e b e d a
15	e d a
11	f d e b e d a
3	f d e b d a b c f d e b e d a

Die Präfix-Länge jeder zwei benachbarten Suffixe werden verglichen und der längste Präfix gesucht.

Freie Universität Berlin

Implementierung

```
def generate_suffixes(A):
    """  Find all suffixes of the String A """
    n = len(A)
    suffixes = [0 for i in range(len(A))]

    for i in range(n):
        suffixes[i] = A[i:n]

    return suffixes
```

```
0  a a c f d e b d a b c f d e b e d a
1  a c f d e b d a b c f d e b e d a
2  c f d e b d a b c f d e b e d a
3  f d e b d a b c f d e b e d a
4  d e b d a b c f d e b e d a
5  e b d a b c f d e b e d a
6  b d a b c f d e b e d a
7  d a b c f d e b e d a
8  a b c f d e b e d a
9  b c f d e b e d a
10 c f d e b e d a
11 f d e b e d a
12 d e b e d a
13 e b e d a
14 b e d a
15 e d a
16 d a
17 a
```

Implementierung

```
def find_longest_seq(B):
    max = 0

    for i in range(len(B)-1):
        min_len = min(len(B[i]),len(B[i+1]))
        j = 0
        while j<min_len and B[i][j]==B[i+1][j]:
            j += 1
        if j > max:
            max = j
            if j == min_len:
                seq = B[i][:j+1]
            else:
                seq = B[i][:j]
    return seq
```

Implementierung

```
def LRS_Algorithmus(A):
        B = generate_suffixes(A)
        B.sort()
        seq = find_longest_seq(B)
        return seq
```