Practicum 3

21-01-2015

V1.0

Thimo van Leeuwen: 500689074

Tim Kroon: 500696505

IS203

Inhoudsopgave

Inleiding 3
A. Implementatie van substring in programmeertalen 4
Java 4
Python 5
Regular expressions 9
De regular expression 9
Tests 9
De positieve test gevallen 9
De negatieve test gevallen 10

Inleiding

Deze opdracht bestaat uit drie onderdelen. In onderdeel 1 moesten we kijken hoe het substring algoritme van Java is geïmplementeerd werkte en we moesten van een anderen programmeertaal uitzoeken hoe daar het substring algoritme is geïmplementeerd, wij hebben als tweede programmeer taal Python gekozen omdat we de source code daarvan konden vinden.

In onderdeel 2 moesten we het Knuth-Morris-Prat en het Boyer Moore algoritme met elkaar vergelijken. We moesten 10 zelf gekozen woorden opzoeken in het gedicht “Mei” van Herman Gorter. Ook moesten we bijhouden hoe vaak en of de woorden voor kwamen.

In onderdeel 3 moesten we een RegEx maken voor het controleren van telefoon nummers. Hier ging het om verschillende soorten nummer zoals een nummer dat begint met 06, +316 of 00316 en het nummer 112 en ook 0900- en 0800-nummers

A. Implementatie van substring in programmeertalen

Java

De IndexOf in Java maakt gebruik van een brute force methode.

```
static int indexOf(char[] source, int sourceOffset, int sourceCount,
        char[] target, int targetOffset, int targetCount,
        int fromIndex) {
    if (fromIndex >= sourceCount) {
        return (targetCount == 0 ? sourceCount : -1);
    }
    if (fromIndex < 0) {
        fromIndex = 0;
    }
    if (targetCount == 0) {
        return fromIndex;
    }

    char first = target[targetOffset];
    int max = sourceOffset + (sourceCount - targetCount);

    for (int i = sourceOffset + fromIndex; i <= max; i++) {
        /* Look for first character. */
        if (source[i] != first) {
            while (++i <= max && source[i] != first);
        }

        /* Found first character, now look at the rest of v2 */
        if (i <= max) {
            int j = i + 1;
            int end = j + targetCount - 1;
            for (int k = targetOffset + 1; j < end && source[j]
                    == target[k]; j++, k++);

            if (j == end) {
                /* Found whole string. */
                return i - sourceOffset;
            }
        }
    }
    return -1;
}
```

Python

De Python substring methode is geschreven in de programmeertaal C. Python maakt gebruik van de Boyer-Moore algorithme.

```
/* stringlib: find/index implementation */

#ifndef STRINGLIB_FIND_H
#define STRINGLIB_FIND_H

#ifndef STRINGLIB_FASTSEARCH_H
#error must include "stringlib/fastsearch.h" before including this module
#endif

Py_LOCAL_INLINE(Py_ssize_t)
stringlib_find(const STRINGLIB_CHAR* str, Py_ssize_t str_len,
               const STRINGLIB_CHAR* sub, Py_ssize_t sub_len,
               Py_ssize_t offset)
{
    Py_ssize_t pos;

    if (str_len < 0)
        return -1;
    if (sub_len == 0)
        return offset;

    pos = fastsearch(str, str_len, sub, sub_len, -1, FAST_SEARCH);

    if (pos >= 0)
        pos += offset;

    return pos;
}

Py_LOCAL_INLINE(Py_ssize_t)
stringlib_rfind(const STRINGLIB_CHAR* str, Py_ssize_t str_len,
                const STRINGLIB_CHAR* sub, Py_ssize_t sub_len,
                Py_ssize_t offset)
{
    Py_ssize_t pos;

    if (str_len < 0)
        return -1;
    if (sub_len == 0)
        return str_len + offset;

    pos = fastsearch(str, str_len, sub, sub_len, -1, FAST_RSEARCH);

    if (pos >= 0)
        pos += offset;

    return pos;
}
```

```
/* helper macro to fixup start/end slice values */
#define ADJUST_INDICES(start, end, len)         \
   if (end > len)                           \
      end = len;                            \
   else if (end < 0) {                      \
      end += len;                           \
      if (end < 0)                          \
         end = 0;                           \
   }                                        \
   if (start < 0) {                         \
      start += len;                         \
      if (start < 0)                        \
         start = 0;                         \
   }

Py_LOCAL_INLINE(Py_ssize_t)
stringlib_find_slice(const STRINGLIB_CHAR* str, Py_ssize_t str_len,
                const STRINGLIB_CHAR* sub, Py_ssize_t sub_len,
                Py_ssize_t start, Py_ssize_t end)
{
   ADJUST_INDICES(start, end, str_len);
   return stringlib_find(str + start, end - start, sub, sub_len, start);
}

Py_LOCAL_INLINE(Py_ssize_t)
stringlib_rfind_slice(const STRINGLIB_CHAR* str, Py_ssize_t str_len,
                 const STRINGLIB_CHAR* sub, Py_ssize_t sub_len,
                 Py_ssize_t start, Py_ssize_t end)
{
   ADJUST_INDICES(start, end, str_len);
   return stringlib_rfind(str + start, end - start, sub, sub_len, start);
}

#ifdef STRINGLIB_WANT_CONTAINS_OBJ

Py_LOCAL_INLINE(int)
stringlib_contains_obj(PyObject* str, PyObject* sub)
{
   return stringlib_find(
      STRINGLIB_STR(str), STRINGLIB_LEN(str),
      STRINGLIB_STR(sub), STRINGLIB_LEN(sub), 0
      ) != -1;
}

#endif /* STRINGLIB_WANT_CONTAINS_OBJ */

#if STRINGLIB_IS_UNICODE

/*
This function is a helper for the "find" family (find, rfind, index,
rindex) of unicodeobject.c file, because they all have the same
behaviour for the arguments.

It does not touch the variables received until it knows everything
is ok.

Note that we receive a pointer to the pointer of the substring object,
```

```
so when we create that object in this function we don't DECREF it,
because it continues living in the caller functions (those functions,
after finishing using the substring, must DECREF it).
*/

Py_LOCAL_INLINE(int)
_ParseTupleFinds (PyObject *args, PyObject **substring,
                  Py_ssize_t *start, Py_ssize_t *end) {
    PyObject *tmp_substring;
    Py_ssize_t tmp_start = 0;
    Py_ssize_t tmp_end = PY_SSIZE_T_MAX;
    PyObject *obj_start=Py_None, *obj_end=Py_None;

    if (!PyArg_ParseTuple(args, "O|OO:find", &tmp_substring,
         &obj_start, &obj_end))
        return 0;

    /* To support None in "start" and "end" arguments, meaning
       the same as if they were not passed.
    */
    if (obj_start != Py_None)
        if (!_PyEval_SliceIndex(obj_start, &tmp_start))
            return 0;
    if (obj_end != Py_None)
        if (!_PyEval_SliceIndex(obj_end, &tmp_end))
            return 0;

    tmp_substring = PyUnicode_FromObject(tmp_substring);
    if (!tmp_substring)
        return 0;

    *start = tmp_start;
    *end = tmp_end;
    *substring = tmp_substring;
    return 1;
}
```

B. Vergelijken van algoritmes

Woord	BM aantal	BM vergelijk	KMP aantal	KMP vergelijk
het	552	186142	552	178106
meisje	12	202484	12	178106
programmeren	0	201423	0	178106
banaan	0	195853	0	178106
gras	32	183072	32	178106
Holland	5	187095	5	178106
mei	22	186618	22	178106
snuisterijenkramen	1	201188	1	178106
laat	72	186519	72	178106
blonde	13	206024	13	178106

Onze resultaten komen niet overheen met de gegevens uit het boek. Bij het Boyer-Moorse algoritme zijn het aantal vergelijkingen bijna overal hetzelfde, ook als het woord maar 1 keer voor komt.

Bij het Knuth-Morris-Pratt algoritme zijn het aantal vergelijkingen elke keer hetzelfde. Wij denken dat dit komt doordat hij elke letter 1 voor 1 afgaat.

C. Regular expressions

De regular expression

```
Pattern p = Pattern.compile(""
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers
```

Tests

De positieve test gevallen

06-12345678 is een geldig mobiel nummer

Dit nummer test of een nummer met een – wordt geaccepteerd

```
Pattern p = Pattern.compile(""
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers

Matcher m = p.matcher("06-12345678");
boolean b = m.matches();
System.out.println(b);
```

+31201234567 is een geldig mobiel nummer

Dit nummer test of een nummer met +31 wordt geaccepteerd

```
Pattern p = Pattern.compile(""
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers

Matcher m = p.matcher("+31201234567");
boolean b = m.matches();
System.out.println(b);
```

0900-4356 is een geldig 0900-nummer

Dit nummer test of een 0900-nummer met een – wordt geaccepteerd

```
Pattern p = Pattern.compile("
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers

Matcher m = p.matcher("0900-4356");
boolean b = m.matches();
System.out.println(b);
```

De negatieve test gevallen

0612345323432 is een ongeldig telefoon nummer omdat het te lang is

Dit nummer test of een te lang nummer wordt geaccepteerd

```
Pattern p = Pattern.compile("
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers

Matcher m = p.matcher("0612345323432");
boolean b = m.matches();
System.out.println(b);
```

S&S is een leuk vak! Is een ongeldig telefoon nummer omdat het alleen maar letter bevat

Dit “nummer” test of er letters worden geaccepteerd

```
Pattern p = Pattern.compile("
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers

Matcher m = p.matcher("S&S is een leuk vak!");
boolean b = m.matches();
System.out.println(b);
```

00006372893 is een ongeldig nummer omdat het begint met 0000

Dit nummer test of een nummer met een netnummer van 0000 wordt geaccepteerd

```
Pattern p = Pattern.compile(""
        + "([+]316|00316|06)(-)?[1-9][0-9]{7}|" //het checken voor een mobiel nummer
        + "([+]31|0031|0)[1-9]{2}[0-9](-)?[1-9][0-9]{6}|" // het checken voor een nummer met een 4-cijferig net nummer
        + "([+]31|0031|0)[1-9][0-9](-)?[1-9][0-9]{6}|"  // het checken voor een nummer met een 3-cijferig net nummer
        + "112|" // het checken van het noodnummer
        + "0900(-)?[0-9]{4}|" //  het checken van 0900-XXXX nummers
        + "0800(-)?[0-9]{4}"); //  het checken van 0800-XXXX nummers

Matcher m = p.matcher("00006372893");
boolean b = m.matches();
System.out.println(b);
```