td1a_cython_edit_correction

July 1, 2022

1 1A.soft - Calcul numérique et Cython - correction

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Exercice : python/C appliqué à une distance d'édition

On reprend la fonction donnée dans l'énoncé.

[2]:
```
def distance_edition(mot1, mot2):
    dist = { (-1,-1): 0 }
    for i,c in enumerate(mot1) :
        dist[i,-1] = dist[i-1,-1] + 1
        dist[-1,i] = dist[-1,i-1] + 1
        for j,d in enumerate(mot2) :
            opt = [ ]
            if (i-1,j) in dist :
                x = dist[i-1,j] + 1
                opt.append(x)
            if (i,j-1) in dist :
                x = dist[i,j-1] + 1
                opt.append(x)
            if (i-1,j-1) in dist :
                x = dist[i-1,j-1] + (1 if c != d else 0)
                opt.append(x)
            dist[i,j] = min(opt)
    return dist[len(mot1)-1,len(mot2)-1]

%timeit distance_edition("idstzance","distances")
```

```
188 µs ± 28.6 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

1.2 solution avec notebook

Les préliminaires :

[3]:
```
%load_ext cython
```

Puis :

[4]:
```
%%cython --annotate
cimport cython

def cidistance_edition(str mot1, str mot2):
    cdef int dist [500][500]
    cdef int cost, c
    cdef int l1 = len(mot1)
    cdef int l2 = len(mot2)

    dist[0][0] = 0
    for i in range(l1):
        dist[i+1][0] = dist[i][0] + 1
        dist[0][i+1] = dist[0][i] + 1
        for j in range(l2):
            cost = dist[i][j+1] + 1
            c    = dist[i+1][j] + 1
            if c < cost : cost = c
            c = dist[i][j]
            if mot1[i] != mot2[j] : c += 1
            if c < cost : cost = c
            dist[i+1][j+1] = cost
    cost = dist[l1][l2]
    return cost
```

[4]:
```
<IPython.core.display.HTML object>
```

[5]:
```
mot1, mot2 = "idstzance","distances"
%timeit cidistance_edition(mot1, mot2)
```

```
16.9 µs ± 3.47 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

1.3 solution sans notebook

[6]:
```
import sys
from pyquickhelper.loghelper import run_cmd

code = """
def cdistance_edition(str mot1, str mot2):
    cdef int dist [500][500]
    cdef int cost, c
    cdef int l1 = len(mot1)
    cdef int l2 = len(mot2)

    dist[0][0] = 0
    for i in range(l1):
        dist[i+1][0] = dist[i][0] + 1
        dist[0][i+1] = dist[0][i] + 1
        for j in range(l2):
            cost = dist[i][j+1] + 1
            c    = dist[i+1][j] + 1
            if c < cost : cost = c
            c = dist[i][j]
            if mot1[i] != mot2[j] : c += 1
```

```
            if c < cost : cost = c
            dist[i+1][j+1] = cost
    cost = dist[l1][l2]
    return cost
"""

name = "cedit_distance"
with open(name + ".pyx","w") as f : f.write(code)

setup_code = """
from distutils.core import setup
from Cython.Build import cythonize
setup(
    ext_modules = cythonize("__NAME__.pyx",
                            compiler_directives={'language_level' : "3"})
)
""".replace("__NAME__",name)

with open("setup.py","w") as f:
    f.write(setup_code)

cmd = "{0} setup.py build_ext --inplace".format(sys.executable)

out,err = run_cmd(cmd)
if err is not None and err != '':
    raise Exception(err)

import pyximport
pyximport.install()
import cedit_distance

from cedit_distance import cdistance_edition

mot1, mot2 = "idstzance","distances"
%timeit cdistance_edition(mot1, mot2)
```

```
11.4 µs ± 1.93 µs per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

La version Cython est 10 fois plus rapide. Et cela ne semble pas dépendre de la dimension du problème.

[7]:
```
mot1 = mot1 * 10
mot2 = mot2 * 10
%timeit distance_edition(mot1,mot2)
%timeit cdistance_edition(mot1, mot2)
```

```
11.5 ms ± 561 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
724 µs ± 30 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

[8]:
```
```