GO FOR IT Q2

s336651 大垣慶介

2012 年 2 月 6 日

q4_decode.py のドキュメント

```
============================================================
 q3_decode.py
------------------------------------------------------------
 arguments
    |- (1.targetstring)
        |- default : sony
    |- (2.codestring)
        |- default : from problem
 output
    |- skip[0],index[0]\n .... skip[n],index[n]\n
    |- time= $(hour):$(minute):$(second)
 example
    |- $ q3_decode.py
    |- $ q3_decode.py sony ssoonnyy
------------------------------------------------------------
 for Python 2.7
 K.Ogaki(ogaki@iis.u-tokyo.ac.jp)
 2012/02/10
------------------------------------------------------------
 This program is released under GPL.
 http://www.opensource.jp/gpl/gpl.ja.html.euc-jp
============================================================
```

表 1: 環境

言語	python2.7
動作確認	Linux(Ubuntu10.10)
言語	C++
動作確認	Windows 7 64bit / Linux(Ubuntu10.10)
コンパイラ	Visual C++ / gcc 4.4.5
最適化	/O2 / -O3

仕様

2 つのプログラムを用意した。python2.7 による q3_decode.py と c++による q3_fast.cpp である。それぞれ、上記のドキュメントのように、$q3_decode.py と入力することで問題分のランダム文字列から “sony” の文字列を検索する。また、 $q3_decode.py targetstring codestring と入力することで codestring 中から targetstring を検索する。引数が 0 の場合ドキュメントを出力する。実行環境は表のとおりである。

アルゴリズム

q3_decode.py

```
list1 = [ i for i in xrange(len(codestr)) if codestr[i]==targetstr[0] ]
list2 = [ i for i in xrange(len(codestr)) if codestr[i]==targetstr[1] ]

for i1 in list1:
    for i2 in list2:
        if(istrueindex(i2, i2-i1, targetstr[2:])):
            anslist.append
```

単純な探索であり、1 文字目、2 文字目のリストを作り、そこから skip:i2-i1 と 1 文字目:i1 を使ってそれが正しい skip と 1 文字目かを istrueindex() で判断している。

```
def istrueindex(now, skip, targetstr):
    if( len(targetstr)==0 ):
        return True
    if( (now+skip)<0 or (now+skip)>=len(codestr)):
        return False
    if(codestr[now+skip]==targetstr[0]):
        return istrueindex(now+skip, skip, targetstr[1:])
    else:
        return False
```

istrueindex() は再帰で、tagetstring の先頭が見つかれば先頭を抜いた targetstring[1:] の先頭のチェックを行う。渡される targetstring の長さが 0, つまりすべて見つかったら True である。

q3_fast.cpp

```
std::list<long long> list1;
std::list<long long> listlast;

for(std::list<long long>::iterator it1 = list1.begin(); it1 != list1.end(); it1++)

    for(std::list<long long>::iterator itlast = listlast.begin(); itlast != lis
if( (*itlast-*it1) % skiptime  != 0)continue枝刈り;//
else{
    // search //
}
```

アルゴリズムの変更を行なったのは skip と 1 文字目の候補が先のプログラムでは 1 文字目と 2 文字目から算出していたのを 1 文字目と最後の文字から算出している。これによるメリットは以下。

- codestring 中の index について、もし targetstring が見つかるなら、1 文字目の index1 と n 文字目の indexn には (indexn-index1)%(n-1)==0 の関係が成り立つはずである。これによって (1/(targetstring.size()-1)) に枝刈りが出来る。
- istrueindex において、検索文字が codestring の範囲を超えることがない (最後の文字は存在している保証があるので)。

また、その他には c++で書き換えることによって、最適化の恩恵とプロファイラの恩恵にあずかった。プロファイラの結果から、istrueindex() を search() の中のループにすることで高速化をはかった。

出力 (時間のみ)

出力行数が多いので各プログラムの実行時間と見つけた答えの数のみを記す。

```
$ ./q3_decoding.py
time= 0:01:37.641841
ansnum = 66375
$ ./q3_fast
time= 3.1
ansnum = 66375
```

python 版は h:minute:second,c++版は second のみである約 31.5 倍の高速化がなされている。