電子計算機入門

池田, 大輔
九州大学情報基盤センター

電子計算機入門
第7回

池田 大輔

daisuke@cc.kyushu-u.ac.jp

情報基盤センター

目次

- 第 1 回課題の解説
- 配列操作 (再び)
- 関数
- 前回課題の回答例

最長共通部分文字列問題

- 入力：２つの文字列
  - 引数で与えるものとする
- 出力：共通部分文字列で最長のものすべて
- 例

最長共通部分文字列問題

- 入力：２つの文字列
  - 引数で与えるものとする
- 出力：共通部分文字列で最長のものすべて
- 例
  - $AGACCTC, GCGACCTGTA$

最長共通部分文字列問題

- 入力：２つの文字列
  - 引数で与えるものとする
- 出力：共通部分文字列で最長のものすべて
- 例
  - $AGACCTC, GCGACCTGTA$

最長共通部分文字列問題

- 入力：２つの文字列
  - 引数で与えるものとする
- 出力：共通部分文字列で最長のものすべて
- 例
  - $AGACCTC$, $GCGACCTGTA$
  - 他に A, C, G, T, CT, AC なども共通部分文字列

アルゴリズム(1)

２つの入力引数を変数 $str1$ と $str2$ に代入

$str1$ のすべての部分文字列を配列 $ss1$ に代入

$str2$ のすべての部分文字列を配列 $ss2$ に代入

結果を格納する空配列 res

$ss1$ の各要素 e に対し

　　e が $ss2$ にはいっていて res にはいっていない

　　　　res に e を追加

res の各要素 e から最も長い文字列を抽出

解答例：アルゴリズム(1)

```
import sys
s1=sys.argv[1]
s2=sys.argv[2]
ss1=[]
for i in range(len(s1)):
    for j in range(i,len(s1)):
        # s1 の部分文字列を ss1 に格納
        if s1[i:j+1] not in ss1:
            ss1.append(s1[i:j+1])
ss2=[]
for k in range(len(s2)):
    for l in range(k,len(s2)):
        # s2 の部分文字列を ss2 に格納
        if s2[k:l+1] not in ss2:
            ss2.append(s2[k:l+1])
```

解答例(Cont.)

```
common = [] # 共通部分文字列とその長さを格納
for i in ss1:
    if i in ss2: common.append((len(i), i))
common.sort() # 長さでソート
common.reverse() # 大きい順に並べかえ
#common=[(3,"abc"),(3,"bcd"),(2,"ab"),....]

MAX = common[0][0] # 最初の最初の要素が最大の長さ
res = [] # 結果を収める配列
for (l, i) in common:
    if l == MAX: res.append(i)
print res
```

問題点の整理

- `max` や `reverse, sort` の使い方
- 同じようなコードが複数ある
  - 関数により整理
- 効率
  - 検索による効率化 ($ss2$ を作らない) ←次回
  - ループの数

ループの数

- 入力文字列の長さを n と置く

```
for i in range(len(s1)):
    for j in range(i,len(s1)):
```

だいたい n^2 回実行される

ループの数

- 入力文字列の長さを n と置く

```
for i in range(len(s1)):#← n 回
    for j in range(i,len(s1)):#← n 回
```

だいたい n^2 回実行される

ループの数

- 入力文字列の長さを n と置く

```
for i in range(len(s1)):#← n 回
    for j in range(i,len(s1)):#← n 回
```

だいたい n^2 回実行される

- 4 重ループだと n^4 程度
- 2 重ループを独立に 2 個だと $2n^2$ 程度

ループの数

- 入力文字列の長さを n と置く

```
for i in range(len(s1)):#← n 回
    for j in range(i,len(s1)):#← n 回
```

だいたい n^2 回実行される

- 4 重ループだと n^4 程度
- 2 重ループを独立に 2 個だと $2n^2$ 程度
- 長さが 1000 の文字列の場合
  - 4 重ループは $10^{12} = 1$ 兆回
  - 2 重ループは $2 \times 10^6 = 2$ 百万回
  - 最近の CPU は数ギガヘルツ=20 億回／秒

アルゴリズム(2)

- $ss2$ を作るけど保存しない!

---

2 つの入力引数を変数 $str1$ と $str2$ に代入

$str1$ のすべての部分文字列を配列 $ss1$ に代入

結果を格納する空配列 res

$0 \leq i < j \leq len(str2)$ なる i, j に対し

$len(str2[i:j+1]) < len(res[0])$ なら何もしない

$len(str2[i:j+1]) == len(res[0])$ なら

res に $str2[i:j+1]$ を追加

$len(str2[i:j+1]) > len(res[0])$ なら

$res = [str2[i:j+1]]$

解答例：アルゴリズム(2)

```
import sys
s1=sys.argv[1]
s2=sys.argv[2]
ss1=[]
for i in range(len(s1)):
    for j in range(i,len(s1)):
        tmp=s1[i:j+1]
        if tmp not in ss1: ss1.append(tmp)
        # s1 の部分文字列 tmp を ss1 に格納
```

解答例(Cont.)

```
res=[""] # 結果を収める配列
# res = [] だと len(res[0]) がエラーになる
for k in range(len(s2)):
    for l in range(k,len(s2)):
        tmp=s2[k:l+1]
        # s2 の部分文字列 tmp を生成
        if tmp in res: # s1 の部分文字列ならば
            # いままでの最長と長さを比較
            if len(tmp)>len(res[0]):
                del res[0:]
                res.append(tmp)
            elif len(tmp)==len(res[0]):
                res.append(tmp)
print res
```

配列操作(再び)

- `max()`, `min()` 関数
  - 配列の「最大」「最小」を返す
- `sort()`, `reverse()` メソッド
  - 配列を並べ替える、逆順に並べかえる

実習：配列要素の大小関係

- 以下のプログラムを入力し、実行してみなさい

```
array = [7, -1, 3, 0]
array.sort()
print array
array = ["baa", "accc", "abc", "bab"]
array.sort()
print array
```

文字列配列のソート

- 配列を要素の頻度や長さなど二次的な情報で並べかえ
  - 二次的な情報を含めた多次元配列にする
  - 二次的な情報を先に配置する

文字列配列のソート

- 配列を要素の頻度や長さなど二次的な情報で並べかえ
  - 二次的な情報を含めた多次元配列にする
  - 二次的な情報を先に配置する

```
array = [(3, "baa"), (4, "accc"),
        (3, "abc"), (3, "bab")]
# (長さ, 文字列) の配列
array.sort() # 長さの小さい順
array.reverse()
print array
```

```
[(4,'accc'),(3,'bab'),(3,'baa'),(3,'abc')]
```

実習：文字列配列のソート

- 文字列を要素とする配列から、長さが最大のものを (1 つ) 出力せよ

実習：文字列配列のソート

- 文字列を要素とする配列から、長さが最大のものを (1 つ) 出力せよ

```
array=["baa","accc","abc","bab"]#入力
new = []
for i in array:
    new.append((len(i), i)) # (長さ, 要素)
print max(new)
```

内包表記

- 配列 (リスト) から別の配列を作る
  - 条件を満たす要素をとりだす
  - 要素を加工する
- 数式の集合表現と同じ

  `array` から `new` を作る場合

  e.g., $\texttt{new} = \{(x, 2x) \mid x \in \texttt{array}, x > 3\}$

内包表記の例

- 例：文字列要素の配列から (長さ、文字列) 配列を作る

```
array=["baa","accc","abc","bab"]#入力
new = [(len(i), i) for i in array]
```

→ `array = [(3, "baa"), (4, "accc"), ...`

内包表記の例

- 例：文字列要素の配列から (長さ、文字列) 配列を作る

```
array=["baa","accc","abc","bab"]#入力
new = [(len(i), i) for i in array]
→array = [(3, "baa"), (4, "accc"), ...
```

- 例：文字列要素の配列から長さが 4 以上の要素のみを抽出

```
array=["baa","accc","abc","bab"]#入力
new = [i for i in array if len(i) >= 4]
→array = ["accc"]
```

内包表記の例

- 例：文字列要素の配列から (長さ、文字列) 配列を作る

```
array=["baa","accc","abc","bab"]#入力
new = [(len(i), i) for i in array]
→array = [(3, "baa"), (4, "accc"), ...
```

- 例：文字列要素の配列から長さが４以上の要素のみを抽出

```
array=["baa","accc","abc","bab"]#入力
new = [i for i in array if len(i) >= 4]
→array = ["accc"]
```

- 例：複数の配列から

```
xs = (1, 2, 3, 4, 5)
ys = (9, 8, 7, 6, 5)
z = [(x, y) for x in xs for y in ys if
x*y > 25]
```

実習：内包表記の例

- “algorithm1.py” を share フォルダからコピーし、配列 `common` と `res` 作成部分を内包表記を使って書きかえなさい

実習：内包表記の例

- “algorithm1.py” を share フォルダからコピーし、配列 `common` と `res` 作成部分を内包表記を使って書きかえなさい

```
common = [(len(i), i) for i in ss1
    if i in ss2]
common.sort()
common.reverse()
MAX = common[0][0]
res=[i for (l,i) in common if l==MAX]
print res
```

関数

- プログラムのうち、特定の機能に名前をつけて再利用可能にしたもの
- 入力を受けとり、値を返す (出力する)

```
def function(x, y):
    '''
    ここの文字列は function の説明
    一種のコメント文字列
    '''
    ここに function の動作記述
    function の範囲はインデント
    return(var) # 配列を返してもよい
```

実習：関数の使用例

- さきほどの部分文字列生成部分を関数化しなさい

```
def substrings(s):
    ''' 文字列 s のすべての部分文字列を生成する'''
    result = []
    for i in range(len(s)):
        for j in range(i,len(s)):
            if s[i:j+1] not in result:
                result.append(s[i:j+1])

    return(result) # 関数の定義終了
import sys
s1=sys.argv[1]
s2=sys.argv[2]
ss1=substrings(s1)
ss2=substrings(s2)
common = [] # 以下は同じ
```

今日の課題

- “algorithm1.py” を以下のように修正しなさい
  - 配列 `common` と `res` の生成に内包表記を利用する
  - 部分文字列の生成を関数化する

前回課題の解答例

- ファイルの中身を行番号付きで表示するプログラムを書け

```
import sys
file = open(sys.argv[1], 'r')
num = 0 #行数をカウントするための変数
line = file.readline() # 1 行目
while line:
    print num+1, line, #+1 がミソ
    num += 1
    line = file.readline() #次の行
```

前回課題の解答例

- ファイルの中身を行番号付きで表示するプログラムを書け

```
import sys
file = open(sys.argv[1], 'r')
num = 0 #行数をカウントするための変数
line = file.readline() # 1 行目
while line:
    print num+1, line, #+1 がミソ
    num += 1
    line = file.readline() #次の行
```