

電子計算機入門

池田, 大輔
九州大学附属図書館

<https://hdl.handle.net/2324/2841>

出版情報 : 2004
バージョン :
権利関係 :

電子計算機入門 第11回(07/07)

池田 大輔

`z4id01in@cse.ec.kyushu-u.ac.jp`

附属図書館

目次

第3回レポート課題予告

正規表現

文字列の統合

第3回レポート予告

問題

入力: FASTA 形式のファイル名とモチーフ (後述)

出力: “>” で始まる行の次の行から、次の “>” で始まる行の前の行までを1つの領域としたとき、これらの領域に与えられたモチーフの出現回数

第3回レポート予告

問題

入力: FASTA 形式のファイル名とモチーフ (後述)

出力: “>” で始まる行の次の行から、次の “>” で始まる行の前の行までを1つの領域としたとき、これらの領域に与えられたモチーフの出現回数

「モチーフ」とは

DNA やタンパク質の短い文字列で、多くの種に共通していたり、DNA の制御部分などの重要な部分文字列
ただし、固定された文字列とは限らず「CT の後は C か G か T でその後は ACG である」というように、曖昧さを含んでいる

第3回レポート予告

問題

入力: FASTA 形式のファイル名とモチーフ (後述)

出力: “>” で始まる行の次の行から、次の “>” で始まる行の前の行までを1つの領域としたとき、これらの領域に与えられたモチーフの出現回数

「モチーフ」とは

DNA やタンパク質の短い文字列で、多くの種に共通していたり、DNA の制御部分などの重要な部分文字列
ただし、固定された文字列とは限らず「CT の後は C か G か T でその後は ACG である」というように、曖昧さを含んでいる

このような曖昧さを表現するのに**正規表現**を使う

FASTA形式のファイル

FASTA形式は、遺伝子配列を表現するための形式で

```
>gi | 16127994 : 190 - 255  
ATGAAACGCATTAGCACCCACCAT.....  
>gi | 16127994 : 337 - 2799  
ATGCGAGTGTTGAAGTTCGGCGG.....  
TGGAAAGCAATGCCAGGCAGGGG.....  
GGTGGCGATGATTGAAAAAACCA.....  
...
```

という形式である

'>'で始まる行がDNAの位置などを表し、それ以外の行が遺伝子等を表す

ただし、長い場合は配列部分は適宜改行されている

'>'以外で始まる箇所だけ进行处理する必要があり、この処理にも**正規表現**を使う

方針

まず簡単な場合を考えよう

'>' 以外の行はそのまま出力する

つまり、本来は1本の長い遺伝子だとしても、細かく分割して出力する

ある行が'>'で始まるかどうかを判断できればよい

正規表現

正規表現 (Regular Expression) とは

複数 (無限個でも可) の語や文字列を簡単に表す手段

例：“ACCTで始まりTで終わる配列”

多くのプログラミング言語 (Perl や Ruby など) で利用可能

ただし、記法が異なる場合もある

文字列処理に必須の機能

正規表現の記法

“.”(ピリオド): 任意の一文字

ただし、改行文字とは一致しない

“*”: 直前の正規表現の 0 回以上の繰り返し

"ACCT.*T"は"ACCT"で始まり"T"で終わる文字列全部

注意: "ACCTT"も含まれる

“^”で文字列の先頭

“^>gi”で“>gi”が先頭にある時に一致

“[]”で複数文字からの選択

[abc] で abc のうちどれか 1 文字と一致

具体例

「CTの後はCかGかTでその後はACGである」

“CT[CGT]ACG”

「行の先頭にある“>gi”」

“^>gi.*”

正規表現の使い方

```
import re # re モジュール  
pat = re.compile("正規表現") # パターン作成  
# "文字列"中のパターンを探す  
m = pat.search("文字列")
```

search() は**マッチオブジェクト**を返す

与えた文字列に正規表現に一致する部分があれば真、
そうでなければ偽となる

実習

以下のプログラムを作成し、実行結果を確認しなさい。
また、*line* の値や `compile()` の引数を変化させ、実行結果を確認しなさい。

```
line = "GAACCTGATA"  
pat = re.compile("ACCT.+T")  
m = pat.search(line)  
if m: # 一致したら出力  
    print line  
else: # 一致したら出力  
    print "Not Found"
```

実習：FASTA形式

FASTA形式のファイル名を入力として受けとり、“>gi”で始まる行以外のみを出力するプログラムを作成せよ。

サンプルデータとして授業のWebページに“fasta.txt”を用意した

```
import sys, re # 2つ同時に import
pat = re.compile("^>gi") # パターン作成
f = open(sys.argv[1], 'r')
all = f.readlines() # ファイル読込
for line in all:
    m = pat.search(line) # 照合
    if not m:
        print line,
```

文字列の統合

FASTA 形式では、長い配列は分割されている

'>' で始まる行から次の '>' で始まる行までを 1 本の長い配列にしないといけない

文字列配列の統合：join() を使う

統合すべき個々の文字列をあらかじめ配列に格納しておく

join() は文字列のメソッドなので "" (空文字列) から呼び出す

実習：join()

以下のプログラムを作成し、実行せよ

```
array=["ab", "cd", "ef"]
```

```
seq="" .join(array)
```

```
print seq
```


FASTA 形式の場合...

```
f = open('ファイル名', 'r') # ファイルを開く
alllines に全部の行を読込
seq = [] # 一遺伝子分をまとめるための配列
genes = [] # 各遺伝子を一要素として格納する配列
for 各行に対し:
    if '>' で始まっている:
        配列 seq の中身を統合し、
        genes に追加
        配列 seq = [] # 再初期化
    else:
        配列 seq に現在の行を追加
```

FASTA 形式の場合...

```
'>gi|161... ' # seq=[]  
ATGAAACGCT # seq.append()  
'>gi|161... ' # join&seq=[]  
AACGCATTAG # seq.append()  
AACGCATTAG # seq.append()  
ATAACATTAG # seq.append()  
'>gi|161... ' # seq=[]  
AGAACGCTTA # seq.append()  
TAAGCATTAG # seq.append()  
'>gi|161... ' # seq=[]
```

今日の課題

入力：FASTA 形式のファイル

出力：各遺伝子 (^>gi の次の行から次の^>gi の前の行まで) を1本の文字列として出力するプログラムを作成せよ。

ただし、まだ改行文字が残っているため、以下のような出力になってよい

```
( )  
( ATGA  
)  
( ATGCGA  
TGGAAA  
GGTGGC  
)
```

付録：split()

join() の逆で、規則性のある文字列をその規則で分ける

```
seq="ab:cde:af:dde"
```

```
array=seq.split(':')
```

何で分けるか明記する

array=["ab", "cde", "af", "dde"] となる

付録：正規表現のその他の記法

“+”: 直前の正規表現の **1** 回以上の繰り返し

"ACCT.**+**T"は"ACCT"で始まり1つ以上文字があり"T"で終わる文字列全部

"ACCTT"は含まれない

正規表現 A と B に対し“A|B”で A または B のどちらかに一致する

“\$”で文字列の最後

“[**^**]”で複数文字**以外**からの選択

[**^**abc] で abc 以外の1文字と一致