第 9 回　整列アルゴリズム（バブルソート）

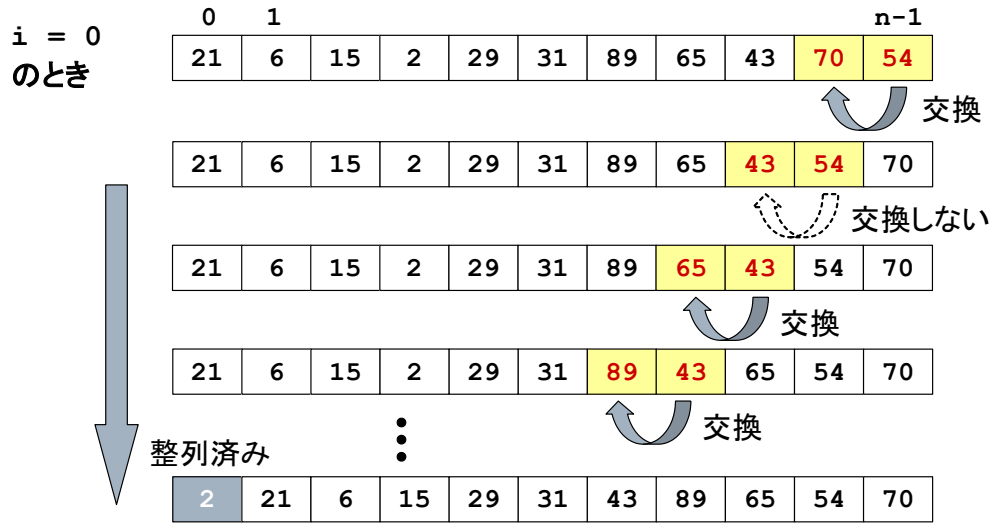
今回と次回は整列アルゴリズムについて学ぶ。

【単純交換ソート（バブルソート）】
リストの要素を昇順（または降順）に並べ替えるためのもっとも単純な方法として、単純交換ソート（またはバブルソート）と呼ばれる方法がある。
単純交換ソートでは、リストの後ろ側から二つの要素を比較して、先頭側が大きければ交換するという操作を、先頭の要素に到達するまで繰り返す。この繰り返し処理を一回実施すると、リスト中の最小値が先頭に置かれ、リストの先頭が「整列済み」となる。同じ処理を「整列済み」でない要素に対して繰り返す。この処理を、リスト中のすべての要素が「整列済み」となるまで繰り返す。小さな値が水底から浮かび上がってくるように見えることから、バブルソートとも呼ばれる。

単純交換ソートの繰り返し部分は、次のようになる。

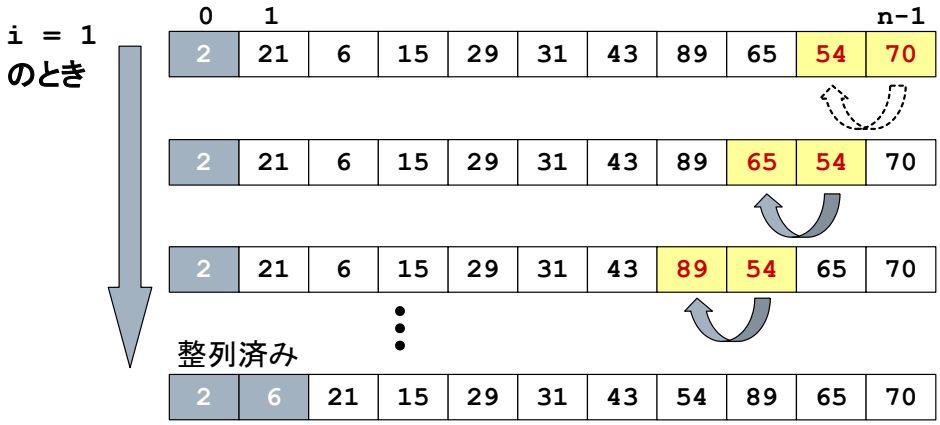
```
for i in range(len(list)):
    list[i], list[i+1], ..., list[n-1] に対して，後ろ側から二つの要素を
    比較して，先頭側が大きければ交換する；
```

図で表すと、このようになる。

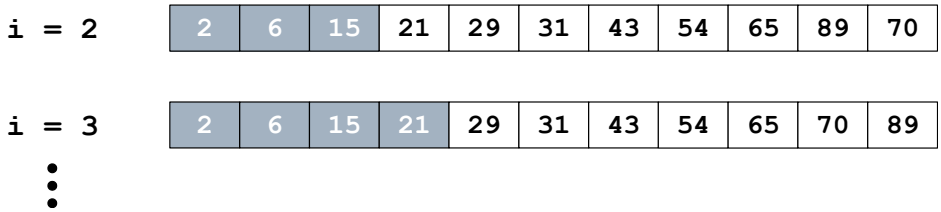


まず，繰り返し処理の 1 回目では，最初に n-1 番目と n-2 番目の要素を比較し、n-2 番目の要素の方が大きい値であればこれらを交換する。次に n-2 番目と n-3 番目を比較、n-3 番目と n-4 番目を比較、・・・、最後に 0 番目と 1 番目を比較して、先頭の要素のみ整列済みとなる。

続いて繰り返し処理の 2 回目である。n-1 番目から順に 1 番目の要素までを並べ替える。



同様に繰り返し処理を進めていけば、先頭から順に「整列済み」になる。



★次の単純交換ソートのプログラムを実行してみましょう。

```
list = list(map(int,input().split()))
for i in range(len(list)):
    for j in range(len(list) - 1, i, -1):
        if list[j-1] > list[j]:
            list[j-1], list[j] = list[j], list[j-1]
print(list)
```

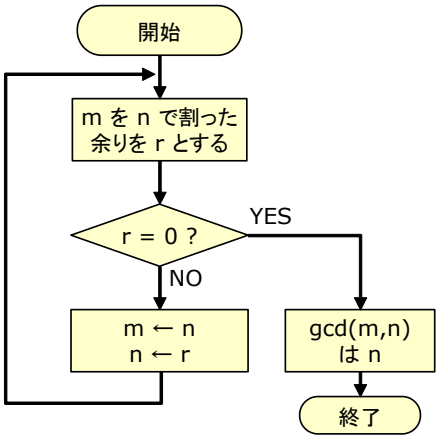
入力：21 6 15 2 29 31 89 65 43 70 54
出力：[2, 6, 15, 21, 29, 31, 43, 54, 65, 70, 89]

入力：21 6 15 2 29
出力：[2, 6, 15, 21, 29]

【ユークリッドの互除法：最大公約数を求めるアルゴリズム】

ソートアルゴリズムについては、次回の「クイックソート」でさらに詳しく学ぶが、今回は前回の授業で学んだ「再帰的アルゴリズム」の例として、ユークリッドの互除法を学習し、そのプログラムを完成させることを課題とする。

ユークリッドの互除法は、最大公約数を再帰的処理によって求めるアルゴリズムである。紀元前 300 年頃に記されたユークリッドの「原論」に書かれてあり、今日も使われている最古のしっかりとしたアルゴリズムとして知られる。フローチャートを以下に示す。



【★課題】
以下のユークリッドの互除法により最大公約数を求めるプログラムについて、????の部分を書き換えて完成させ、正しく実行されることを確認してから、ToyoNet-ACE から提出して下さい。????以外の箇所は変えないこと。

```
def gcd(m, n):
    r = m % n
    if r == ????:
        return ????
    else:
        return gcd(????, ????)

m, n = map(int,input().split())
print(gcd(m,n))
```

入力：42 28
出力：14

入力：7 21
出力：7

【剰余】
剰余とは、割り算をしたときの余りである。整数 a を正の整数 n で割り算をして、商 q 余り r になるということは、
a = nq + r (0 ≦ r < n)
となるような q と r を求めるということである。
Python では、% が剰余を計算する演算子である。上の式において、a % n という演算で剰余 r が計算される。

【計算の流れ】
例えば，m=42，n=28 が与えられたとする。
① 42 を 28 で割った余り 14 が r に入る。
② r はゼロではないので、else 節に入り，gcd(28, 14) となるよう再帰呼び出しをする。
③ 28 を 14 で割った余り 0 が r に入る。
④ r はゼロなので、最大公約数 14 が得られる。
⑤ 以上をまとめると gcd(42,28) = gcd(28,14) = gcd(14,0) のように計算される。

【再帰エラー】

今回は、RecursionError: maximum recursion depth exceeded in comparison という実行時エラーが生じる可能性がある。これは、最大の再帰の深さを超えてしまったということである。

前回学習した階乗のプログラムでは、 factorial(n) のメソッドの中で factorial(n-1) のように、引数を n から n-1 へと減らして呼び出すことで、引数が 5,4,3,2,1 のように減っていって処理が終了したが、もし factorial(n) の中で factorial(n) を呼び出せば、100 万回自分自身を呼び出しても、プログラムが終了しない。コンピュータの資源は有限なので、最大の再帰深さが設定されていて、それを超えたらエラーが出るように設計されているため、このエラーが生じる。

つまり、この実行時エラーが出る場合には、再帰呼び出しの引数がおかしいであろうということが想定できるので、なぜそうなるのかよく考えてみること。

【発展：GUI アプリケーションの作成】

この授業で扱うのは、Paiza によって「入力」をテキストで与えたときに「出力」がテキストで返るようなテキスト入出力のプログラムである。画像を表示したり、マウスで操作をしたりする GUI（グラフィカルユーザーインターフェース）のアプリケーションを作るためにはどうすれば良いであろうか。ここでは、2 つ紹介する。

(1) Tkinter

Tkinter は、Python から GUI を構築・操作するための標準ライブラリである。標準ライブラリなので別途インストールする必要はない。スクリプト言語 Tcl/Tk（ティクル・ティーケー）の Tk を使うインターフェイスということで、Tkinter という名称となっている。

Tkinter を使ったプログラミングの例として、すがや(2020)の Python 入門漫画で解説されているマウスを使ったスカッシュゲームを紹介する。サポートページ[1]からダウンロードできる。『ゲームセンターあらし』は、私が小学生のときによく読んでいた「コロコロコミック」に連載されていた漫画である。すがやみつるは、その後パソコン入門書などの大人向け学習漫画、娯楽小説作家などを経て、2005 年、54 歳から大学、大学院で学び直して 2013 年から京都精華大学マンガ学部教授に就任し、2020 年にはマンガ学部を離れて同大学国際マンガ研究センター教授に就任した。35 年ぶりの描き下ろし単行本が、ゲームセンターあらしによる Python の入門漫画であった。

(2) Kivy

Python の GUI ライブラリには標準ライブラリの Tkinter 以外にも様々な外部モジュールがあり、近年人気が高いライブラリが Kivy である。パソコンだけではなく、スマホ用 OS の Android と iOS、そして Rasbpberry Pi でも動作するという汎用性の高さがメリットである。ただし、レイアウト用の言語 Kv Language を覚える必要があり、難易度は若干高い。Python でスマホアプリを開発したい場合には、Kivy を使うのが良いであろう。ただし、スマホアプリの開発環境を構築する必要がある。詳しくは、Kivy ドキュメントの翻訳[2]を参照。

【文献】

すがやみつる(2020)『ゲームセンターあらしと学ぶ プログラミング入門 まんが版こんにちは Python』日経 BP

1 http://www.m-sugaya.jp/manga_python/

2 https://pyky.github.io/kivy-doc-ja/