たティムソート(Timsort)が採用されている。Python のソースコードリポジトリの中にそのドキュメントがある[1]。Timsort は、Java や Swift にも採用されている。

【発展：Python の実装】

Python はプログラミング言語の名称であり、Python で書かれたプログラムを動かすためのソフトウェア（実装）は一通りではない。その中のいくつかを紹介する。

(1) CPython[2]
Python の標準的な実装（リファレンス実装）であり、C 言語で実装されている。Python と言えばこの実装を意味し、他の実装を区別するときには CPython と表記される。バイトコードインタプリタであり、バイトコードが仮想マシンで実行されるが、バイトコードへの変換は暗黙に行われるために、ユーザーは通常コンパイラを意識しない。

(2) PyPy[3]
JIT（Just-in-Time）コンパイラ（実行時コンパイラ）によって、実行時にコードのまとまりを機械語に逐一コンパイルすることで高速に動作する。PyPy のサイトでは、グイド・ヴァン・ロッサムによる"If you want your code to run faster, you should probably just use PyPy."という言葉が引用されている。

(3) Jython[4]
Java で実装され、Java クラスライブラリを利用できる。Python のプログラムを Java バイトコードにコンパイルできる。

(4) IronPython[5]
C#で実装され、.NET Framework および.NET Framework 互換の Mono 上で動作する。.NET Framework のクラスライブラリを利用できる。

(5) Cython[6]
Python を拡張したプログラミング言語であり、Python のソースファイルを C のコードに変換し、コンパイルすることで Python の拡張モジュールとして出力する。Python のコードの一部に静的な型を宣言することで、コンパイル後のコードの実行速度を高速化できることがある。

---

1 https://github.com/python/cpython/blob/master/Objects/listsort.txt
2 https://www.python.org/
3 https://www.pypy.org/
4 https://www.jython.org/
5 https://ironpython.net/
6 https://cython.org/

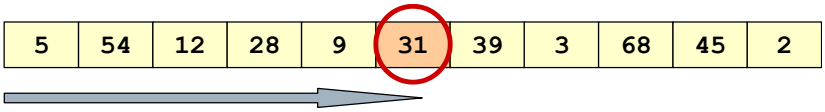
第 11 回　探索アルゴリズム

探索アルゴリズムについて学ぶ。

【線形探索（Linear Search)】

配列の中からある特定の値を持った要素を探すアルゴリズムを、一般に探索アルゴリズムと呼ぶ。探索アルゴリズムには様々なものがあり、もっとも単純な方法が「線形探索」である。線形探索は、配列の先頭から順番に要素を調べていき、探索している数値があるかどうかを調べる。

以下に、線形探索のイメージ図を示す。配列の先頭から順番に調べていき、探索したい値（この例では 31）を探索する。



★線形探索を実現するプログラムを実行しましょう。これまでの知識を活用して「ここではこういう処理をしているのだな」と考えながら打ち込むと良いです。

```
import sys
list = list(map(int,input().split()))
key = int(input())
for i in range(len(list)):
    if list[i] == key:
        print("探索成功：配列の {0} 番目です。".format(i+1))
        sys.exit()
print("探索に失敗しました。")
```

入力：5 54 12 28 9 31 39 3 68 45 2
　　28
出力：探索成功：配列の 4 番目です。

入力データには、1 行目に探索をするデータの列をスペースで区切って入れて、2 行目には探索する数を入れる。

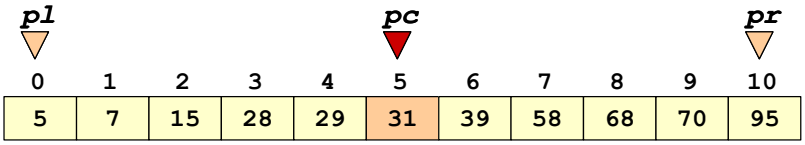
sys.exit() は、プログラムを終了する命令である。sys モジュールの exit 関数を使うので、最初に import sys で sys モジュールを読み込んでいる。このプログラムは、2 つ以上の同じ値がある場合にも、最初の 1 つを見つけた時点で終了する。

【二分探索（Binary Search)】

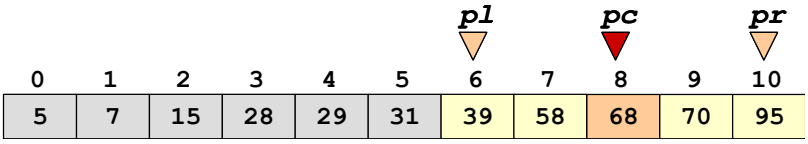
線形探索では，配列の要素の並び順には何も制約がなかった。しかし、これをあらかじめ昇順に整列させておくことで，「二分探索」と呼ばれる、線形探索よりも効率の良い方法で探索できる。たとえば、10 億個のデータから探索する場合には、線形探索では平均で 5 億回、最悪で 10 億回の探索が必要であるが、二分探索であれば、2 の 30 乗が約 10 億なので、10 億個のデータから 30 回以内の探索で目的のデータが見つかる。5 億回と 30 回でどっちが速いかは考えるまでもない。このように、データの数が増えると探索効率の差が歴然とする。

二分探索とは、あらかじめ整列されたデータに対して、探索対象の中央に位置するデータと探索したいデータの大小を比較して探索範囲を半分に絞るという手順を繰り返すことにより、探索したいデータを探すというアルゴリズムである。

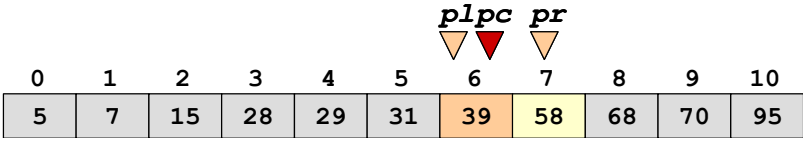
二分探索のイメージ図を以下に示す。二分探索を実行するために、探索範囲の左端を表す変数 pl、探索範囲の右端を表す変数 pr、探索範囲の中央を表す変数 pc を用いる。以下の配列から、39 を探索する場合を考える。まず、pl を左端の 0 番目に、pr を右端の 10 番目に、pc を中央の 5 番目にセットする。



ここで、pc が指し示す 31 と、探索対象の 39 の大小を比較する。探索対象の 39 は 31 より大きいので、31 の右隣から右端までの範囲を新たな探索対象とする。そのために、pl を pc の 1 つ右隣へ移動させ、pc を新 pl と pr の中間にセットする。



ここで、pc が指し示す 68 と、探索対象の 39 の大小を比較する。探索対象の 39 は 68 より小さいので、pl から，68 の 1 つ左隣までを新たな探索対象にする。pc は、pl と pr を足して 2 で割った値を切り捨てて、pl と同じ 6 番目とする。



ここで、pc が指し示す 39 は探索対象に等しいので、探索を終了する。

さて、探索に失敗した場合、何を終了の条件とすればよいだろうか。例えば、上記の例で 40 を探す場合を考えてみよう。pl と pr の位置関係がどうなったときに探索を終了させればよいでしょうか？

【★課題】

二分探索を実現するプログラムの????部分に入る適切なプログラムコードを記述して完成させ、実行して下さい。入力は、小さい数から大きい数へと順番に並んでいる必要があることに注意すること。完成したら、ToyoNet-ACE に提出して下さい。

```
import sys
list = list(map(int,input().split()))
key = int(input())
pl = 0
pr = len(list) - 1
while pl <= pr:
    pc = (pl + pr) // 2
    if ????:
        print("探索成功：配列の {0} 番目です。".format(pc+1))
        sys.exit()
    elif ????:
        pl = ????
```