ソート・二分探索のやり方

ひらく 🔗

ソートとは（1/2）

データを何らかの規則に沿って並べ替えること

例）数字の配列をソート

```
a = [3, 4, 2, 5, 1]

a.sort() # 昇順でソート
print(a) # [1, 2, 3, 4, 5]

a.sort(reverse=True) # 降順でソート
print(a) # [5, 4, 3, 2, 1]
```

ソートとは（2/2）

例）文字列の配列をソート

```
b = ['Bob', 'Alice', 'Charlie']
c = ['Bob', 'Alice', 'Charlie']

b.sort() # 辞書順でソート
print(b) # ['Alice', 'Bob', 'Charlie']

# sorted関数を使うと元の配列を変えずにソートできる
print(sorted(c)) # ['Alice', 'Bob', 'Charlie']
print(c)         # ['Bob', 'Alice', 'Charlie']
```

ソートアルゴリズムの種類

ソートアルゴリズムはたくさんある

15 Sorting Algorithms in 6 Minutes-YouTube
https://www.youtube.com/watch?v=kPRA0W1kECg

Wikipedia には**35 種類**載ってました

ソートの計算量

色々なアルゴリズムがあるので計算量は様々（$O(\infty)$のものもある）

早いもので計算量は$O(n \log n)$
　→ $n \leqslant 10^6$ 個くらいのデータはソートが可能

python ではティムソートというアルゴリズムが採用されている

ちなみに C++ではイントロソートというアルゴリズムが採用されている

演習問題

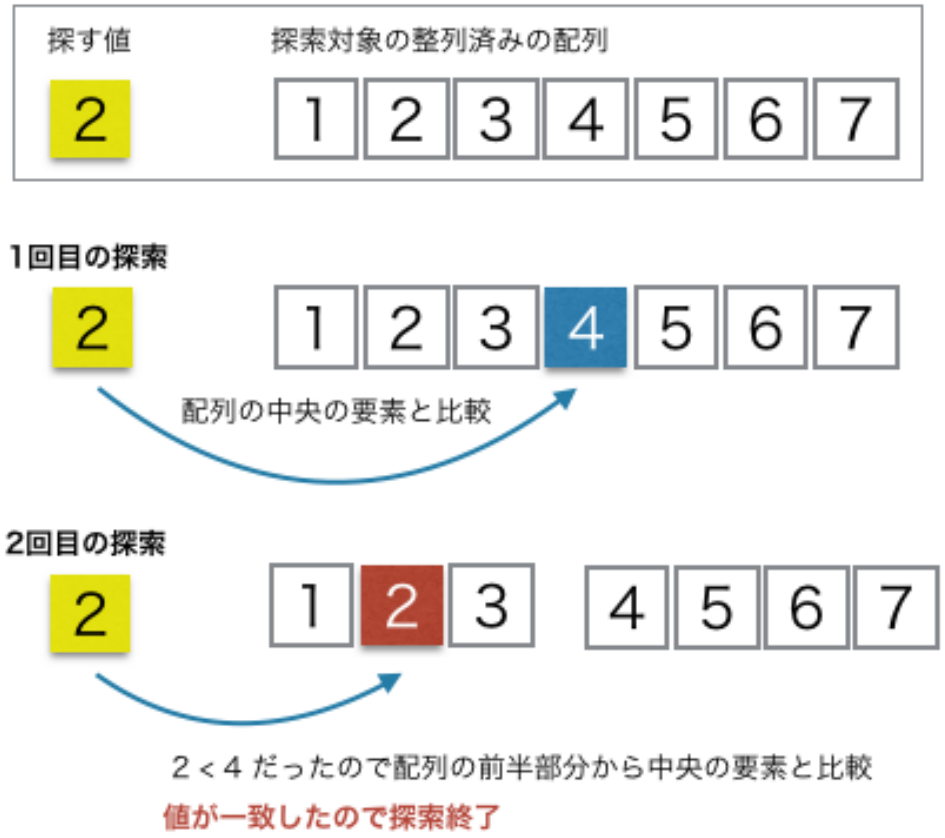
- Q1. 中央値（アルゴ式）
  https://algo-method.com/tasks/501
- Q3. 総和の最大値（アルゴ式）
  https://algo-method.com/tasks/484

二分探索とは（1/2）

ソート済みである配列やリストに対して探索を行う手法

探索対象を1/2ずつ削り落としていくので効率よく探索することができる

値の比較によって絞り込むので, ソート済みであることが絶対必要



https://www.codereading.com/algo_and_ds/algo/binary_search

二分探索の実装（1/2）

例）配列 $[2, 4, 6, 8, 10, 12, 14, 16]$ から 8 を探す

```
a = [2, 4, 6, 8, 10, 12, 14, 16]
a.sort() # ソート済みであることが必要
x = 8    # 探したい値
left = 0, right = len(a)-1  # 探索範囲の左端と右端
```

二分探索の実装（2/2）

例）配列 $[2, 4, 6, 8, 10, 12, 14, 16]$ から 8 を探す

```
while(right >= left):    # 探索範囲がなくなるまで繰り返す
  mid = (left+right)//2 # 探索範囲の中間を取得
  if(a[mid] == x):
    print(mid)     # 値が見つかったら終了
    break
  elif a[mid] < x:
    left = mid+1   # 中間より左の区間を削る
  else:
    right = mid-1 # 中間より右の区間を削る
```

モジュールを使った実装

python には `bisect` という二分探索のためのモジュールがある

```
import bisect
a = [2, 4, 6, 8, 10, 12, 14, 16]
a.sort() # ソート済みであることが必要
x = 8    # 探したい値

pos = bisect.bisect_left(a, x)  # x以上の要素のインデックス
print(pos) # 3
pos = bisect.bisect_right(a, x) # xより大きい要素のインデックス
print(pos) # 4
```

めぐる式二分探索

条件を満たす値の最大(小)値を求める問題でよく使うアルゴリズム

```
# [ok, ng)の区間に解が存在する
ok = 0
ng = 1000
while(ng-ok > 1):    # 区間に含まれる数が1つになるまで繰り返す
  mid = (ok+ng)//2   # 区間の中央値を取得
  if solve(mid):
    ok = mid  # 区間を左半分にする
  else:
    ng = mid  # 区間を右半分にする
print(ok)
```

二分探索の計算量

探索を行うごとに範囲が 1/2 され, 探索範囲が 0 になるまで繰り返す

→ $\log n$回行えば探索範囲がなくなるので, $O(\log n)$

※ただし, ソートも含めて考えるのであれば$O(n \log n)$

演習問題

- Q2-3. 最小の添字（アルゴ式）
  https://algo-method.com/tasks/370
- A12 - Printer （AtCoder）ちょっと応用的
  https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_l

  ヒント：時間を探索対象として, ある時間で印刷できる枚数を求め、それを元に範囲を絞っていく