

2023年11月10日(金)

アルゴリズム数理B 第7週 課題

理工学部 数学科 3年

1070 富山和甫

問題1.

quick_sort1.py, quick_sort2.py を実行したキャプチャを図1, 図2に示す.

```
リスト1 quick_sort1.py
>python quick_sort1.py
ソートしたい値をスペース区切りで入力 : 1 4 1 5 9 2 6 5 3 5
1 import random
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
def quick_sort_recursive(a, first, last):
    """クイックソートの再帰部分"""
    Args:
        a (list[str or int]): ソート対象のリスト
        first (int): 先頭インデックス
        last (int): 末尾インデックス
    Returns:
        list[str or int]: ソートしたリスト
    """
    # pivot を選択
    # random.choice() はランダムに要素を1つ返す
    pivot = random.choice(a[first:last])
```

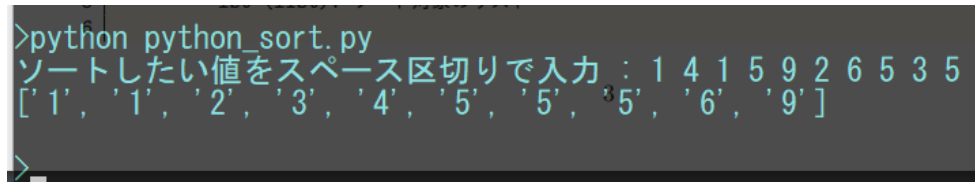
図1 quick_sort1.py の実行結果

```
リスト2 quick_sort2.py
>python quick_sort2.py
ソートしたい値をスペース区切りで入力 : 1 4 1 5 9 2 6 5 3 5
1 import random
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
def quick_sort_recursive(a, first, last):
    """クイックソートの再帰部分"""
    Args:
        a (list[str or int]): ソート対象のリスト
        first (int): 先頭インデックス
        last (int): 末尾インデックス
    Returns:
        list[str or int]: ソートしたリスト
    """
    # pivot を選択
    # random.choice() はランダムに要素を1つ返す
    pivot = random.choice(a[first:last])
```

図2 quick_sort2.py の実行結果

問題 2

python_sort.py を実行した結果を図 3 に示す.



```
>python python_sort.py
ソートしたい値をスペース区切りで入力 : 1 4 1 5 9 2 6 5 3 5
['1', '1', '2', '3', '4', '5', '5', '5', '6', '9']
>
```

図 3 python_sort.py の実行結果

問題 3

以下の各ソートアルゴリズムについてベンチマークを用いて実行時間を測定した. 初期のリストの状態はソート済み, 降順, ランダムの 3 種類で実行した.

- ・バブルソート
- ・選択ソート
- ・ヒープソート
- ・マージソート
- ・クイックソート (再帰版)
- ・クイックソート (非再帰版)
- ・Python のリスト型メソッドによるソート

また, ソートする配列のサイズはどれも 10000 とした.

ソート済みの配列に対するソート時間のベンチマークの結果を図 4 に, 降順配列の結果を図 5 に, ランダムな配列の結果を図 6 に示す.

```

>python sort_benchmark_sort.py
ソートする値のサイズ = 10000
## benchmarker:      release 4.0.1 (for python)
## python version:    3.12.0
## python compiler:   MSC v.1935 64 bit (AMD64)
## python platform:   Windows-10-10.0.19045-SP0
## python executable: C:\Users\cotan\AppData\Local\Programs\Python\Python312\python.exe
## cpu model:         Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
## parameters:        loop=10000, cycle=1, extra=0

##
##          real      (total    = user    + sys)
bubble_sort      4.7392      4.4688      4.4531      0.0156
selection_sort    3.4616      3.4531      3.4375      0.0156
insertion_sort    0.0023      0.0000      0.0000      0.0000
heap_sort         0.0054      0.0000      0.0000      0.0000
merge_sort        0.0276      0.0312      0.0312      0.0000
quick_sort1(recursive) 0.0157      0.0156      0.0156      0.0000
quick_sort2(not recursive) 0.0161      0.0156      0.0156      0.0000

## Ranking
##          real
insertion_sort    0.0023      (100.0) *****
heap_sort         0.0054      ( 41.9) *****
quick_sort1(recursive) 0.0157      ( 14.4) ***
quick_sort2(not recursive) 0.0161      ( 14.1) ***
merge_sort        0.0276      (  8.2) **
selection_sort    3.4616      (  0.1)
bubble_sort       4.7392      (  0.0)

## Matrix
##          real      [01]      [02]      [03]      [04]      [05]      [06]      [07]
[01] insertion_sort    0.0023      100.0      238.9      694.7      710.9      1216.4      152687.0      209038.2
[02] heap_sort         0.0054      41.9      100.0      290.8      297.6      509.1      63906.9      87492.6
[03] quick_sort1(recursive) 0.0157      14.4      34.4      100.0      102.3      175.1      21978.5      30090.0
[04] quick_sort2(not recursive) 0.0161      14.1      33.6      97.7      100.0      171.1      21477.2      29403.6
[05] merge_sort        0.0276      8.2      19.6      57.1      58.4      100.0      12552.1      17184.6
[06] selection_sort    3.4616      0.1      0.2      0.5      0.5      0.8      100.0      136.9
[07] bubble_sort       4.7392      0.0      0.1      0.3      0.3      0.6      73.0      100.0

```

図4 昇順配列の各ソートアルゴリズムのソート時間

```

>python sort_benchmark_reverse.py
ソートする値のサイズ = 10000
## benchmarker:      release 4.0.1 (for python)
## python version:    3.12.0
## python compiler:   MSC v.1935 64 bit (AMD64)
## python platform:   Windows-10-10.0.19045-SP0
## python executable: C:\Users\cotan\AppData\Local\Programs\Python\Python312\python.exe
## cpu model:         Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
## parameters:        loop=10000, cycle=1, extra=0

##
##          real      (total    = user    + sys)
bubble_sort      10.0657      9.7344      9.7344      0.0000
selection_sort    3.8540      3.7188      3.7188      0.0000
insertion_sort    6.1259      6.0625      6.0625      0.0000
heap_sort         0.0036      0.0156      0.0156      0.0000
merge_sort        0.0232      0.0156      0.0156      0.0000
quick_sort1(recursive) 0.0147      0.0156      0.0156      0.0000
quick_sort2(not recursive) 0.0107      0.0156      0.0156      0.0000

## Ranking
##          real
heap_sort         0.0036      (100.0) *****
quick_sort2(not recursive) 0.0107      ( 33.4) *****
quick_sort1(recursive) 0.0147      ( 24.3) *****
merge_sort        0.0232      ( 15.4) ***
selection_sort     3.8540      (  0.1)
insertion_sort     6.1259      (  0.1)
bubble_sort       10.0657      (  0.0)

## Matrix
##          real      [01]      [02]      [03]      [04]      [05]      [06]      [07]
[01] heap_sort         0.0036      100.0      299.6      412.1      650.9      108204.5      171991.7      282607.8
[02] quick_sort2(not recursive) 0.0107      33.4      100.0      137.5      217.2      36113.3      57402.3      94320.5
[03] quick_sort1(recursive) 0.0147      24.3      72.7      100.0      158.0      26258.4      41737.9      68581.5
[04] merge_sort        0.0232      15.4      46.0      63.3      100.0      16624.2      26424.2      43418.9
[05] selection_sort     3.8540      0.1      0.8      0.4      0.6      100.0      159.0      261.2
[06] insertion_sort     6.1259      0.1      0.2      0.2      0.4      62.9      100.0      164.3
[07] bubble_sort       10.0657      0.0      0.1      0.1      0.2      33.3      60.9      100.0

```

図5 降順配列の各ソートアルゴリズムのソート時間

```

>python sort_benchmark_shuffle.py
ソートする値のサイズ = 10000
## benchmarker:      release 4.0.1 (for python)
## python version:    3.12.0
## python compiler:   MSC v.1935 64 bit (AMD64)
## python platform:   Windows-10-10.0.19045-SP0
## python executable: C:\Users\cotan\AppData\Local\Programs\Python\Python312\python.exe
## cpu model:         Intel64 Family 6 Model 140 Stepping 1, GenuineIntel
## parameters:        loop=10000, cycle=1, extra=0

##
##          real      (total    = user    + sys)
bubble_sort      7.5855      7.3125      7.3125      0.0000
selection_sort   3.2357      3.0625      3.0625      0.0000
insertion_sort   3.2793      3.1562      3.1562      0.0000
heap_sort        0.0050      0.0000      0.0000      0.0000
merge_sort       0.0248      0.0312      0.0312      0.0000
quick_sort1(recursive)  0.0229      0.0156      0.0156      0.0000
quick_sort2(not recursive) 0.0243      0.0156      0.0156      0.0000
python_sort      0.0023      0.0000      0.0000      0.0000

## Ranking
##          real
python_sort      0.0023 (100.0) *****
heap_sort        0.0050 ( 45.3) *****
quick_sort1(recursive)  0.0229 (  9.9) **
quick_sort2(not recursive) 0.0243 (  9.3) **
merge_sort       0.0248 (  9.1) **
selection_sort   3.2357 (  0.1)
insertion_sort   3.2793 (  0.1)
bubble_sort      7.5855 (  0.0)

## Matrix
##          real      [01]      [02]      [03]      [04]      [05]      [06]      [07]      [08]
[01] python_sort      0.0023      100.0      220.6      1012.2      1071.5      1096.6      142812.4      144737.9      334799.9
[02] heap_sort        0.0050      45.3      100.0      458.8      485.6      497.0      64727.7      65600.4      151743.4
[03] quick_sort1(recursive)  0.0229      9.9      21.8      100.0      105.9      108.3      14109.2      14299.4      33076.6
[04] quick_sort2(not recursive) 0.0243      9.3      20.6      94.5      100.0      102.3      13328.4      13508.1      31246.1
[05] merge_sort       0.0248      9.1      20.1      92.3      97.7      100.0      13022.7      13198.3      30529.5
[06] selection_sort   3.2357      0.1      0.2      0.7      0.8      0.8      100.0      101.3      234.4
[07] insertion_sort   3.2793      0.1      0.2      0.7      0.7      0.8      98.7      100.0      231.3
[08] bubble_sort      7.5855      0.0      0.1      0.3      0.3      0.3      42.7      43.2      100.0

```

図6 ランダムな配列の各ソートアルゴリズムのソート時間

問題4

- (1) データをシャッフルした場合
ヒープソート
- (2) データを昇順にソートした場合
挿入ソート
- (3) データを降順にソートした場合
挿入ソート

実質、バブルソート同じような動作をするため、計算量が $O(n^2)$ になるから。