データ構造とアルゴリズム

演習課題 2

提出日 2024/05/25

HI4 45 号 山口惺司

レポート課題

1. アルゴリズムの説明：

ダブルセレクションソートには、単純に最大値選択法と最小値選 択法を組み合わせた方法以外に、次のようなアルゴリズムもある。

① {2　8　3　4　7　6　1　5} が与えられたとする（ただし、偶数個の 場合のみ）。

② {2　8　3　4} {7　6　1　5} の 2 つに分割する。

③ {2　6　1　4} {7　8　3　5}　…2 つのグループで先頭から 1 個ずつ取り出して大小を比較して、左側が大きい時は入れ替える（水色はグループ をまたいだ入れ替え）。

④ 1 {6　2　4} {7　5　3} 8 … 左のグループの最小値、右のグルー プの最大値を求め、それぞれ先頭、末尾と入れ替えて、グループから外す（緑 色はグループ内の入れ替え）。

⑤ 1 {6　2　3} {7　5　4} 8

⇒ 1　2 {6　3} {4　5} 7　8 …2 つのグループについて③、④を行う。

⑥ 1　2　{4　3} {6　5} 7　8

⇒ 1　2　3　{4} {5} 6　7　8

⇒ 1　2　3　4　5　6　7　8 … 各グループとも 1 個になったら終了。

2. 課題

(1) この方法について、{8 4 3 9 6 5}がソートされる経過を図に示せ。また、比較と交換の回数も書け。

与えられた配列がソートされる経過を図 1 に示す。

配列						比較回数	交換回数	操作
{8	4	3	9	6	5}			①
{8	4	3}	{9	6	5}			②
{8	4	3}	{9	6	5}	3	0	③
{8	4	3}	{9	6	5}	4	2	④
3	{4	8}	{5	6}	9	2	1	③
3	{4	6}	{5	8}	9	2	0	④
3	4	{6}	{5}	8	9	1	1	③
3	4	5	6	8	9			

図 1 {8 4 3 9 6 5}がソートされる経過

図 1 より今回与えられた配列をソートするときの比較回数は 12 回、交換回数は 4 回となった。

(2) この方法について、フローチャートと疑似コードを書き、Python で実装して結果を確認せよ。

フローチャート：

上記のアルゴリズムからフローチャートを作成し、図 2 に示す。

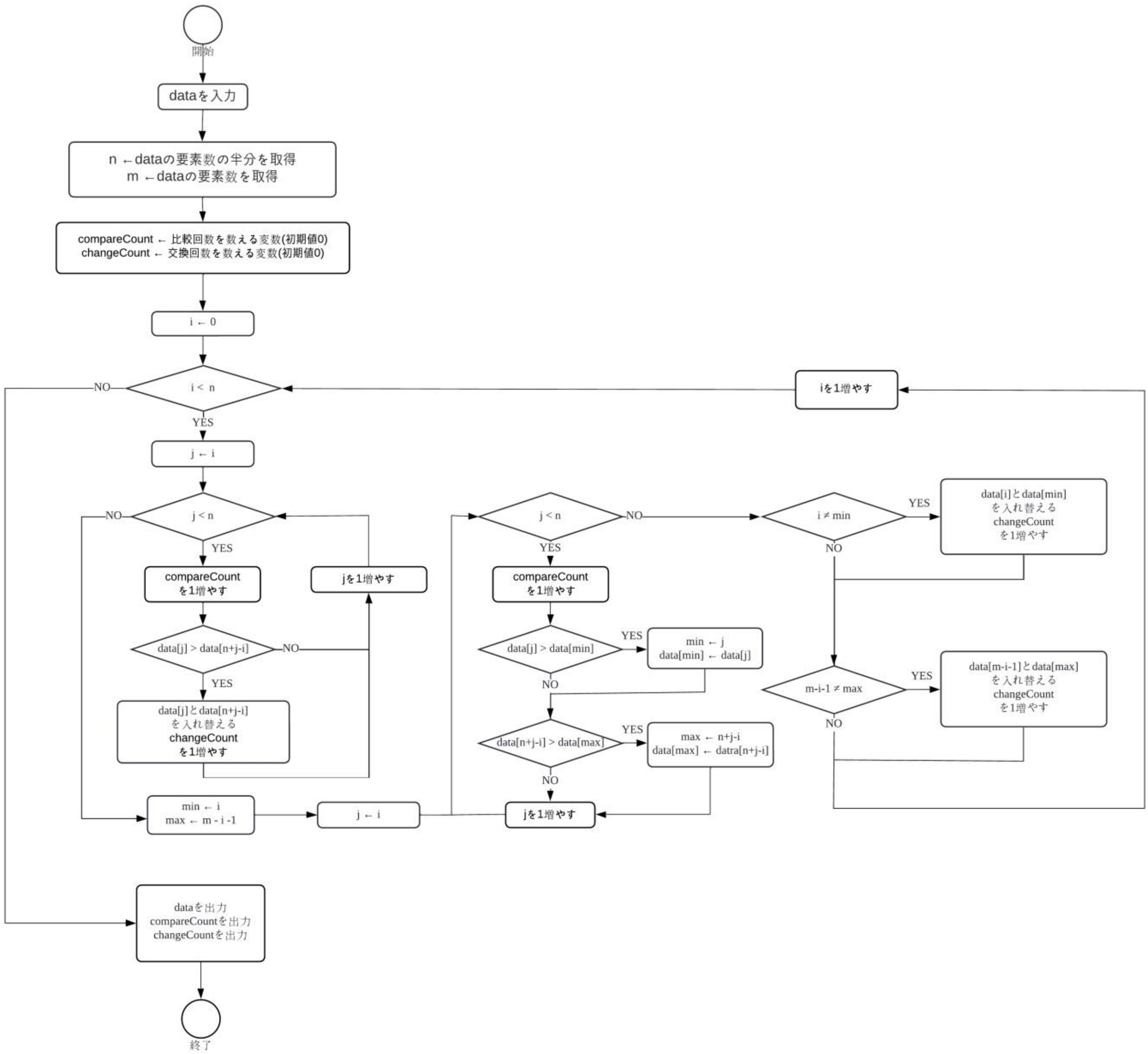

図 2 ダブルセレクションソートフローチャート

疑似コード：

フローチャートから疑似コードを以下に示す。

① ソートするデータ data を取得する。

② 配列の半分の長さ n, 配列の長さ m を取得する

③ 比較回数と交換回数を数える compareCount, changeCount = 0 を取得する。

④ i = 0 を取得する。

⑤ i < n でなければ⑳に進む

⑥ j = i を取得する。

⑦ j<n でなければ⑪に進む

⑧ compareCount を 1 増やす

⑨ data[j] > data[n+j+i]であれば 2 つを入れ替え、changeCount を 1 増やす。そうでなければそのまま進む。

⑩ j を 1 増やし⑦に戻る

⑪ min=i, max=n, j=i を取得する。

⑫ j<n でなければ⑰に進む

⑬ compareCount を 1 増やす

⑭ data[j]>data[min]であれば min=j, data[min]=data[j]を取得する。そうでなければそのまま進む。

⑮ data[n+j-i]>data[max]であれば max=n+j-i, data[max]=data[j]を取得する。そうでなければそのまま進む。

⑯ j を 1 増やし⑫に戻る

⑰ i≠min であれば、data[i]と data[min]を入れ替え、changeCount を 1 増やす。そうでなければそのまま進む

⑱ m-i-1≠max であれば、data[m-i-1]と data[max]を入れ替え、changeCount を 1 増やす。そうでなければそのまま進む

⑲ i を 1 増やし、⑤に戻る

⑳ data, compareCount, changeCount を出力して終了する

Python プログラム：

疑似コードを基に Python プログラムを作成し、以下に示す。

```
data = [8, 4, 3, 9, 6, 5]
n = int(len(data)/2)
m = int(len(data))
print("ソート前:", data)
changeCount = 0
compareCount = 0
for i in range(n):
    for j in range(i, n):
        compareCount += 1
        if data[j] > data[n + j - i]:
            data[j], data[n + j - i] = data[n + j -i], data[j]
            changeCount += 1

    min = i
    max = m - i - 1
    for j in range(i, n):
        compareCount += 1
        if data[j] < data[min]:
            min = j
            data[min] = data[j]
```

```python
        if data[n + j - i] > data[max]:
            max = n + j -i
            data[max] = data[n + j - i]

    if i != min:
        data[i], data[min] = data[min], data[i]
        changeCount += 1
    if (m-i-1) != max:
        data[m-i-1], data[max] = data[max], data[m-i-1]
        changeCount += 1

print("ソート後:", data)
print("比較回数:", compareCount)
print("交換回数:", changeCount)
```

実行結果：

上のプログラムを実行した時の実行結果を図 3 に示す。

```
ソート前： [8, 4, 3, 9, 6, 5]
ソート後： [3, 4, 5, 6, 8, 9]
比較回数： 12
交換回数： 4
```

図 3 {8 4 3 9 6 5}が与えられた時の実行結果

また、与える配列を変えて実行した時の実行結果を図 4〜8 に示す。

```
ソート前： [2, 10, 1, 4, 5, 5, 8, 6]
ソート後： [1, 2, 4, 5, 5, 6, 8, 10]
比較回数： 20
交換回数： 5
```

図 4 {2 10 1 4 5 5 8 6}が与えられた時の実行結果

```
ソート前： [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
ソート後： [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
比較回数： 30
交換回数： 0
```

図 5 {1 2 3 4 5 6 7 8 9 10}が与えられた時の実行結果

```
ソート前： [6, -1, -4, 3, 11, 2, 0, 28]
ソート後： [-4, -1, 0, 2, 3, 6, 11, 28]
比較回数： 20
交換回数： 5
```

図 6 {6 -1 -4 3 11 2 0 28}が与えられた時の実行結果

```
ソート前： [5, 2, 3, 1, 4]
ソート後： [1, 3, 2, 4, 5]
比較回数： 6
交換回数： 5
```

図 7 {5 2 3 1 4}が与えられた時の実行結果

```
ソート前： [2, 2, 2, 2, 2, 2]
ソート後： [2, 2, 2, 2, 2, 2]
比較回数： 12
交換回数： 0
```

図 8 {2 2 2 2 2 2}が与えられた時の実行結果

3. 考察

図 7 は配列の要素数を奇数にした場合だが、要素数が奇数個だとうまく動作しないことが分かった。
図 3～図 6 は全て正しくソートできているため、プログラムは正しく実装できていると言える。
図 3 と図 8 から、ダブルセレクションソートは、同じ要素数の配列でも与えられた配列によって交換回数が違うということがわかった。
また、比較回数は表 1 のようになる

表 1 要素数と比較回数の関係

要素数	0	2	4	6	8	10
比較回数	0	2	6	12	20	30

つまり要素数 n における比較回数は以下の式で表せる。

$$比較回数 = \frac{n(n+2)}{4}$$