2006 年 2 月 24 日

紙の束の配分による整列化について

新潟工科大学 情報電子工学科 竹野茂治

1 はじめに

私は定期試験の試験用紙は、採点のときにその順番を変えてしまうことも多いので、回収時は特に番号順には集めてはいない。

既にコンピュータ上に履習学生の一覧リストがある場合は、成績の入力時に採点済みの答案用紙を整列化 (ソート) してから入力する場合もあるが、答案用紙の整列化は人間には面倒でも、データの整列化自体はコンピュータでは簡単にできるので、答案を整列化せずにそのまま学籍番号と点数を入力して、入力したデータをコンピュータの方で整列化する場合もある。つまり、少なくとも私にとっては、この段階までは必ずしも答案用紙を整列化する必要はない。

しかし、そのテストの答案用紙を学生に返却する場合は、番号順に読み上げた方が学生もわかりやすいので、答案用紙の整列化が必要になる。

その場合、例えば 10 番ずつ、20 番ずつの山に配り分けて、それぞれの山をさらに小さく分けながら整列化したりしていたが、上に述べたような状況の場合、答案用紙の整列化されていない並びが既にコンピュータに入力されているので、コンピュータに整列化するための手順を求めさせれば、コンピュータの補助の下で早く答案用紙の整列化ができるのではないか、と漠然と思っていた。

今回、研究室の学生の卒業研究 ([1]) とも関連して、それをちゃんと考えてみたので、それをここにまとめておくことにする。

2 問題設定

ここでは、問題の設定を明確にする。

それぞれに番号が書かれている N 枚の紙を、いくつかの山に適当に配り分け、そしてそれを積み上げて再び配り分ける、といったことを何回か繰り返すことで整列化する、ということを考える。配ることができる場所の数を M $(M \geq 2)$ とし、それらにそれぞれ $A_1, A_2, \ldots, A_M$ と名前をつけることにする。

- ルール **1**: N 枚の紙には同じ番号が書かれたものはないとし、よって、簡単のため、1 から N までの数が書かれているものとする。

- ルール **2**: 手持ちの紙は、上から 1 枚ずつ、M 個の場所のいずれかの場所の一番上に置いていく。
- ルール **3**: 山に配ったものを手の方に戻したり、配っている途中で山の紙を別の山へ移動したりはしないものとする。
- ルール **4**: N 枚を配り終えたら、A_j の山を順番に重ね、それを新たな手持ちの山とし、必要ならば何回かこの操作を繰り返す。
- ルール **5**: N 枚を配り終えるまで作業は中断しないものとし、ルール 4 の最後の手持ちの山が整列化されることを目標とする。
- ルール **6**: M 個の場所には、紙が 1 枚も置かれない場所があってもよい。

配り方、集め方としては、次の 2 種類を考えることとする。

- 方法 **A**: 山に置くときには紙を裏返して置き、最後に集めるときは、A_1 の上に A_2 を、A_2 の上に A_3 を、という順に重ねていき、最後にそれ全体を引っくり返して手持ちとする。すなわち、A_1 の一番下 $(一番初めに配ったもの)$ が手持ちの一番上に、A_M の一番上 $(一番最後に配ったもの)$ が手持ちの一番下にくることになる。
- 方法 **B**: 山に置くときには紙を表のまま置き、最後に集めるときは、A_1 の下に A_2 を、A_2 の下に A_3 を、という順に重ねていき、それを手持ちとする。すなわち、A_1 の一番上 $(一番最後に配ったもの)$ が手持ちの一番上に、A_M の一番下 $(一番最初に配ったもの)$ が手持ちの一番下にくることになる。

方法 A の方が考察には単純であるが、実際に配るときは番号等を確認できる方法 B の方が多少便利である。

ルール 5 により、手持ちの紙を N 回分配する作業が単位となるので、作業回数は、それを 1 回、と数えることにする。よって、1 回の作業は紙の N 回の分配の手順からなることになる。

なお、[1] では、$M = 2$ で、かつ上のルール 3、ルール 4 に対し、

- ルール **3'**: 配っている途中で、右の山の一番上を左の山に移動したり、左の山の一番上を右の山に移動したりしてもよい
- ルール **4'**: 配り終ったものを再び手に戻すことはしない $(1$ 回で終り$)$

というルールの元での考察を行っているが、実際にはルール 3' のような答案用紙の移動はやりづらいし、そこに時間のロスが起きる。しかも、上記のルールの場合は、機械化も可能だと思われるが、ルール 3' の場合は機械化も複雑になると予想される。

実行に必要な手順数についても、[1] の設定よりもこのルールの方がいい結果が得られることもわかった。

3 簡単な解

ここでは、まず簡単に思いつく解を一つ紹介する。例として、$N = 16$, $M = 2$ で、方法は A の場合で説明する。

1. 1 回目に、A_1 に 8 以下、A_2 に 9 以上の番号を配り分け、それを集める。すると、手持ちの紙は上半分が 1 から 8, 下半分が 9 から 16 になる (図 1)。

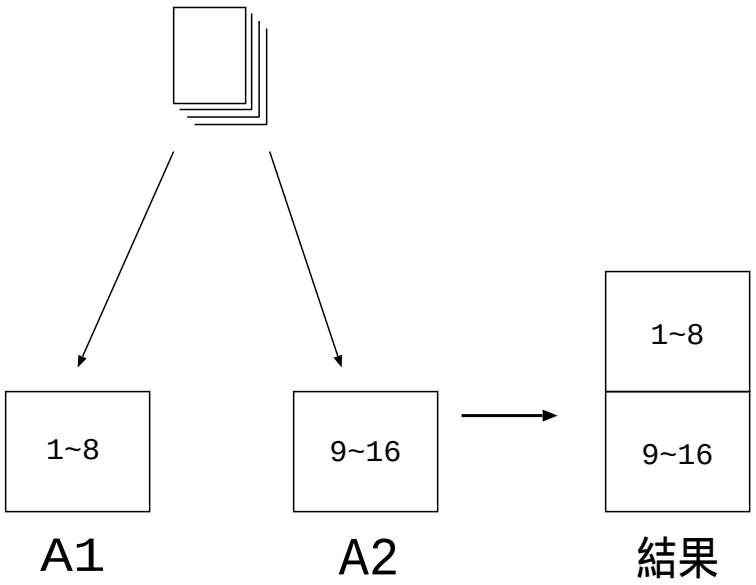

図 1: 単純な解: 1 回目

2. 2 回目の分配では、

   - 最初の 8 枚は、A_1 に $1 \sim 4$ を、A_2 に $5 \sim 8$ を、
   - 残りの 8 枚は、A_1 に $9 \sim 12$ を、A_2 に $13 \sim 16$ を

   それぞれ配分する。その結果は、上から 4 枚ずつ $1 \sim 4$, $9 \sim 12$, $5 \sim 8$, $13 \sim 16$ と分けられることになる (図 2)。

3. 3 回目、4 回目も同様に、それぞれの小ブロックを、A_1 には小さい数、A_2 には大きい数となるように 2 つに分けて分配していく (図 3)。その結果、4 回目にすべて 1 枚ずつのブロックに分割され、上から順に、

   $$1, 9, 5, 13, 3, 11, 7, 15, 2, 10, 6, 14, 4, 12, 8, 16$$

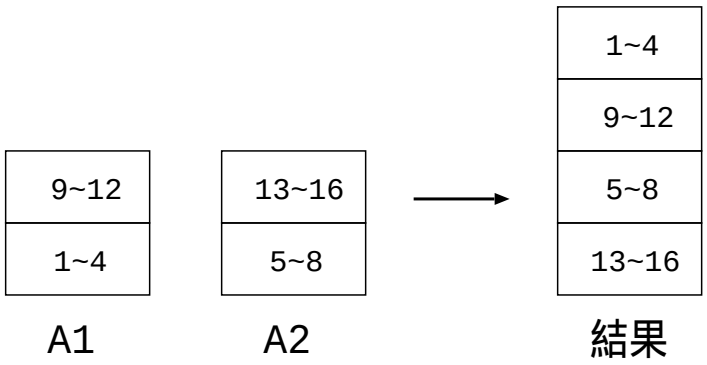

図 2: 単純な解: 2 回目

と並ぶことになる。

この最後の並びは、最初の並びが何であっても上の手順で確実にその並びにできる。

よって、この最後の並びが 1,2,3,. . . , 16 となるように

$$\begin{array}{cccccccccccccccc} 1 & 9 & 5 & 13 & 3 & 11 & 7 & 15 & 2 & 10 & 6 & 14 & 4 & 12 & 8 & 16 \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) & (9) & (10) & (11) & (12) & (13) & (14) & (15) & (16) \end{array}$$

という対応をつけて逆に考え、このかっこの方の数字を紙の数字と見ることで昇順に整列化できることになる。

例えば、上の 1 回目の配分は、A_1 が $1 \sim 8$, A_2 が $9 \sim 16$, すなわち、かっこの数字で言えば

$$\begin{cases} A_1 : & \{(1), (9), (5), (13), (3), (11), (7), (15)\} \\ A_2 : & \{(2), (10), (6), (14), (4), (12), (8), (16)\} \end{cases}$$

と分けることに相当する ({} 内は順不同)。これまでの作業をこのように読み変えていけばよい。

ついでに 2,3,4 回目の配分も書き上げると、

- 2 回目

$$\begin{cases} A_1 : & \{(1), (9), (5), (13)\} & \{(2), (10), (6), (14)\} \\ A_2 : & \{(3), (11), (7), (15)\} & \{(4), (12), (8), (16)\} \end{cases}$$

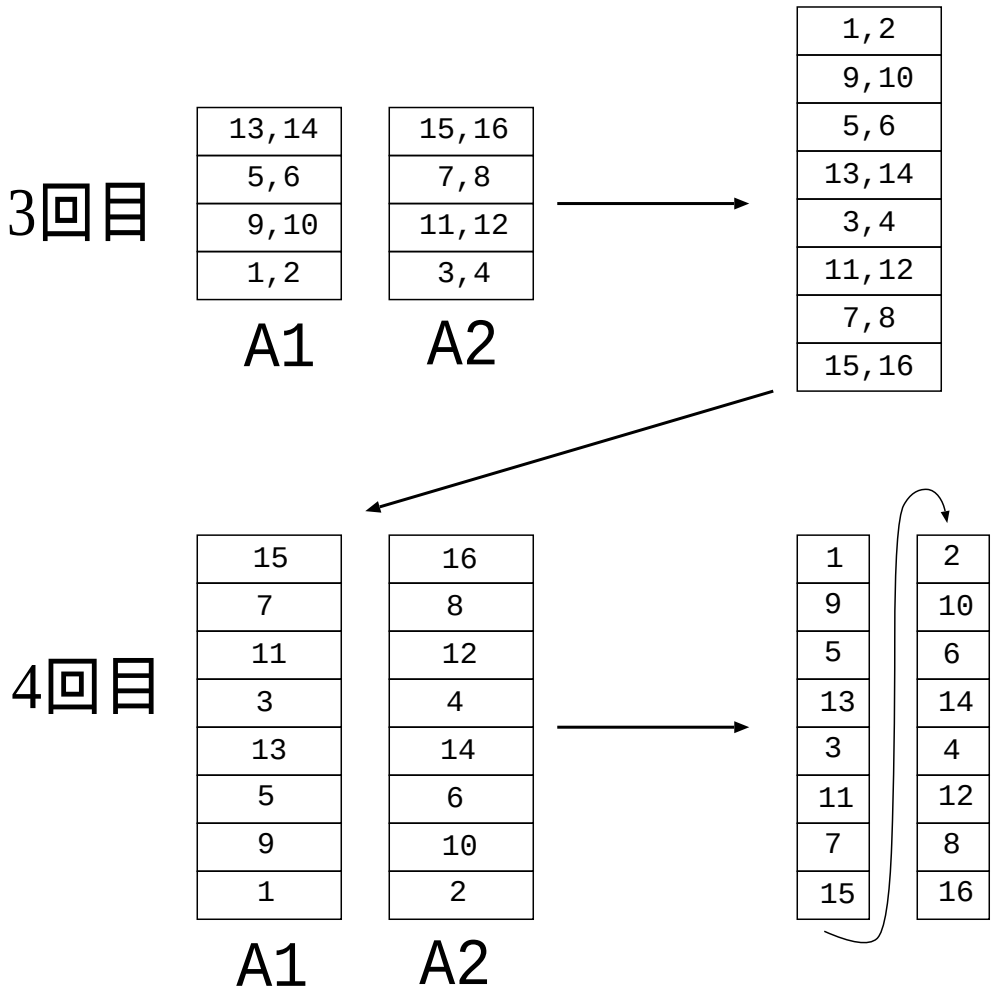

図 3: 単純な解: 3, 4 回目

- 3 回目

$$\begin{cases} A_1: & \{(1),(9)\} & \{(2),(10)\} & \{(3),(11)\} & \{(4),(12)\} \\ A_2: & \{(5),(13)\} & \{(6),(14)\} & \{(7),(15)\} & \{(8),(16)\} \end{cases}$$

- 4 回目

$$\begin{cases} A_1: & (1) & (2) & (3) & (4) & (5) & (6) & (7) & (8) \\ A_2: & (9) & (10) & (11) & (12) & (13) & (14) & (15) & (16) \end{cases}$$

と、このように整列化が進むことになる。

しかし、これらの分配は人間がすぐに判定して左右に分けることは難しいので、コンピュータの音声ガイドなどが必要であると思われる。

この方法だと、$9 \leq N \leq 16$ の場合は今と同じやり方で 4 回で整列化が行なえる。一般の N, M の場合は、

$$M^{K-1} < N \leq M^K \text{ ならば } K \text{ 回で整列化可能} \tag{1}$$

となることが言え、よって必要な回数は $\lceil \log_M N \rceil$ となる ($\lceil x \rceil$ は x 以上の最小の整数)。

しかし、これはどのような初期配列に対しても同じ回数かかる手順であり、最適な解ではない。

4 A 列と B 列

次に、初期配列に依存した最適解について考察する。ここでもしばらくは $M = 2$ の場合の方法 A について考えることにする。

3 節同様に最初の列が 1,2,...,N の並びであるとして (前節のかっこのついた数字)、この配分でどのような列が生成するかを考えることにする。1,2,...,N の初期配列から全ての順列が生成できれば、逆に考えれば、全ての順列を整列化できることになる。

今後、この初期配列を 1,2,...,N のように見る方を **A 列** と呼び、元々紙に書かれている数字の並びの方を **B 列** と呼ぶこととする。

$$\begin{array}{lccc} \text{B 列}: & 4,2,1,3 & \longrightarrow & 1,2,3,4 \\ & \updownarrow & & \updownarrow \\ \text{A 列}: & 1,2,3,4 & \longrightarrow & 3,2,4,1 \end{array}$$

3 節の例で言えば、A 列の初期配列が 1,2,...,16 で B 列の最終配列が 1,2,...,16 であることになる。

なお、この A 列の最後の並びを $a(1)$,$a(2)$,...,$a(N)$, B 列の初期配列を $b(1)$,$b(2)$,..., $b(N)$ とすると、任意の j に対し

$$a(b(j)) = j, \quad b(a(j)) = j$$

の関係があることがわかる。これは数学的には、2 つの置換

$$\begin{pmatrix} i \\ a(i) \end{pmatrix}, \quad \begin{pmatrix} i \\ b(i) \end{pmatrix}$$

が逆置換の関係にあることを意味している。

5 必要な回数の評価

A 列の配分によって、何が起こるかを考えてみる。例えば、$N = 6$ のものを 1 回配分してみる。

$$1\ 2\ 3\ 4\ 5\ 6\ \rightarrow \begin{cases} A_1: & 2\ 4\ 5 \\ A_2: & 1\ 3\ 6 \end{cases} \rightarrow (2\ 4\ 5),\ (1\ 3\ 6)$$

この例からもわかるように、A 列はそれぞれの山への配分により 2 つのグループに分けられ、それをまとめて作った結果の列は、各グループだった部分は必ず増加列であり、減少するところは、そのグループの切れ目でのみ起こることになる。

もう 1 回これを配分すると、次のようになる:

$$(1\text{ 回目 }A_1),\ (1\text{ 回目 }A_2) \rightarrow \begin{cases} A_1: & (1\text{ 回目 }A_1),\ (1\text{ 回目 }A_2) \\ A_2: & (1\text{ 回目 }A_1),\ (1\text{ 回目 }A_2) \end{cases}$$
$$\rightarrow (1\text{: }A_1,\ 2\text{: }A_1),\ (1\text{: }A_2,\ 2\text{: }A_1), (1\text{: }A_1,\ 2\text{: }A_2),\ (1\text{: }A_2,\ 2\text{: }A_2)$$

$((1\text{: }A_1,\ 2\text{: }A_2)$ は 1 回目 A_1, 2 回目 A_2 だった部分を意味する$)$

一般には 2 回の手順によりこのような 4 ブロックに分かれるが、実際にはそのうちいくつかは空集合で、3 ブロック以下であることも起こりうる。

このように考えると、「k 回配分した後では、A 列には最高 2^k 個の増加列のブロック、(すなわち $(2^k - 1)$ 箇所の減少箇所) ができることになる」、ということがわかる。これは言いかえれば、「$(k - 1)$ 回の手順では 2^{k-1} 個の増加列ブロックしか作られない」、ということも言えるから、結局次が言えることになる。

命題 1

A 列の最終列に含まれる増加列のブロック数が $(2^{k-1} + 1)$ 個以上ならば、少なくとも k 回の手順が必要。

$M > 2$ の場合は、この $(2^{k-1} + 1)$ を $(M^{k-1} + 1)$ で置きかえれば同じことが言える。

また、容易に次も言える。

系 2

$M^{k-1} < N \leq M^k$ のとき、少なくとも k 回の手順が必要となる B 列の初期配列が存在する。

証明

これは、丁度逆順の B 列の初期配置 N,$(N-1)$,...,3,2,1 がそれに相当する。この B 列の初期配置に対しては、A 列の最終配置も丁度逆順の N,$(N-1)$,...,3,2,1 なので、増加列のブロックは N 個あることになる。仮定より $N \geq M^{k-1}+1$ だから命題 1 より少なくとも k 回の手順が必要。■

6 最適解の構成

ここでは、命題 1 の十分性を与える手順を構成することを考える。すなわち、

$M^{k-1}+1 \leq$ (増加列ブロック数) $\leq M^k$ のとき、k 回の手順で整列化する

ような構成法を考える。このような構成法が見つかれば、命題 1 より方法 A に関しては明らかにそれが最適な手順となる。

A 列の最終形の増加列ブロックが 2 つである場合をまず考える。例えばそれが

2,3,5,1,4,6

である場合、これは、前の (2,3,5) のブロックが A_1 で、後ろの (1,4,6) のブロックが A_2 にあったはずで、これを元の 1 増加列ブロックに直すには、単純にそれを集めて整列化すればよい。

$$(2\ 3\ 5),\ (1\ 4\ 6) \leftarrow \left\{ \begin{array}{ll} A_1: & (2\ 3\ 5) \\ A_2: & (1\ 4\ 6) \end{array} \right. \leftarrow \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ A_2 & A_1 & A_1 & A_2 & A_1 & A_2 \end{array}$$

なお、この表の見方は、本来は矢印の方向に (右から左へ) 配分作業は進むのであるが、今は A 列の最終形から A 列の初期配列に戻すということを考えているので、それを左から右に書き表した。以後、このような表現を用いることにする。

今度は、A 列の最終形の増加列ブロックが 3 つの場合を考える。この場合は一つ前の段階が 2 つの増加列ブロックで、そこからこれが作られたと考えればよい。

例えばそれが 2,3,5,4,1,6 である場合、これは、2 回の配分後の 4 ブロックのうち一つが空集合になっていて、例えば

$$\left\{ \begin{array}{lcl} (2,3,5) & = & 1 \text{ 回目 } A_1,\ 2 \text{ 回目 } A_1, \\ (4) & = & 1 \text{ 回目 } A_2,\ 2 \text{ 回目 } A_1, \\ (1,6) & = & 1 \text{ 回目 } A_2,\ 2 \text{ 回目 } A_2 \end{array} \right.$$

であると考えれば、

$$(2\ 3\ 5),\ (4),\ (1\ 6)\ \leftarrow \begin{cases} A_1: & (2\ 3\ 5) \\ A_2: & (4),\ (1\ 6) \end{cases} \leftarrow\ (2\ 3\ 4\ 5),\ (1\ 6)$$

と、一つ前の 2,3,4,5,1,6 の形に復元できることになる。これは増加列ブロックが 2 つなので、前と同じようにすれば全体を整列化できる。

$M > 2$ の場合も、例えば

$$a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ a_7,\ a_8 \quad (\text{各 } a_j \text{ はそれぞれ増加列ブロック})$$

のように増加列ブロックが 8 つで、$M = 3$ の場合、

$$a_1,\ a_2,\ a_3,\ a_4,\ a_5,\ a_6,\ a_7,\ a_8\ \leftarrow \begin{cases} A_1: & \{a_1,\ a_2\} \\ A_2: & \{a_3,\ a_4,\ a_5\} \\ A_3: & \{a_6,\ a_7,\ a_8\} \end{cases}$$
$$\leftarrow\ \{a_1,\ a_3,\ a_6\},\ \{a_2,\ a_4,\ a_7\},\ \{a_5,\ a_8\}$$

(例えば $\{a_1, a_3, a_6\}$ は、a_1, a_3, a_6 を整列化した増加列ブロックを意味する)。

のようにすれば 3 個の増加列ブロックに復元できる

このようにして、L 個の増加列ブロックのものを、一回戻して $\lceil L/M \rceil$ 個の増加列ブロックにすることができる。

命題 3

A 列の最終列に含まれる増加列のブロック数が M^k 個以下ならば、k 回の手順で生成でき、その手順は上のように構成可能である。

7 並べ直しの具体例

ここでは、いくつか並べ直しの具体例を紹介する。

例 4

元々の数の並び (B 列の初期配列) が

9,4,3,5,8,2,1,7,6

であるとし、$M = 2$ で考える。

1. まず A 列の最終形を求める。

$$\begin{array}{lccc} \text{B 列}: & 9\ 4\ 3\ 5\ 8\ 2\ 1\ 7\ 6 & \rightarrow & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 \\ & \updownarrow & & \updownarrow \\ \text{A 列}: & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9 & \rightarrow & 7\ 6\ 3\ 2\ 4\ 9\ 8\ 5\ 1 \end{array}$$

2. それを増加列ブロックに分割

(7), (6), (3), (2, 4, ,9), (8), (5), (1)

ブロック数は 7 個だから $M = 2$ の場合は 3 回かかることになる。

3. ブロックの統合を行ない、昇順の初期配列に戻していく

$$(7)\ (6)\ (3)\ (2\ 4\ 9)\ (8)\ (5)\ (1) \leftarrow \begin{cases} A_1: & (7)\ (6)\ (3) \\ A_2: & (2\ 4\ 9)\ (8)\ (5)\ (1) \end{cases}$$

$$\overset{(3\text{ 回目})}{\longleftarrow} \begin{array}{l} (2\ 4\ 7\ 9)\ (6\ 8)\ (3\ 5)\ (1) \\ [2\ 2\ 1\ 2 \quad 1\ 2 \quad 1\ 2 \quad 2]\ (\text{ア}) \end{array} \leftarrow \begin{cases} A_1: & (2\ 4\ 7\ 9)\ (6\ 8) \\ A_2: & (3\ 5)\ (1) \end{cases}$$

$$\overset{(2\text{ 回目})}{\longleftarrow} \begin{array}{l} (2\ 3\ 4\ 5\ 7\ 9)\ (1\ 6\ 8) \\ [1\ 2\ 1\ 2\ 1\ 1 \quad 2\ 1\ 1]\ (\text{イ}) \end{array} \leftarrow \begin{cases} A_1: & (2\ 3\ 4\ 5\ 7\ 9) \\ A_2: & (1\ 6\ 8) \end{cases}$$

$$\overset{(1\text{ 回目})}{\longleftarrow} \begin{array}{l} (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9) \qquad (\text{A 列の初期状態}) \\ [2\ 1\ 1\ 1\ 1\ 2\ 1\ 2\ 1]\ (\text{ウ}) \end{array}$$

ここで、(ア), (イ), (ウ) の [ ] 内の列は、その上の行の A 列をどの山へ配分するかを意味する。

4. この手順 (ア), (イ), (ウ) を B 列で逆にたどれば B 列の整列化が行なえる

$$\begin{array}{l} 9\ 4\ 3\ 5\ 8\ 2\ 1\ 7\ 6 \\ [2\ 1\ 1\ 1\ 1\ 2\ 1\ 2\ 1]\ (\text{ウ}) \end{array} \overset{(1\text{ 回目})}{\longrightarrow} \begin{cases} A_1: & 4\ 3\ 5\ 8\ 1\ 6 \\ A_2: & 9\ 2\ 7 \end{cases}$$

$$\rightarrow \begin{array}{l} 4\ 3\ 5\ 8\ 1\ 6\ 9\ 2\ 7 \\ [1\ 2\ 1\ 2\ 1\ 1\ 2\ 1\ 1]\ (\text{イ}) \end{array} \overset{(2\text{ 回目})}{\longrightarrow} \begin{cases} A_1: & 4\ 5\ 1\ 6\ 2\ 7 \\ A_2: & 3\ 8\ 9 \end{cases}$$

$$\rightarrow \begin{array}{l} 4\ 5\ 1\ 6\ 2\ 7\ 3\ 8\ 9 \\ [2\ 2\ 1\ 2\ 1\ 2\ 1\ 2\ 2]\ (\text{ア}) \end{array} \overset{(3\text{ 回目})}{\longrightarrow} \begin{cases} A_1: & 1\ 2\ 3 \\ A_2: & 4\ 5\ 6\ 7\ 8\ 9 \end{cases}$$

$$\rightarrow\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$

例 5

例 4 と同じ初期配列のものを $M = 3$ で整列化する。この場合ブロック数は 7 個だから 2 回でできる。

1. A 列の最終形を求めて増加列ブロックに分割するところまでは例 4 と同じ。

2. ブロックの統合

$$(7)\ (6)\ (3)\ (2\ 4\ 9)\ (8)\ (5)\ (1)\ \leftarrow \begin{cases} A_1: & (7)\ (6) \\ A_2: & (3)\ (2\ 4\ 9) \\ A_3: & (8)\ (5)\ (1) \end{cases}$$

$$\overset{(2\text{ 回目})}{\longleftarrow}\quad \begin{matrix}(3\ 7\ 8)\ (2\ 4\ 5\ 6\ 9)\ (1) \\ [2\ 1\ 3\quad 2\ 2\ 3\ 1\ 2\quad 3]\ (\text{ア})\end{matrix} \quad \leftarrow \begin{cases} A_1: & (3\ 7\ 8) \\ A_2: & (2\ 4\ 5\ 6\ 9) \\ A_3: & (1) \end{cases}$$

$$\overset{(1\text{ 回目})}{\longleftarrow}\quad \begin{matrix}(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9) \\ [3\ 2\ 1\ 2\ 2\ 2\ 1\ 1\ 2]\ (\text{イ})\end{matrix} \qquad (\text{A 列の初期状態})$$

3. この手順 (ア), (イ) を B 列で逆にたどればB 列の整列化が行なえる

$$\begin{matrix} 9\ 4\ 3\ 5\ 8\ 2\ 1\ 7\ 6 \\ [3\ 2\ 1\ 2\ 2\ 2\ 1\ 1\ 2]\ (\text{イ})\end{matrix} \overset{(1\text{ 回目})}{\longrightarrow} \begin{cases} A_1: & 3\ 1\ 7 \\ A_2: & 4\ 5\ 8\ 2\ 6 \\ A_3: & 9 \end{cases}$$

$$\begin{matrix}\rightarrow\quad 3\ 1\ 7\ 4\ 5\ 8\ 2\ 6\ 9 \\ [2\ 1\ 3\ 2\ 2\ 3\ 1\ 2\ 3]\ (\text{ア})\end{matrix} \overset{(2\text{ 回目})}{\longrightarrow} \begin{cases} A_1: & 1\ 2 \\ A_2: & 3\ 4\ 5\ 6 \\ A_3: & 7\ 8\ 9 \end{cases}$$

$$\rightarrow\quad 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$

8 方法 B の場合

方法 B の場合は、方法 A のやり方を応用して解くことができる。

この場合、1 回の手順では、5 節で述べたのとは異なり、1 回の配分では元の増加列は減少列へと変化する:

$$1\ 2\ 3\ 4\ 5\ 6\ \rightarrow \begin{cases} A_1: & 5\ 4\ 2 \\ A_2: & 6\ 3\ 1 \end{cases} \rightarrow (5\ 4\ 2),\ (6\ 3\ 1)$$

もう 1 回これを配分すれば、また増加列へと変わる。

よって、方法 A の場合に偶数回で終了する場合には方法 B でも偶数回で整列化でき、奇数回で終了する場合には、方法 B で同様のことを行なうと降順になってしまうことになる (正確には B 列ではそうではないが少なくとも A 列ではそうなる)。その場合は、もう 1 回、一つの山にだけ配分すれば正しく昇順に直せることになる。

これだけみると、方法 B の方が方法 A より手数がかかりそうに見えるかもしれないが、実際にはそうではない。例えば、丁度逆順の並び N,$(N-1)$,...,3,2,1 を考えればわかるが、これは方法 A では最悪の手順数が必要となるが、方法 B なら 1 つの山に配るだけで 1 回で終了する。つまり、方法 B の場合は、増加列ブロック数だけでなく、減少列ブロック数 によっても最短手順が決まることになる。

増加列ブロック数を s_1、減少列ブロック数を s_2 とすると、これらはそれぞれ減少箇所、増加箇所より 1 ずつ大きい。減少箇所、増加箇所はもちろん同じ場所にはなく、よってそれらの和は $(N-1)$ であるから、

$$s_1 + s_2 = N + 1 \tag{2}$$

が言える。

なお、減少列ブロック数は、A 列を逆転 (前後を反対にした列) の増加列ブロック数に等しく、実際の作業でも、実はこの逆転させた列の増加列で考えることになる。

容易にわかるように、減少列ブロック数で考えた場合は、増加列ブロックで考えるのとは逆で、これが方法 A で奇数回で終わるブロック数ならば、方法 B では同じ回数で昇順にでき、方法 A で隅数回で終わるブロック数ならば、方法 B では同じ回数では降順になってしまうのでもう 1 回必要となる。

よって、この場合は、

$$\begin{cases} M^{2J-2} < s_1 \leq M^{2J-1} & \text{ならば } (2J-1)+1 = 2J \text{ 回、} \\ M^{2J-1} < s_1 \leq M^{2J} & \text{ならば } 2J \text{ 回} \end{cases}$$

となるので、結局 $M^{2J-2} < s_1 \leq M^{2J}$ ならば $2J$ 回となる。同様に考えると結局次が言える。

命題 6

方法 B の場合、

$$\begin{cases} M^{2J-2} < s_1 \leq M^{2J} & \text{ならば } 2J \text{ 回、} \\ M^{2J-1} < s_2 \leq M^{2J+1} & \text{ならば } (2J+1) \text{ 回} \end{cases}$$

で昇順に直せるので、この小さい方を選択すればよい。

これが最適な回数であることも容易に示される。$M = 2$ でそれを説明する。例えば、A 列の 1,2,...,N は、方法 B の 1 回の操作では高々 2 つの減少列ブロックからなる。よって、3 つ以上の減少列ブロックを含む A 列の最終形は、方法 B 1 回では元には戻せない。

2 回では高々 4 つの増加列ブロックからなる列になるので、5 つ以上の増加列ブロックを持つものは方法 B 2 回では元には戻せない。このように考えれば、命題 6 の評価が最適であることがわかる。

命題 7

方法 B の場合、昇順への整列化には最低でも命題 6 の回数だけ必要

9 方法 B の整列化の例

ここでは、方法 B の場合の整列化の方法の具体例を紹介する。8 節でも述べたように、実際には減少列ブロックではなく、逆転させた増加列で考える方が自然であるが、それについても説明する。簡単のため、$M = 2$ で考える。

例えば、B 列の初期配列が

6,2,8,3,7,1,5,4

である場合、A 列の最終形は

6,2,4,8,7,1,5,3

であるので、

$$\begin{cases} \text{増加列ブロック}: & (6),\ (2\ 4\ 8),\ (7),\ (1\ 5),\ (3) & 5\ \text{個} \\ \text{減少列ブロック}: & (6\ 2),\ (4),\ (8\ 7\ 1),\ (5\ 3) & 4\ \text{個} \end{cases}$$

となり、増加列ブロックで行なうと方法 A ならば 3 回で終わるが、方法 B だと 3 回では降順になってしまうので、全部を一度ひっくり返すために 4 回かかることになる。

減少列ブロックで考えると、これを統合すると 2 回で一つのブロックにはできるが、最終的には降順なので、もう一回必要で、3 回かかることになる。

まず、この減少列ブロックの復元を考えてみる。

$$(6\ 2)\ (4)\ (8\ 7\ 1)\ (5\ 3) \leftarrow \begin{cases} A_1: & (6\ 2)\ (4) \\ A_2: & (8\ 7\ 1)\ (5\ 3) \end{cases}$$

$$\overset{(3\text{回目})}{\longleftarrow} \begin{array}{l} (3\ 4\ 5)\ (1\ 2\ 6\ 7\ 8) \\ [2\ 1\ 2 \quad 2\ 1\ 1\ 2\ 2]\ (\text{ア}) \end{array} \leftarrow \begin{cases} A_1: & (3\ 4\ 5) \\ A_2: & (1\ 2\ 6\ 7\ 8) \end{cases}$$

$$\overset{(2\text{回目})}{\longleftarrow} \begin{array}{l} (8\ 7\ 6\ 5\ 4\ 3\ 2\ 1) \\ [2\ 2\ 2\ 1\ 1\ 1\ 2\ 2]\ (\text{イ}) \end{array} \leftarrow \begin{cases} A_1: & (8\ 7\ 6\ 5\ 4\ 3\ 2\ 1) \\ A_2: & \end{cases}$$

$$\overset{(1\text{回目})}{\longleftarrow} \begin{array}{l} (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8) \\ [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]\ (\text{ウ}) \end{array} \quad (\text{A 列の初期状態})$$

これを B 列で行なってみる。

$$\begin{array}{l} 6\ 2\ 8\ 3\ 7\ 1\ 5\ 4 \\ [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1]\ (\text{ウ}) \end{array} \overset{(1\text{回目})}{\longrightarrow} \begin{cases} A_1: & 4\ 5\ 1\ 7\ 3\ 8\ 2\ 6 \\ A_2: & \end{cases}$$

$$\rightarrow \begin{array}{l} 4\ 5\ 1\ 7\ 3\ 8\ 2\ 6 \\ [2\ 2\ 2\ 1\ 1\ 1\ 2\ 2]\ (\text{イ}) \end{array} \overset{(2\text{回目})}{\longrightarrow} \begin{cases} A_1: & 8\ 3\ 7 \\ A_2: & 6\ 2\ 1\ 5\ 4 \end{cases}$$

$$\rightarrow \begin{array}{l} 8\ 3\ 7\ 6\ 2\ 1\ 5\ 4 \\ [2\ 1\ 2\ 2\ 1\ 1\ 2\ 2]\ (\text{ア}) \end{array} \overset{(3\text{回目})}{\longrightarrow} \begin{cases} A_1: & 1\ 2\ 3 \\ A_2: & 4\ 5\ 6\ 7\ 8 \end{cases}$$

$$\rightarrow 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$$

ところが、この例では 1 回目に全体を逆転させる手順が入っている。降順でもいい場合、あるいは降順への整列化の場合への応用も考えるとこの方法は無駄で、むしろ最後に全体を逆転させる手順が来る方がよい。

これは A 列で言えば、A 列の最終形を最初に逆転させ、その増加列ブロックを統合することに他ならない。今度はその手順を紹介する。逆転された A 列の最終形 3,5,1,7,8,4,2,6 から開始する。

$$(3\ 5)\ (1\ 7\ 8)\ (4)\ (2\ 6) \leftarrow \begin{cases} A_1: & (3\ 5)\ (1\ 7\ 8) \\ A_2: & (4)\ (2\ 6) \end{cases}$$

$$\overset{(2\text{回目})}{\longleftarrow} \begin{array}{l} (8\ 7\ 6\ 2\ 1)\ (5\ 4\ 3) \\ [1\ 1\ 2\ 2\ 1 \quad 1\ 2\ 1]\ (\text{ア})' \end{array} \leftarrow \begin{cases} A_1: & (8\ 7\ 6\ 2\ 1) \\ A_2: & (5\ 4\ 3) \end{cases}$$

$$\overset{(1\text{回目})}{\longleftarrow} \begin{array}{l} (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8) \\ [1\ 1\ 2\ 2\ 2\ 1\ 1\ 1]\ (\text{イ})' \end{array} \quad (\text{A 列の初期状態})$$

これで B 列を整列化する。

$$\begin{array}{ll} & \begin{array}{l} 6\ 2\ 8\ 3\ 7\ 1\ 5\ 4 \\ [1\ 1\ 2\ 2\ 2\ 1\ 1\ 1]\ (\text{イ})' \end{array} \xrightarrow{(1\text{ 回目})} \begin{cases} A_1: & 4\ 5\ 1\ 2\ 6 \\ A_2: & 7\ 3\ 8 \end{cases} \\ \rightarrow & \begin{array}{l} 4\ 5\ 1\ 2\ 6\ 7\ 3\ 8 \\ [1\ 1\ 2\ 2\ 1\ 1\ 2\ 1]\ (\text{ア})' \end{array} \xrightarrow{(2\text{ 回目})} \begin{cases} A_1: & 8\ 7\ 6\ 5\ 4 \\ A_2: & 3\ 2\ 1 \end{cases} \\ \rightarrow & \begin{array}{l} 8\ 7\ 6\ 5\ 4\ 3\ 2\ 1 \\ [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1] \end{array} \xrightarrow{(3\text{ 回目})} \begin{cases} A_1: & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \\ A_2: & \end{cases} \\ \rightarrow & 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8 \end{array}$$

これで、回数は変わらずに、逆転は最後に行なうようにすることができる。

この方法ならば、コンピュータのプログラム化も容易で、増加列ブロックの統合部分を作ってしまえば、それを方法 A でも方法 B でも利用することができることになる。

実は、ある A 列に対する方法 A の手順列が、$e_1, e_2, \ldots e_K$ の場合、それと同じ A 列を用いた方法 B の手順列は、$\bar{e_1}, \hat{e_2}, \bar{e_3}, \hat{e_4}, \ldots$ となることが容易示される。ここで、$\bar{e}$ は e の置く位置を反転させたもの、すなわち、A_j の山へ置くことを A_{M-j+1} の山へ置くことと読み変えたものであり、$\hat{e}$ は e の前後を逆転させたもの、すなわち、手順の前後を入れかえたものである。

よって、増加列ブロックの統合ルーチンによる手順列を適当に反転、逆転させれば方法 A でも方法 B でも、整列化手順列の生成ができることになる。

10 方法 A, B の比較

前節までに、方法 A, B の最適手順が構成されたことになるが、これらの方法の平均回数を比較してみることとする。この平均回数 とは、N 枚の $N!$ 種類の順列に対する手順数の平均を意味し、それらが均等にランダムに起こるとした場合の手順数の期待値、と言うこともできる。

例えば、$N = 4, M = 2$ の場合、増加列ブロック数 s_1、減少列ブロック数 s_2 と方法 A、方法 B の回数は以下の関係にある:

s_1	s_2	A	B
1	4	0	0
2	3	1	2
3	2	2	1
4	1	2	1

一見、方法 B の方がかなり少ないように見えるかもしれないが、4! 種類のすべての順列に対してその回数を書きあげてみると以下のようになる。

A 列の最終形	s_1	s_2	A	B	A 列の最終形	s_1	s_2	A	B
1 2 3 4	1	4	0	0	3 1 2 4	2	3	1	2
1 2 4 3	2	3	1	2	3 1 4 2	3	2	2	1
1 3 2 4	2	3	1	2	3 2 1 4	3	2	2	1
1 3 4 2	2	3	1	2	3 2 4 1	3	2	2	1
1 4 2 3	2	3	1	2	3 4 1 2	2	3	1	2
1 4 3 2	3	2	2	1	3 4 2 1	3	2	2	1
2 1 3 4	2	3	1	2	4 1 2 3	2	3	1	2
2 1 4 3	3	2	2	1	4 1 3 2	3	2	2	1
2 3 1 4	2	3	1	2	4 2 1 3	3	2	2	1
2 3 4 1	2	3	1	2	4 2 3 1	3	2	2	1
2 4 1 3	2	3	1	2	4 3 1 2	3	2	2	1
2 4 3 1	3	2	2	1	4 3 2 1	4	1	2	1

この総数を比較すると、平均は確かに方法 B の方が小さいが、その違いはわずかであることがわかる:

方法	0 回	1 回	2 回	平均回数
方法 A	1 通り	11 通り	12 通り	35/24
方法 B	1 通り	12 通り	11 通り	34/24

s_1 と総数の関係を表にすると以下のようになるので、これは $s_1 = 4$ のところの違いが出ているようにも見える。

s_1	1	2	3	4
総数	1	11	11	1

一般の N に対する考察を行うために、この増加列ブロック数毎の総数を求めることができれば、平均の計算はそれに各 s_1 に対する回数をかければよい。よって、今度はその増加列ブロック数毎の総数の計算を行なう。

11 増加列ブロック数の分布

$1 \sim N$ の $N!$ 通りの順列に対して、増加列ブロック数がどのように分布しているのかを調べることにする。

定義 8

$A_k^N = 1 \sim N$ の順列の中で、増加列ブロック数が k であるものの個数 $(1 \leq k \leq N)$

例えば、$N = 3$ のときは、$A_1^3 = 1$ ((1,2,3) の一つ), $A_2^3 = 4$, $A_3^3 = 1$ ((3,2,1) の一つ)、$N = 4$ のときは $(A_1^4, A_2^4, A_3^4, A_4^4) = (1, 11, 11, 1)$ である。容易に次が言える。

$$A_1^N = A_N^N = 1, \quad \sum_{k=1}^{N} A_k^N = N!$$

また、対称性

$$A_k^N = A_{N-k+1}^N \tag{3}$$

も成り立ちそうに見えるが、これはそう難しくなく示される。

例えば、1,3,2,4 は増加列 2 ブロックであるが、これを逆転させた列 4,2,3,1 は増加列 3 (=4-2+1) ブロックになる。この逆転列の対応を考えると、増加列 2 ブロックのものと増加列 3 ブロックのものが 1 対 1 に対応することになる。よって $A_2^4 = A_3^4$ となる。一般の N,k の場合も同様である。

以降は、この A_k^N を N, k の式で表わすことを考えてみる。例えば、$N = 3$, $k = 2$ の場合の A_2^3 は、

(1,3,2), (2,1,3), (2,3,1), (3,1,2)

の 4 つであるが、これは 1,2,3 の数字を 2 つの組に分けて、それぞれを昇順に整列化したものを並べて書いた、と見ることができる:

$$1, 2, 3 \ \rightarrow \ \left\{ \begin{array}{l} 2, \ 3 \\ 1 \end{array} \right. \ \rightarrow \ (2, 3, 1)$$

1,2,3 を 2 つのブロックに分けるやり方は、各ブロックに B_1, B_2 の番号をつけて、1,2,3 各々がそのどちらに入るかを考えれば $2^3 = 8$ 通りあるが、しかしこれには増加列 2 ブロックを生成しない余計なもの 4 つが含まれる:

{(1,2,3),()}, {(1,2),(3)}, {(1),(2,3)}, {(),(1,2,3)}　(前の方が B_1)

これらは、本来 1 ブロックの並び (1,2,3) に仕切りを入れて分けたものの個数 $= 3+1 = 4$ に等しい。よって、

$$A_2^3 = 2^3 - 4 = 4$$

となる。

同様に、$N=4,\ k=2$ の場合は、2^4 通りから $(1,2,3,4)$ に一つの仕切りを入れる入れ方である 5 通りを引いた

$$A_2^4 = 2^4 - 5 = 11$$

となる。

$N=4,\ k=3$ の場合も、$1,2,3,4$ を 3 つのブロックに分けてそれぞれを整列化して並べて書いて、そこから余計なものを引けばよい。3 つのブロックの分け方の総数は 3^4、余計なものは、$(1,2),(3),(4)$ のように、1 ブロックのものを 3 ブロックに分けたもの、および $(1,2),(4),(3)$ のように、2 ブロックのものを 3 ブロックに分けたものの 2 種類がある。

2 ブロックのものを 3 ブロックに分けるやり方は、例えば $(1,2,4),(3)$ の場合、

$$\{(),(1,2,4),(3)\}(B_1 \text{ が空}),\ \{(1),(2,4),(3)\},\ \{(1,2),(4),(3)\},$$
$$\{(1,2,4),(),(3)\}(B_2 \text{ が空}),\ \{(1,2,4),(3),()\}(B_3 \text{ が空})$$

の 5 通りある $(=5$ 箇所に 1 つの仕切りを入れる方法$)$。2 ブロックは全部で A_2^4 だけあるから、2 ブロックのものを 3 ブロックに分けたものの総数は $5\cdot A_2^4$ となる。

1 ブロックのものを 3 ブロックに分けるやり方は、例えば

$$(1),(),(2,3,4)$$

のようになるが、これは 5 箇所に 2 つの仕切りを入れる入れ方で、$\circ$ 4 つと$\times$ 2 つを並べる総数に等しく、よってこれは

$$\left(\begin{array}{c} 4+2 \\ 2 \end{array} \right)$$

に等しい。よって、結局

$$A_3^4 = 3^4 - 5A_2^4 - \left(\begin{array}{c} 4+2 \\ 2 \end{array} \right) A_1^4 = 81 - 55 - 15 = 11$$

となる。

これを続けると、結局次の漸化式が得られることになる。

$$\left\{ \begin{array}{ll} A_k^N & = k^N - \displaystyle\sum_{j=1}^{k-1} \left(\begin{array}{c} N+j \\ j \end{array} \right) A_{k-j}^N \quad (k \geq 2), \\ A_1^N & = 1 \end{array} \right. \tag{4}$$

ここから A_k^N の式を推論するために、2,3 の計算をしてみる。

$$\begin{aligned}
A_2^N &= 2^N - \left(\begin{array}{c} N+1 \\ 1 \end{array} \right) A_1^N = 2^N - (N+1), \\
A_3^N &= 3^N - \left(\begin{array}{c} N+1 \\ 1 \end{array} \right) A_2^N - \left(\begin{array}{c} N+2 \\ 2 \end{array} \right) A_1^N \\
&= 3^N - (N+1)\{2^N - (N+1)\} - \frac{(N+2)(N+1)}{2} \\
&= 3^N - (N+1)2^N + (N+1)^2 - \frac{(N+2)(N+1)}{2} \\
&= 3^N - (N+1)2^N + \frac{N(N+1)}{2} \\
&= 3^N - (N+1)2^N + \left(\begin{array}{c} N+1 \\ 2 \end{array} \right)
\end{aligned}$$

同様に、

$$A_4^N = 4^N - \left(\begin{array}{c} N+1 \\ 1 \end{array} \right) 3^N + \left(\begin{array}{c} N+1 \\ 2 \end{array} \right) 2^N - \left(\begin{array}{c} N+1 \\ 3 \end{array} \right)$$

となることも言えるので、一般に、

$$A_k^N = \sum_{j=0}^{k-1} (-1)^j \left(\begin{array}{c} N+1 \\ j \end{array} \right) (k-j)^N \tag{5}$$

であることが予想される。これを、(4) を用いて証明する。

そのためにまず、次の補題を示す。

補題 9

全て の $1 \leq q \leq N+1$ に対して、次が成り立つ。

$$\sum_{j=1}^{q} (-1)^{j+1} \left(\begin{array}{c} N+j \\ j \end{array} \right) \left(\begin{array}{c} N+1 \\ q-j \end{array} \right) = \left(\begin{array}{c} N+1 \\ q \end{array} \right) \tag{6}$$

証明

$$\left(\begin{array}{c} N+j \\ j \end{array} \right) = \left(\begin{array}{c} N+j \\ N \end{array} \right) = \frac{(N+j)(N+j-1)\cdots(1+j)}{N!}$$

であり、これは関数 $x^{N+j}/N!$ を N 回微分した係数、すなわち N 回微分して $x=1$ としたものに等しい:

$$\left(\frac{d}{dx}\right)^N \left(\frac{x^{N+j}}{N!}\right) = \frac{(N+j)(N+j-1)\cdots(1+j)}{N!}x^j$$

よって、

$$f(x) = \sum_{j=1}^{q}(-1)^{j+1}\left(\begin{array}{c} N+1 \\ q-j \end{array}\right)\frac{x^{N+j}}{N!} \tag{7}$$

とすれば、(6) の左辺は $f(x)$ を N 回微分して $x=1$ を代入したもの、すなわち $f^{(N)}(1)$ に等しい。

ところで、$f(x)$ は $(N+1)$ から $(N+q)$ 次の項からなる多項式であるが、これに x^{N-1} 次以下の項をつけ加えても N 回微分には影響はない。よって、

$$g(x) = \sum_{j=q-N-1}^{q}(-1)^{j+1}\left(\begin{array}{c} N+1 \\ q-j \end{array}\right)\frac{x^{N+j}}{N!} \tag{8}$$

とすると、

$$\begin{aligned} g(x) &= f(x) + \sum_{j=q-N-1}^{0}(-1)^{j+1}\left(\begin{array}{c} N+1 \\ q-j \end{array}\right)\frac{x^{N+j}}{N!} \\ &= f(x) - \left(\begin{array}{c} N+1 \\ q \end{array}\right)\frac{x^N}{N!} + \sum_{j=q-N-1}^{-1}(-1)^{j+1}\left(\begin{array}{c} N+1 \\ q-j \end{array}\right)\frac{x^{N+j}}{N!} \end{aligned}$$

となるので、これを N 回微分すると、

$$g^{(N)}(x) = f^{(N)}(x) - \left(\begin{array}{c} N+1 \\ q \end{array}\right) \tag{9}$$

となる。ところで $g(x)$ は、$j=q-i$ として変形すると

$$\begin{aligned} g(x) &= \sum_{i=0}^{N+1}(-1)^{q+1-i}\left(\begin{array}{c} N+1 \\ i \end{array}\right)\frac{x^{N+q-i}}{N!} \\ &= \frac{(-1)^{N+q}}{N!}x^{q-1}\sum_{i=0}^{N+1}(-1)^{N+1-i}\left(\begin{array}{c} N+1 \\ i \end{array}\right)x^{N+1-i} \\ &= \frac{(-1)^{N+q}}{N!}x^{q-1}(1-x)^{N+1} \end{aligned}$$

であり、N 回微分を行なうと、積の微分で現れる全ての項に少なくとも $(1-x)^1$ が含まれることになるので、明らかに $g^{(N)}(1)=0$ である。よって、(9) より

$$f^{(N)}(1)=g^{(N)}(1)+\left(\begin{array}{c}N+1\\q\end{array}\right)=\left(\begin{array}{c}N+1\\q\end{array}\right)$$

となって (6) の右辺に等しいことが言える。■

(5) の証明

k に関する帰納法により (5) を証明する。

$k=1$ のときは、

$$(5)\text{ の右辺}=(-1)^0\left(\begin{array}{c}N+1\\0\end{array}\right)(1-0)^N=1$$

より O.K.

$k=1\sim(m-1)$ に対して (5) が成り立つとして、$k=m$ に対して (5) が成り立つことを示す $(m\geq 2)$。(4) より、

$$A_m^N=m^N-\sum_{j=1}^{m-1}\left(\begin{array}{c}N+j\\j\end{array}\right)A_{m-j}^N$$

であるが、右辺の A_{m-j}^N には仮定より (5) が使えるので、

$$\begin{aligned}
A_m^N &= m^N-\sum_{j=1}^{m-1}\left(\begin{array}{c}N+j\\j\end{array}\right)\sum_{p=0}^{m-j-1}(-1)^p\left(\begin{array}{c}N+1\\p\end{array}\right)(m-j-p)^N\\
&= m^N-\sum_{j=1}^{m-1}\sum_{q=j}^{m-1}\left(\begin{array}{c}N+j\\j\end{array}\right)(-1)^{q-j}\left(\begin{array}{c}N+1\\q-j\end{array}\right)(m-q)^N\quad(p=q-j)\\
&= m^N-\sum_{q=1}^{m-1}\sum_{j=1}^{q}\left(\begin{array}{c}N+j\\j\end{array}\right)(-1)^{q-j}\left(\begin{array}{c}N+1\\q-j\end{array}\right)(m-q)^N\\
&= m^N+\sum_{q=1}^{m-1}(-1)^q(m-q)^N\sum_{j=1}^{q}(-1)^{j+1}\left(\begin{array}{c}N+j\\j\end{array}\right)\left(\begin{array}{c}N+1\\q-j\end{array}\right)
\end{aligned}$$

となる。よって、補題 9 により

$$A_m^N=m^N+\sum_{q=1}^{m-1}(-1)^q(m-q)^N\left(\begin{array}{c}N+1\\q\end{array}\right)=\sum_{q=0}^{m-1}(-1)^q(m-q)^N\left(\begin{array}{c}N+1\\q\end{array}\right)$$

となり、(5) が $k = m$ に対して成り立つことになる。■

なお、いくつかの N に対する A_k^N の値を紹介すると、以下のようにその値は N の増加に対して急激に大きくなることがわかる。

N	$(A_1^N, A_2^N, \ldots, A_N^N)$
3	(1, 4, 1)
4	(1, 11, 11, 1)
5	(1, 26, 66, 26, 1)
6	(1, 57, 302, 302, 57, 1)
7	(1, 120, 1191, 2416, 1191, 120, 1)
8	(1, 247, 4293, 15619, 15619, 4293, 247, 1)
9	(1, 502, 14608, 88234, 156190, 88234, 14608, 502, 1)
10	(1, 1013, 47840, 455192, 1310354, 1310354, 455192, 47840, 1013, 1)

12 平均回数の数値計算結果

整列化に必要な手順数は、A 列の増加列ブロック数によって決まり、命題 1, 3, 6, 7 より、$s_1 = k$ のとき、

$$\begin{cases} \text{方法 A}: & M^{j-1} < k \le M^j & \Rightarrow \ (\text{手順数}) = j \\ \text{方法 B}: & \begin{cases} M^{2j-2} < k \le M^{2j}, \\ M^{2l-1} < N - k + 1 \le M^{2l+1} \end{cases} & \Rightarrow \ (\text{手順数}) = \min\{2j, 2l+1\} \end{cases}$$

である。この増加列ブロック数 k に対する方法 A の手順数を a_k^N, 方法 B の手順数を b_k^N と書くことにする。

例えば、$M = 2$, $N = 5$ の場合は

k	1	2	3	4	5
a_k^N	0	1	2	2	3
$2j$	0	2	2	2	4
$2l+1$	3	3	3	1	1
b_k^N	0	2	2	1	1

$N = 10$ の場合は

k	1	2	3	4	5	6	7	8	9	10
a_k^N	0	1	2	2	3	3	3	3	4	4
$2j$	0	2	2	2	4	4	4	4	4	4
$2l+1$	5	5	3	3	3	3	3	3	1	1
b_k^N	0	2	2	2	3	3	3	3	1	1

となる。よって、11 節の表により、$N=5$ の場合方法 A (a_k^N) の平均 $\bar{a^N}$ は、

$$\bar{a^N} = \frac{1\times 0 + 26\times 1 + 66\times 2 + 26\times 2 + 1\times 3}{5!} = \frac{213}{120} = 1.775$$

方法 B (b_k^N) の平均 $\bar{b^N}$ は、

$$\bar{b^N} = \frac{1\times 0 + 26\times 2 + 66\times 2 + 26\times 1 + 1\times 1}{5!} = \frac{211}{120} = 1.758$$

となり、大きな違いはない。$N=10$ の場合は 3) より、

$$\begin{aligned}
\bar{a^N} &= \frac{A_2^{10} + 2(A_3^{10}+A_4^{10}) + 3(A_5^{10}+A_6^{10}+A_7^{10}+A_8^{10}) + 4(A_9^{10}+A_{10}^{10})}{10!} \\
&= \frac{4A_1^{10} + 5A_2^{10} + 5A_3^{10} + 5A_4^{10} + 6A_5^{10}}{10!}, \\
\bar{b^N} & \\
&= \frac{2(A_2^{10}+A_3^{10}+A_4^{10}) + 3(A_5^{10}+A_6^{10}+A_7^{10}+A_8^{10}) + A_9^{10} + A_{10}^{10}}{10!} \\
&= \frac{A_1^{10} + 3A_2^{10} + 5A_3^{10} + 5A_4^{10} + 6A_5^{10}}{10!},
\end{aligned}$$

となり、値がかなり大きい $k=N/2$ 付近の A_k^N については両者の係数が一致し、異なるのは値がかなり小さい A_k^N なので、それほど違いがないことになる。実際、数値でみると、$N=10$ の場合は

$$\begin{cases}
\bar{a^N} &= \dfrac{10382353}{3628800} &= 2.8611 \\
\bar{b^N} &= \dfrac{10380324}{3628800} &= 2.8605 \\
\bar{a^N} - \bar{b^N} &= \dfrac{2029}{3628800} &= 5.59\times 10^{-4}
\end{cases}$$

となる。

同様に、$N=10,20,50,100$ での値をコンピュータに数値計算させた結果は以下の通りである。

N	10	20	50	100
$\bar{a^N}$	2.8611	3.9390	5.000	6.000
$\bar{b^N}$	2.8605	3.9390	5.000	6.000
$\bar{a^N} - \bar{b^N}$	5.59×10^{-4}	8.47×10^{-7}	3.60×10^{-6}	1.03×10^{-10}

$M = 2, N = 100$ くらいでも 6 回くらいで終わり (最悪でも 7 回)、方法 B の方がごくわずかに速いが、ほとんど違いはないことがわかる。

なお、この表に見られるように、この差は N に関して単調減少ではない。それについては今後さらに調べてみたいと思う。

また、$N = 100$ のようにかなり大きな N に対する A_k^N の計算は実はコンピュータでも容易ではなく、計算式としては $(4), (5)$ のどちらを使っても、それらの式を普通に計算すると桁落ちなどによるものと思われる誤差が大きく発生してしまう。

例えば、A_{50}^{100} の $(4), (5)$ の最初の項である 50^{100} は $100 \log_{10} 50 = 169.9$ より 170 桁であるが、実際の A_{50}^{100} は 158 桁しかなく、つまり $(4), (5)$ の第 2 項目以降の項による引き算により上位の桁が下がってしまう。単純な倍精度計算で計算したところ、本来は 1 になるべき $\sum_{k=1}^{100} A_k^{100}/100!$ の値が 1 を大きく超えてしまった。

このような場合、桁落ちが起きないような、同符号のみの項からなるような計算式を求めるか、あるいは多倍長演算を行なう、などの必要があるが、今回は 170 桁の多倍長整数演算を用いて (4) により計算を行なった。しかし、同符号のみの項による展開式があれば計算は容易になるはずなので、それについても今後考えてみたいと思う。

13 最後に

履修者数が多い講義だと、実際に 100 枚近くの答案用紙を整列化することもある。今回の考察により、それは通常行っている方法 B では $M = 2$ なら最悪で 7 回、$M = 3$ なら最悪 4 回で済むことがわかった。

十分実用になる回数のように思うが、実際には 7×100 回、4×100 回の紙の配布を行なうことになるので、驚くほど早く済むわけでもない。

ただ、この配り分けるやり方は機械化も可能だろうと思うので、もしかして将来そういう機械ができると、我々の採点作業も少しは楽になってくれるかもしれない。

参考文献

[1] 森山俊「コンピュータを利用したソーティング手順の作成」新潟工科大学 情報電子工学科卒業論文 (2006 年 2 月)