ショート・ノート

順回表現 queue, deque の入出力アルゴリズムについて*

今　宮　淳　美**

Abstract

This note presents the "effective" algorithms for the intersion and the deletion into/from the circular queue and the circular deque.

According to the old algorithms, M-1 items Can be in queue (or the deque) of M cells at one time with out overflow.

But according to our algorithms to be presented, M items can be in the queue (or the deque) of M cells at one time without overflow.

In these algorithms, the extra variables, except F, R which are respectively a front and a rear variables, are not used.

1. まえがき

基本的なアルゴリズムの設計，解析について，従来のままで見過ごされているのが多いと思われる．このノートでは，そのような基本的アルゴリズムのひとつについて，改良を試みる．

線形リストを表現するひとつの方法として，順回表現がある．文献1)には，M 個のメモリ・セルをもつ順回表現 queue および deque に最高 M-1 個のデータが格納されるアルゴリズムが示されている(文献1); p. 241, EX. 2. 2. 2-1, -2: p. 248, p. 535 参照)．しかし，各メモリ・セルの領域が大なる場合，1個のメモリ・セルも無駄にはできない．

このノートでは，最大 M 個のデータを格納する有効な入出力アルゴリズムが示される．アルゴリズムの評価としては，用いられる変数を少なく，しかも各変数についての計算のオーバヘッドを最小にすることを考える．各アルゴリズムは，algol-like 言語で示される．

2. queue

文献 1) によれば，順回表現 queue の入力 (insertion)，出力 (deletion) アルゴリズムは，各々(1)，(2)で示される．

初期値：$R=F=1$,
　　R, F は各々，queue (deque) の後部 (rear)，前部 (front) を示す変数．
　　M：queue (deque) のメモリセル数．

Q⇐Y (insert into queue)

```
if R=M then R←1
else R←R+1;
if R=F then OVERFLOW
else Q[R]←Y;
```
(1)

Y⇐**Q** (delete from queue)

```
if R=F then UNDERFLOW;
if F=M then F←1
else F←F+1;
Y←Q[F];
```
(2)

上記のアルゴリズムに従えば，queue 中には，最大 M-1 個のデータが格納される．メモリ中の1個のセルが，OVERFLOW UNDERFLOW 検出のために使われている．UNDERFLOW および OVERFLOW なしで，いくつかの入出力後の queue の様子は **Fig. 1** (次頁参照) で示される．変数 F の位置に注意されたい．

最大 M 個のデータを格納するためには，(1)，(2)で用いられた変数 R, F 以外に新らたに変数を加えることが，すぐに考えつくかもしれない．たとえば，変

* On the insertion and the deletion algorithms for the circular queue and deque by Atsumi IMAMIYA (Depertment of Computer Science Yamanashi University)

** 山梨大学計算機科学科

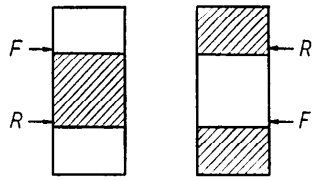

Fig. 1

数 N に queue 中のデータを数えさせるアルゴリズムは，次の(3)，(4)で示される．

初期値：$R=F=1$，$N=0$

Q⇐Y (insert into queue)

```
if N=M then OVERFLOW
else
  if R=M then R←1
  else R←R+1;
Q[R]←Y;
N←N+1;
```
(3)

Y⇐**Q** (delete from queue)

```
if N=0 then UNDERFLOW
if F=M then F←1
else F←F+1;
Y→Q[F];
N→N−1;
```
(4)

しかし，OVERFLOW，UNDERFLOW を検出するのに N を用いることは，N の計算のオーバヘッドおよび情報が多すぎる点で，アルゴリズム(3)，(4)はあまり良いとは言えない．

値として 0, 1 をとる変数を用いて，OVERFLOW，UNDERFLOW を検出する入出力アルゴリズムも考えられるが，次に示すアルゴリズムが最良であると思われる．

初期値：$F=R=0$

Q⇐Y (insert into queue)

```
R←R−1;
if R=F then OVERFLOW;
if R≤0 then R←M;
Q[R]←Y;
```
(5)

Y⇐**Q** (delete from queue)

```
if F=0 then
      if R=0 them UNDERFLOW
      else F←M;
Y←Q[F];
if F=R then (R←0, F←R)
else F←F−1;
```
(6)

* (5)は若干変更される．(10)以下参照．

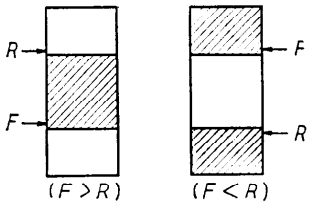

Fig. 2

上記のアルゴリズム(5)，(6)では M 個のデータが格納される UNDERFLOW および OVERFLOW なしで，いくつかの入出力後の queue の様子は，**Fig. 2** で示される (Fig. 1 と Fig. 2 を比較されたい)．しかも(5)，(6)では(1)，(2)で用いた変数 R, F 以外は，用いられていない．R, F の初期値に注意されたい．

3. deque

deque においては，後部 (rear) での入力は queue の入力アルゴリズム，前部(front)からの出力は queue の出力アルゴリズムが用いられる*.

文献 1)，EX. 2.2.2-2 (p. 248, p. 535) に，前部での入力，後部からの出力アルゴリズムが以下の(7)，(8)として示されている．

初期値：$R=F=1$

DQ⇐Y (insert at front)

```
DQ[F]←Y;
if F=1 then F←M
else F←F−1;
if F=R then OVERFLOW;
```
(7)

Y⇐**DQ** (delete from rear)

```
if R=F then UNDERFLOW;
Y←DQ[R];
if R=1 then R←M
else R←R−1;
```
(8)

(1)，(2)，(7)，(8)を用いると，M メモリセルをもつ deque 中に $M-1$ 個のデータが格納される．

UNDERFLOW および OVERFLOW なしで，いくつかの入出力後の deque の様子は Fig. 1 と同じである．

queue の場合と同様に，R, F 以外の変数を用いずに，(7)および(8)のアルゴリズムは改良される．

初期値：$R=F=0$

DQ⇐Y (insert at front)

```
F←F+1
if F=R then OVERFLOW;
```

```
        if F≥M+1 then
            if R≤1 then OVERFLOW
            else F←1;
    DQ[F]←Y;
```
(9)

```
Y⇐DQ (delete from rear)
    if R=0 then
            if F=0 then UNDERFLOW
            else R←1;
    Y←DQ[R];
    if R=F then (F←0, R←F)
    else
        if R=M then R←1)
        else R←R+1;
```
(10)

deque に関するアルゴリズ(9)ムに対応して，後部での入力アルゴリズム(5)における

```
        if R≤0 then R←M;
```

は次の様に変更されなければならない．

```
    if R≤0 then
            if F=M then OVERFLOW
            else R←M;
```

変更された(5)を(5)′とする．

(5)′. (6), (9), (10)を用いることによって，M メモリセルをもつ deque 中に M 個のデータが格納される．

4. アルゴリズムの検討

ここでは，(5), (6)における変数 R, F の動作について説明を加える．

(1), (2)では，Q[F] が次に delete されるべきデータが格納されているメモリ・セルでないこと，初期値の(5), (6)との違いにまず注意されたい．

(5), (6)による入出力は各々の変数で見ると，どちらも $M, M-1, M-2, \cdots, 1, M, M-1, \cdots$, の順でなされている．説明の便宜上，これを ($F, R$ の) **順回性** と言うことにする．(5), (6)の各操作終了後は，$F \neq 0, R=0$ または $F=M$ となることはないことに注意されたい．

OVERFLOW, UNDERFLOW なしで，初期値の状態 ($F=R=0$) から幾つかの(5), (6)の操作後の様子を検討する．

すなわち，変数の順回性により，

(a) $n \cdot M-R+1$, ($n \geq 1$) 回の(5)の操作と，

(b) $m \cdot M-F$, ($m \geq 1$) 回の(6)の操作後の様子について考察する．但し n, m は整数で，$0 \leq n-m \leq 1$ である．そうでなければ，OVERFLOW または UNDERFLOW がすでに生じている．

4.1 $n=m$;

このとき格納されているデータ個数は (a)-(b) より $F-R+1$ で Fig. 2 の $F>R$ の状態にある．ここで(6)をさらに $F-R$ 回行なうことによって，$F=R$ となり，ただ1個のデータが queue 中に存在することになる．さらに(6)をつづけると，二度目の(6)のはじめの部分で UNDERFLOW が検出される．

逆に，$M-(F-R+1)$ 個の空きセルに対して(5)を順次行うことによって，総数 M 個のデータが格納される．途中，R の順回性により Fig. 2 の ($F<R$) の様子に変る．このとき n は $n+1$ となる．この場合，データの個数は，

$$(n+1) \cdot M-R+1-(n \cdot M-F)=M+F-R+1$$

である．$R=F-1$ のとき個数は M となる．さらにもう1個入力しようとすると(5)のはじめの部分で $R=F$ より，OVERFLOW が検出される．

4.2 $n=m+1$;

このとき格納されているデータ個数は (a)-(b) より，$M+F-R+1$. Fig. 2 に $<R$ の状態にある．$(R-F-1)$ 個の空きセルに対して(5)を順次行うことによって，M 個のデータが格納される．この場合 $R=F+1$ で，その後の OVERFLOW 検出は **4.1** と同様．一方，(6)を順次続けることによって，F の順回性より，$m=n$ となり，Fig. 2 の ($F>R$) に変る．さらに(6)を続ければ，UNDERFLOW 検出がなされることは **4.1** と同様である．

以上をみると **4.1** の状態で UNDERFLOW, **4.2** の状態で OVERFLOW が検出されることがわかる．読者は，その他実際に $F=0, R \neq 0$ の場合のアルゴリズムの動き等を確められたい．

UNDERFLOW および OVERFLOW なしで，いくつかの入出力後の deque の様子は Fig. 2 と同じである．

参考文献

1) Knuth, D.E.: Fundamental Algorithms, Addison-Wesley, 2-nd edition, 1973

(昭和52年5月25日受付)
(昭和52年8月26日再受付)