## บทที่ 5 แถวคอย (Queues)





IT 204 99

# บทที่ 5
# แถวคอย
# (Queues)

## 5.1 บทนิยาม (definitions)

แถวคอย หมายถึง โครงสร้างข้อมูลแบบรายการเชิงเส้นชนิดพิเศษ

$$A = [ A_1 , A_2 , \ldots , A_T]$$

ในแถวคอย การใส่ถูกจำกัดเฉพาะปลายด้านหนึ่งของรายการซึ่งเรียกว่า ตัวหลัง (rear) การลบออกเกิดขึ้นที่ปลายอีกด้านหนึ่งของรายการเรียกว่าตัวหน้า (front)
สัญลักษณ์ Front(Q) คือตัวหน้าของแถวคอยชื่อ Q และ Rear(Q) หมายถึงตัวสุดท้ายของแถวคอยชื่อQ

สำหรับแถวคอย Q เมื่อ

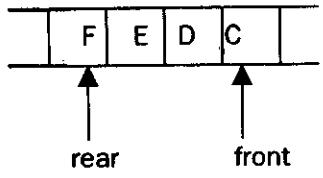
$$Q = [Q_1, Q_2, \ldots ,Q_T]$$

ในที่นี้ Front(Q) คือ $Q_1$ และ Rear(Q) คือ $Q_T$ Noel (Q) แทนจำนวนสมาชิกในแถวคอย Q Noel(Q) คือคุณสมบัติของแถวคอย Q และมีค่าเป็นจำนวนเต็ม จากตัวอย่างข้างต้น

$$Noel(Q) = T$$

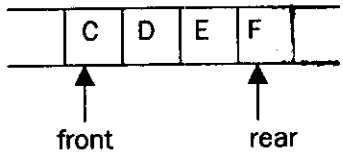
สมาชิกตัวที่ noel หมายถึงสมาชิกตัวหลัง (rear) ของแถวคอยซึ่งเป็นสมาชิกที่เข้ามาอยู่ในแถวคอยในเวลาน้อยที่สุดส่วนสมาชิกตัวแรกของแถวคอยหมายถึง สมาชิกตัวหน้า (front) ของแถวคอย เป็นสมาชิกที่เข้ามาอยู่ในแถวคอยนานที่สุด

การวาดรูปแถวคอยโดยให้เพิ่มจำนวนสมาชิกจากขวาไปซ้ายเป็นดังนี้




รูป 5-1(a) Drawing of a queue

หรือจากซ้ายไปขวาเป็นดังนี้




รูป 5-1(b)

หรือวาดรูปในแนวตั้ง ตราบใดที่มีความพ้องกันในการเก็บ track ของตัวหน้าและตัวหลังของแถวคอย

ในกรณีใด ๆ ก็ตาม สำหรับแถวคอย

$$Q = [ Q_1 , Q_2 , \ldots , Q_{noel}]$$

เราพูดว่าสมาชิก $Q_i$ อยู่ก่อนหน้า สมาชิก $Q_j$ สำหรับ $i < j$ ดังนั้น $Q_i$ จะถูกลบออกจากแถวคอยก่อนสมาชิกใด ๆ ซึ่งอยู่ข้างหลังมัน

$Q_i$ อยู่ในแถวคอยนานกว่าสมาชิกใด ๆซึ่งอยู่ข้างหลังมัน จะเห็นชัดเจนจากตัวอย่างลูกค้าธนาคารที่ยืนในแถวหน้าตู้ ATM

### 5.1.2 การปฏิบัติการบนแถวคอย (Queue operations)

การปฏิบัติการพื้นฐานซึ่งถูกต้องสำหรับข้อมูลชนิดแถวคอยมีสี่ชนิดคือ

1. Create (queue)
2. Isempty (queue)
3. Insert (element, queue)
4. Remove (queue)

Create (Q) returns an empty queue with name Q. By definition

Noel (Create (Q)) is 0
Front (Create(Q)) is undefined
Rear (Create(Q)) is undefined

ตัวดำเนินการ Isempty ระบุว่าแถวคอยว่างหรือไม่ ตัวถูกดำเนินการคือแถวคอย ผลลัพธ์เป็น boolean

Isempyt (Q) เป็น true ถ้าแถวคอย Q ว่าง (นั่นคือ Noel(Q) = 0) และเป็น false กรณีอื่น ๆ โปรดสังเกตว่า Isempty (Create(Q)) เป็น true

Insert (E,Q) หมายถึงตัวดำเนินการซึ่งใส่สมาชิก E เข้าไปในแถวคอย Q โดยบทนิยาม E จะถูกใส่ที่ตัวหลังของแถวคอย ผลลัพธ์ของการ ดำเนินการแถวคอยจะมีขนาดเพิ่มขึ้น

Rear (Insert (E,Q)) คือ E
Noel (Q) เพิ่มขึ้นโดยการปฏิบัติการ insert และ $Q_{noel}$ คือ E

ผลลัพธ์ของการใส่สมาชิกเข้าไปในแถวคอยทำให้ไม่เป็นแถวคอยว่าง

Isempty (Insert (E,Q) ) คือ false

มีเพียงหนึ่งเงื่อนไขเท่านั้นที่ทำให้การใส่สมาชิกเข้าไปในแถวคอยมีผลกระทบกับสมาชิกตัวหน้า (front) ของแถวคอย

```
If  Isempty (Q)
then Front (Insert (E,Q)) is E
else Front (Insert (E,Q)) is Front(Q)
```

Remove (Q) ลบสมาชิกตัวหน้าออกจากแถวคอย Q ผลลัพธ์คือ แถวคอยมีขนาดลดลง ถ้าสมาชิกตัวหน้าควรจะเก็บไว้ การกระทำต้องกระทำก่อนการปฏิบัติการลบออก

Noel (Q) มีขนาดลดลงโดยปฏิบัติการ Remove และสมาชิกตัวที่สองของ Q ก่อนหน้านี้จะกลายเป็นสมาชิกตัวหน้าตัวใหม่ข้อผิดพลาดจะเกิดขึ้นถ้าพยายามลบสมาชิกออกจากแถวคอยว่าง

```
If      Noel(Q) = 0
then    Remove (Q) yields an error condition
```

โปรดสังเกตว่า

```
Remove (Create(Q)) also yields an error condition
```

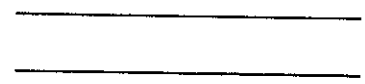
มีเพียงกรณีเดียวเท่านั้นซึ่งการลบสมาชิกออกจากแถวคอยลบล้าง (undo) การใส่สมาชิกก่อนหน้านั้นเข้าไปไว้ในแถวคอย

```
If      Isempty (Q)
then    Remove (Insert(E,Q)) is Q
else    Remove (Insert (E,Q))
                = Insert (E, Remove(Q))
```

ถ้า Qไม่ใช่แถวคอยว่าง (is not empty) , ตัวดำเนินการใส่และตัวดำเนินการลบกระทำเป็นอิสระจากกันและไม่มีผลกระทบซึ่งกันและกัน

### 5.1.2 ตัวอย่าง (Example)

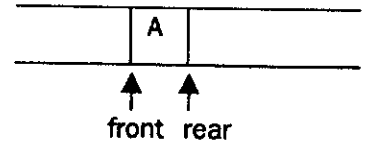
สำหรับตัวอย่างผลกระทบของลำดับของตัวดำเนินการแถวคอย ให้เริ่มต้นด้วยแถวคอยว่างชื่อ Q เพราะฉะนั้น Q = [ ] แทนที่ด้วยรูปข้างล่างนี้



Noel(Q) = 0
Front (Q) undefined
Rear(Q) undefined

รูป 5-2(a)

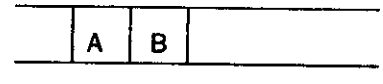
ขั้นแรก ใส่สมาชิก A ทำให้ Q = [ A ]




Noel(Q) = 1
Front(Q) = A
Rear (Q) = A

รูป 5-2(b)

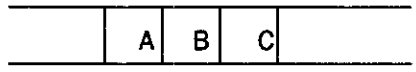
หลังจากนั้น ใส่สมาชิก B ทำให้ Q = [ A, B ]




Noel(Q) = 2
Front(Q) = A
Rear(Q) = B

รูป 5-2( C)

หลังจากนั้นใส่สมาชิก C ทำให้ Q = [ A , B , C ]



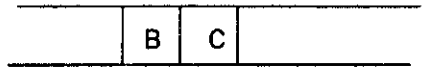

Noel (Q) = 3

Front (Q) = A

Rear (Q) = B

รูป 5-2(d)

หลังจากนั้น ลบสมาชิกออกจาก Q ทำให้ Q = [ B , C ]



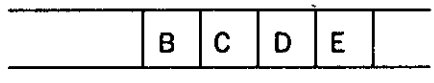

Noel(Q) = 2

Front(Q) = B

Rear(Q) = C

รูป 5-2(e)

หลังจากนั้นใส่สมาชิกเพิ่มอีกสองตัวคือ D และ E ทำให้ Q = [ B, C, D,E ]



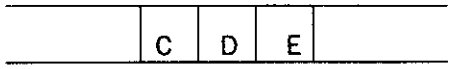

Noel(Q) = 4

Front(Q) = B

Rear(Q) = E

รูป 5-2 (f)

หลังจากนั้นลบสมาชิกหนึ่งตัวออกจากแถวคอย ทำให้ Q [ C, D,E ]




Noel(Q) = 3

Front(Q) = C

Rear(Q) = E

รูป 5-2(g)

เช่นนี้เรื่อยไป แถวคอย กระทำในลักษณะมาก่อนออกก่อน (first-in-first-out (FIFO) manner)สมาชิกถูกลบออกในอันดับซึ่งมันถูกใส่เข้ามาในแถวคอย
ให้เปรียบเทียบกิจกรรมของแถวคอยตัวอย่างนี้กับกองซ้อนตัวอย่างในบทที่แล้ว ลำดับของปฏิบัติการเหมือนกันเกิดขึ้นในทั้งสองกรณีแต่เนื้อหา(contents) ของรายการแตกต่างกันมาก

## 5.2 แถวคอยใน COBOL และ Pascal (Queues in COBOL and Pascal)

### 5.2.1 เก็บแถวคอยในแถวลำดับ (Housing Queues in Arrays)

ภาษาโปรแกรมส่วนใหญ่ไม่มีโครงสร้างข้อมูลกองซ้อนในตัว และไม่มีข้อมูลชนิดแถวคอยในตัว การเขียนโปรแกรมเพื่อแก้ปัญหาซึ่งเรียกใช้แถวคอยโปรแกรมเมอร์ต้องใช้คุณสมบัติที่มีอยู่ของภาษานั้นเพื่อจำลองแบบการปฏิบัติการของแถวคอย
วิธีที่ง่ายที่สุดอย่างหนึ่งเพื่อแทนที่แถวคอยคือเก็บแถวคอยในแถวลำดับ นอกจากความยากที่โปรแกรมเมอร์ต้องรับผิดชอบความบูรณภาพของแถวคอยแล้วโปรแกรมเมอร์ต้องเชื่อมั่นว่าแถวคอยกระทำบนพื้นฐานของ FIFO ด้วย

### 5.2.2 การประกาศแถวคอย (Declaring Queues)

การประกาศแถวคอยใช้ตัวแปรชื่อ Q สมมติว่าสมาชิกแต่ละตัวของ Q เป็นจำนวนเต็ม Q มีสมาชิกมากที่สุด 100 ตัว นอกจากการประกาศแถวลำดับซึ่งจะเก็บ Q เราต้องประกาศตัวแปรซึ่งชี้สมาชิกตัวหน้าและชี้สมาชิกตัวหลังของแถวคอยในที่นี้ FRONT และ REAR จะเป็นตัวแปรจำนวนเต็มซึ่งค่าของมันจะเป็นดรรชนีล่างของสมาชิกตัวแรกและสมาชิกตัวสุดท้ายของแถวคอย ตามลำดับ

เราให้ชื่อรวมของแถวลำดับและตัวชี้ front และ rear ว่า QUEUE-STRUCT ใน COBOL :

```
01    QUEUE-STRUCT.
      02    Q       OCCURS 100 TIMES   PICTURE   9(5).
      02    FRONT           PICTURE   9(5).
      02    REAR            PICTURE   9(5).
```

ใน Pascal

```
type queuestruct =
        record queue : array [1..100] of integer
                front,rear : integer
        end;
var  q : queuestruct;
```

### 5.2.3 การปฏิบัติการบนแถวคอย (Queue operations)

เริ่มต้น สำหรับแถวคอยว่างให้ FRONT และ REAR เท่ากับ 0 ให้ตัวแปร NOEL-MAX คือจำนวนมากที่สุดของสมาชิกซึ่งแถวลำดับของแถวคอยสามารถเก็บได้ ในที่นี้ NOEL-MAX=100

สิ่งสำคัญคือการตรวจสอบว่า FRONT และ REAR ไม่ได้ชี้นอกขอบเขตแถวลำดับของแถวคอยเมื่อมีความพยายามใส่สมาชิกเข้าไปในแถวคอยเต็มเป็นเงื่อนไขภาระส่วนล้น (overflow condition) ความพยายามลบสมาชิกออกจากแถวคอยว่างเกิดเงื่อนไขน้อยเกินเก็บ (underflow condition) โปรดสังเกตว่าการใส่สมาชิกเข้าไปในแถวคอยว่างต้องเคลื่อนย้ายทั้ง พอยน์เตอร์ FRONT และ REAR การลบสมาชิกออกจากแถวคอยถ้าทำให้แถวคอยว่างตัวชี้ FRONT และ REAR ต้อง set ใหม่ให้เป็น 0 เพื่อให้การใส่เข้าไปในแถวคอยหลังจากนั้นประมาลผลจากส่วนล่างสุดของแถวลำดับ
EON คือสมาชิกตัวที่จะใส่บนแถวคอย และ EOFF คือสมาชิกตัวที่ลบออกจากแถวคอย โปรดสังเกตว่าเราเก็บ EOFF และสมมติว่า Q เป็นตัวแปรส่วนกลางในตัวอย่าง Pascal

ใน COBOL:

```
INSERT.
        IF REAR = NOEL-MAX
                overflow condition
        ELSE COMPUTE REAR = REAR +1
                MOVE EON TO Q(REAR)
                IF FRONT = 0
                        COMPUTE FRONT = 1.

REMOVE.
        IF FRONT > 0
                MOVE Q(FRONT) TO EOFF
                IF FRONT = REAR
                        COMPUTE  FRONT = 0
                        COMPUTE  REAR  = 0
                ELSE COMPUTE FRONT = FRONT +1
        ELSE underflow-condition.
```

ใน Pascal :

```
procedure insert (ion: integer);
  begin
     if (q.rear < noelmax)
     then begin
             q.rear := q.rear + 1;
             q.queue [q.rear] := eon;
             if (q.front = 0)
             then q.front :=1
          end
     else  OVERFLOW-CONDITION
  end;
```

```
procedure remove (var eoff : integer);
  begin
      if(q.front>0)
      then begin
            eoff := q.queue[q.front];
            if (q.front=q.rear)
            then begin
                  q.front := 0
                  q.rear  := 0
            else q.front := q.front +1
          end
      else UNDERFLOW-CONDITION
end;
```

จำนวนสมาชิกในแถวคอย ณ.เวลากำหนดให้ใด ๆ ก็ตามคำนวณได้จากค่าของพอยน์เตอร์ front และ rear ดังนี้

```
if FRONT =0
then Noel (Q) is 0
else noel(Q) is FRONT-REAR +1
```

### 5.2.4 การเคลื่อนย้ายผ่านหน่วยเก็บ (Moving through storage)

ขณะนี้จะตรวจสอบอัลกอริทึมเหล่านี้อย่างระมัดระวังเพิ่มขึ้นและใช้แถวลำดับเพื่อเก็บแถวคอยจงพิจารณาการใส่และการลบสมาชิกต่างๆอย่าง สืบเนื่อง

**ตัวอย่าง** ใส่สมาชิกสี่ตัว


1 2 3 4 5 6 7 8 . . .

FRONT = 1

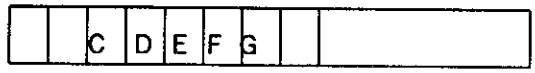
REAR = 4 รูป 5-3(a)

ลบสมาชิกออกสองตัว


1 2 3 4 5 6 7 8 . . .

FRONT = 3
REAR = 4 รูป 5-3(b)

ใส่สมาชิกเพิ่มอีกสองตัว




1 2 3 4 5 6 7 8 . . .

FRONT = 3
REAR = 7 รูป 5-3(c)

ลบสมาชิกออกสองตัว


1 2 3 4 5 6 7 8 . . .

FRONT = 5
REAR = 7 รูป 5-3(c)

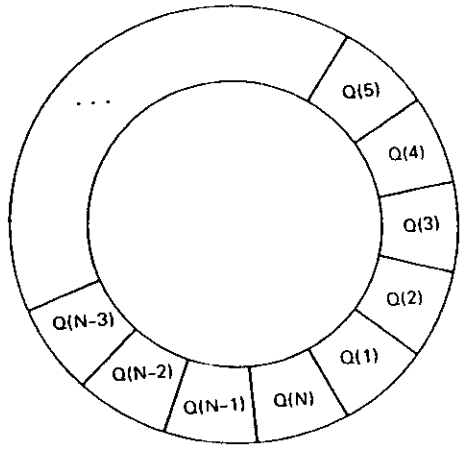
เช่นนี้เรื่อยไป แถวคอยเดินทางจากซ้ายไปขวาผ่านแถวลำดับ เกิดอะไรขึ้นเมื่อ REAR = 100 และเราจำเป็นต้องใส่สมาชิกเพิ่มในแถวคอย

แถวคอยเต็ม ถ้า FRONT =1 และถึงตำแหน่งมากที่สุด เช่น Noel (Q) = 100 อย่างไรก็ตาม เป็นไปได้ที่พอยน์เตอร์ front ของแถวคอยจะเคลื่อนไปทางขวาเนื่องจากการลบสมาชิกหนึ่งตัวหรือมากกว่าหนึ่งตัว และ FRONT > 1 เป็นไปได้ที่ FRONT = 100 และมีสมาชิกเพียงหนึ่งตัวเท่านั้นใน Q แต่ดูเหมือนว่า Q เต็ม

วิธีหนึ่งในการแก้ปัญหานี้คือเก็บแถวคอยในแถวลำดับที่มีขนาดใหญ่ขึ้น การแก้ปัญหาวิธนี้ไม่น่าประทับใจเว้นเสียแต่ว่าเรามีจำนวนหน่วยเก็บมากอย่างไม่จำกัดที่จะยกให้แถวลำดับนั้นหรือสามารถนำข้อดีของความรู้ก่อนหน้าของพฤติกรรมของแถวคอยมาใช้

## 5.3 แถวคอยแบบวงกลม (Circular Queues)

วิธีที่ดีกว่าคือจัดการ Q ให้เป็นวงกลม ให้ Q(1) อยู่หลัง Q(NOEL – MAX) เช่นในรูป 5-4 จากนี้เป็นต้นไปจะแทนที่ตัวแปร NOEL – MAX ด้วยตัวแปร N เพื่อทำให้วาด diagram ง่ายขึ้น

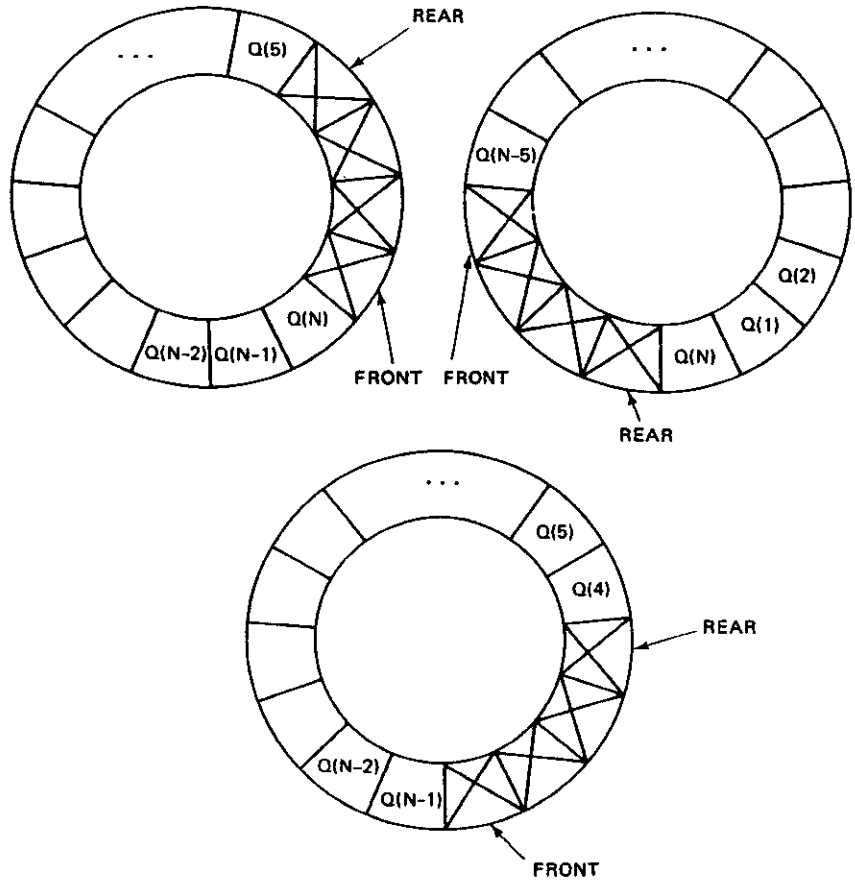



รูป 5-4 Circular queue

ขณะนี้ขึ้นอยู่กับว่าตัวดำเนินการใส่และลบออกจะทำให้เกิดผลในทางปฏิบัติอย่างไร แถวคอยจะใช้แถวลำดับแบบวงกลมแตกต่างกัน ในที่นี้จะนำเสนอสองทางเลือกส่วนทางเลือกอื่น ๆ เป็นแบบฝึกหัดท้ายบท

**ทางเลือกที่หนึ่ง** ตามข้อตกลงของเราให้ตัวแปร FRONT ชี้ที่สมาชิกตัวแรกของแถวคอย และตัวแปร REAR ชี้ที่สมาชิกตัวสุดท้ายของแถวคอย

รูปแบบร่างของแถวคอยชนิดต่าง ๆ ณ.หลายจุดในเวลาที่กำหนดให้อยู่ใน รูป 5 – 5



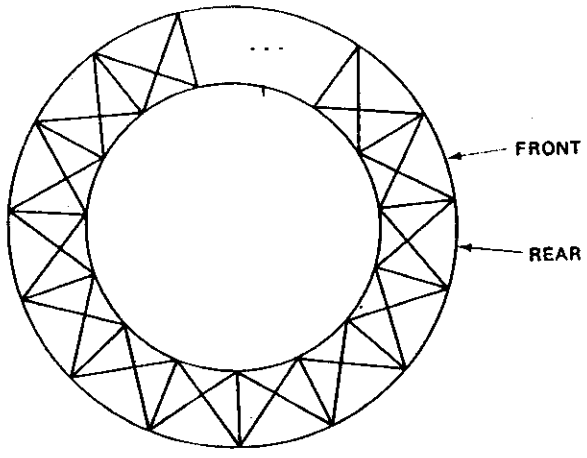

รูป 5-5 snapshots of circular queue with Front and rear pointing to the first and last REAR elements, respectively

เมื่อ x ทำเครื่องหมายช่องที่มีสมาชิกของแถวคอย โปรดสังเกตว่าสมาชิกของแถวคอยระยะกว้างของขอบเขตระหว่าง Q(1) และ Q(N) ตัวดำเนินการใส่และลบออก ทำให้ต้องกำหนดพอยน์เตอร์ FRONT และ REAR ใหม่เมื่อมันข้ามขอบเขตจากค่า N ไป 1

สำหรับแถวคอยว่าง (empty queue) เงื่อนไขคือ

FRONT = REAR = 0

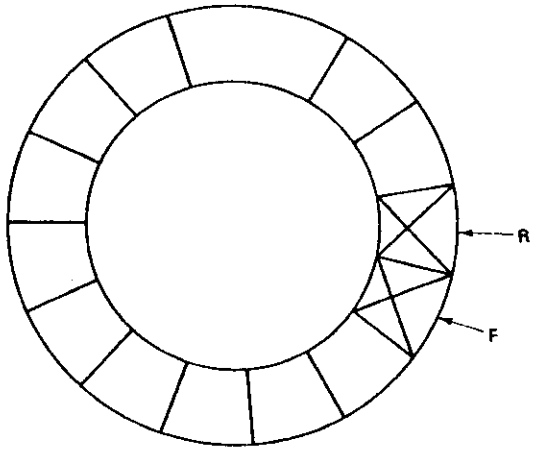
แถวคอยเต็ม (Full queue) คือ รูป 5-6 โปรดสังเกตว่าแถวคอยนี้เพิ่มจำนวน(grows) ในทิศทางทวนเข็มนาฬิกา ( cornterclockwise direction)



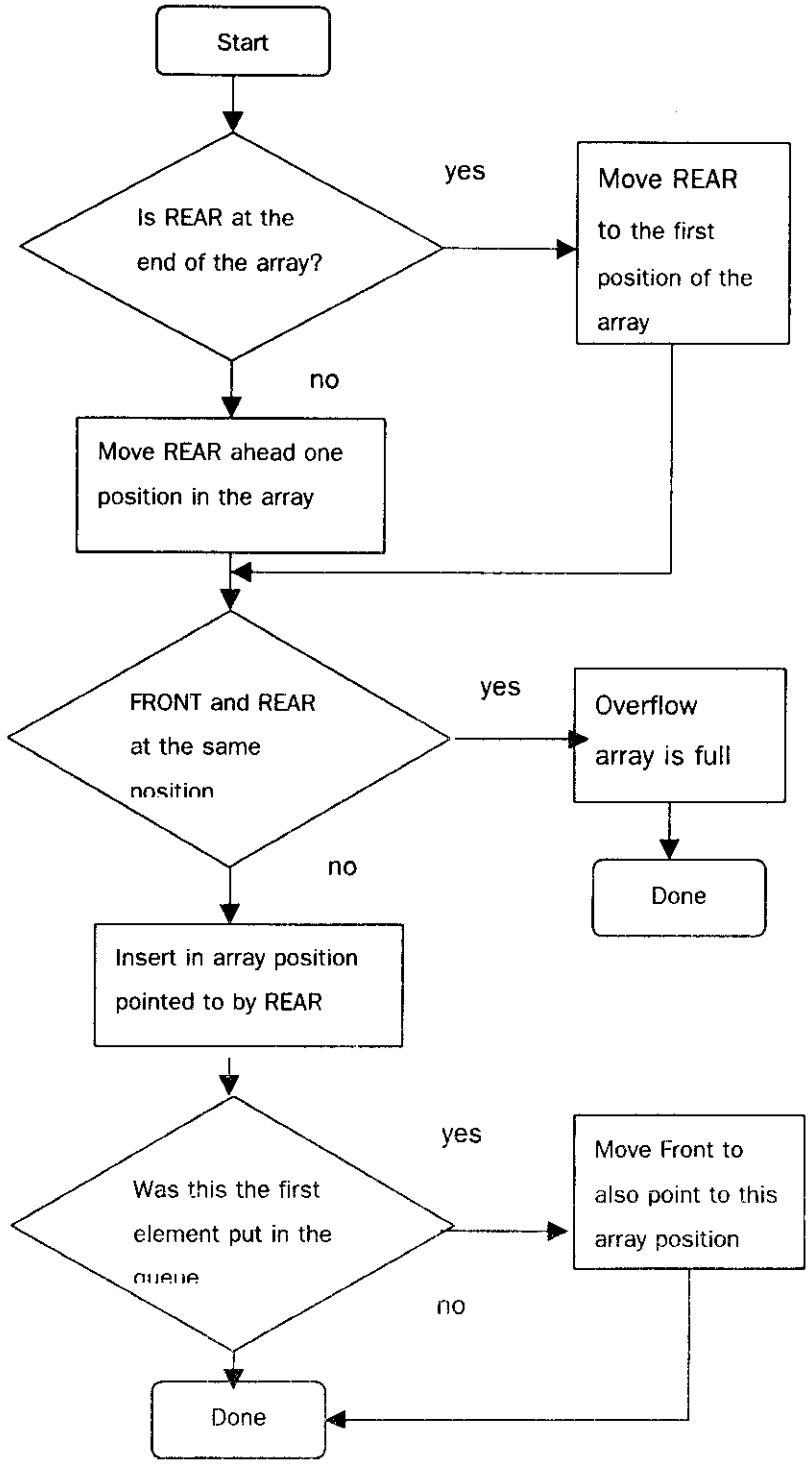

รูป 5-6 Full queue ใช้ข้อตกลงพอยน์เตอร์ ของรูป 5-5

**กิจกรรม**

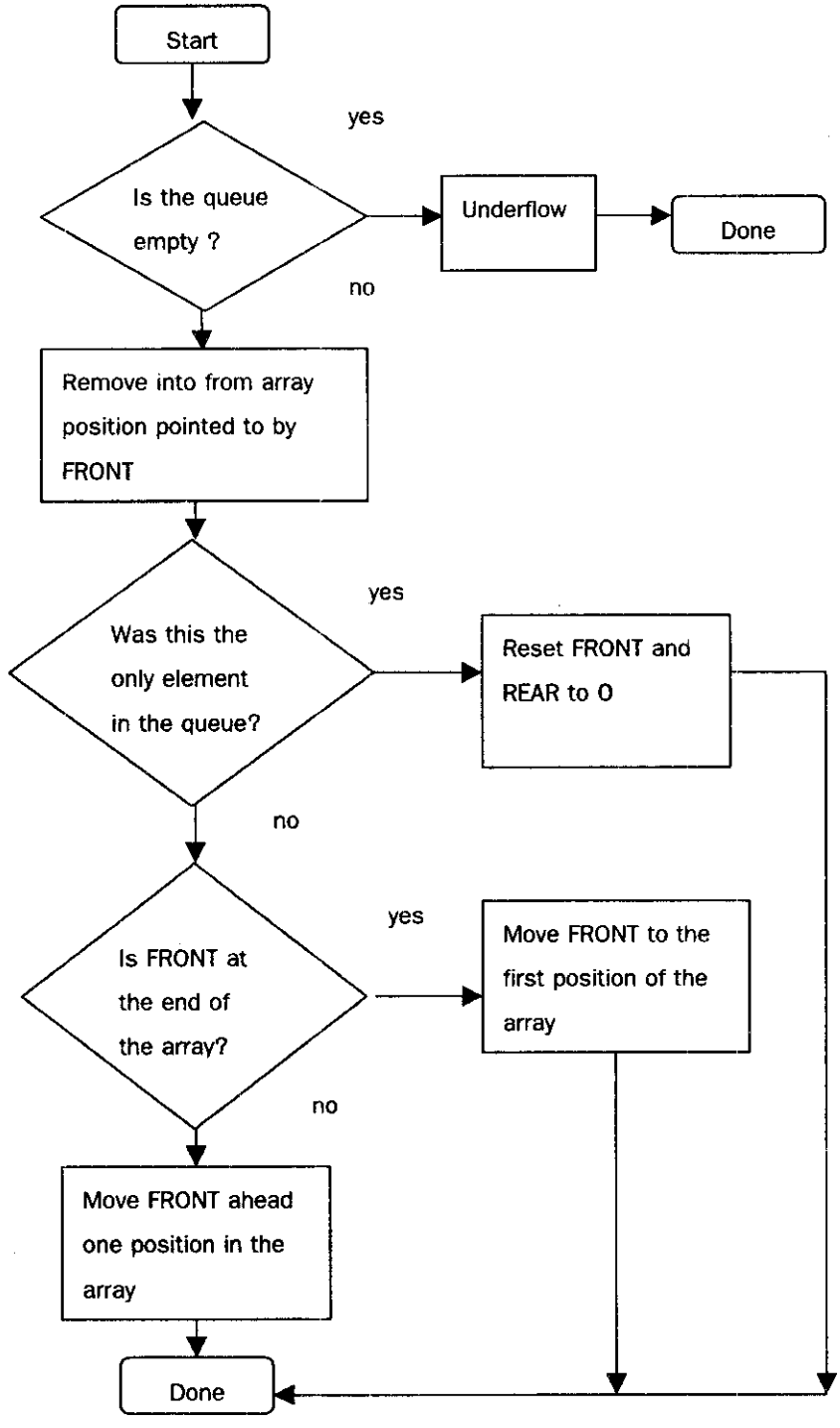
(a) จงวาดรูปแถวคอยที่มีสมาชิกเพียงหนึ่งตัว (b)จงเปรียบเทียบตำแหน่งของพอยน์เตอร์ FRONT และ REAR เมื่อแถวคอยเต็มและเมื่อแถวคอยมีสมาชิกเพียงสองตัว ในรูป 5-7




รูป 5-7 แถวคอยที่มีสมาชิกสองตัวโดยใช้ข้อตกลงพอยน์เตอร์ของรูป 5-5




รูป 5-8 Logic for inserting an element in a circular queue




รูป 5-9 ตรรกะสำหรับลบสมาชิกหนึ่งตัวออกจากแถวคอยแบบวงกลม

### 5.3.1 การใช้แถวลำดับแบบวงกลม (Using Circular Queues)

อัลกอริทึมสำหรับตัวดำเนินการใส่และลบออก สำหรับแถวคอยแบบวงกลม ซึ่งกำหนดให้ในรูป 5-8 และ 5-9 ตามลำดับทำให้เกิดผลในทางปฏิบัติในภาษา COBOL เขียนดังนี้

```
INSERT.
        IF REAR = N
                COMPUTE  REAR = 1
        ELSE COMPUTE REAR = REAR+1;
        IF  REAR = FRONT
                overflow-condition
        ELSE MOVE EON TO Q(REAR)
                IF FRONT = 0
                COMPUTE FRONT = 1

REMOVE.
        IF FRONT = 0
                underflow-condition
        ELSE MOVE Q(FRONT) TO EOFF
                IF FRONT = REAR
                        COMPUTE FRONT = 0
                        COMPUTE REAR  = 0
                ELSE IF FRONT = N
                                COMPUTE FRONT = 1
                        ELSE COMPUTE FRONT = FRONT + 1
```

ใน Pascal

```
procedure insert(eon : integer);
begin
        if (q.rear = n)
        then q.rear :=1
        else q.rear := q.rear + 1;
        if (q.rear = q.front)
        then OVERFLOW – CONDITION
        else begin
                q.queue[q.rear] :=eon;
                if (q.front=0)
                then q.front := 1
                end;
end;

procedure  remove (var eoff : integer);
begin
        if (q.front = 0)
        then UNDERFLOW – CONDITION
        else begin
                eoff := q.queue[q.front];
                if (q.front = q.rear)
                then begin
                        q.front:=0;
                        q.rear:=0;
                      end
                else if (q.front = n)
                     then q.front := 1
                     else q.front := q.front + 1
                end;
end;
```

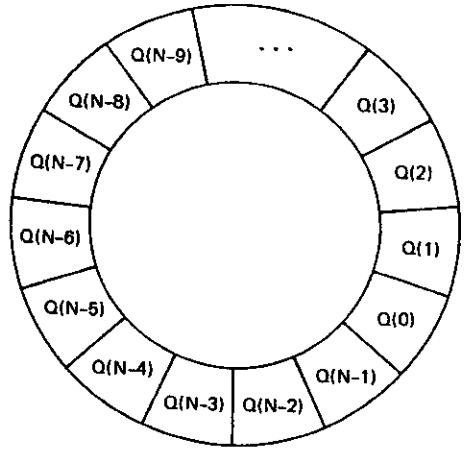
### 5.3.2 ความหลากหลาย (Variations)

ถ้าเราใช้อัลกอริทึมเหมือนเดิมแต่เป็นจำนวนช่องแบบตามเข็มนาฬิกาไม่ใช่ทิศทางทวนเข็มนาฬิกา แถวคอยจะปรากฏ เพื่อเคลื่อนที่ในทิศทางตามเข็มนาฬิกา

ถ้าเรายังคงเก็บโครงร่างการให้เลขช่องว่างแบบทวนเข็มนาฬิกาแต่เปลี่ยนแปลงรหัสการใส่และการลบออก โดยที่ พอยน์เตอร์ REAR และ FRONT มีค่าลดลงไม่ใช่เพิ่มขึ้นแถวคอยจะปรากฏการเคลื่อนที่ในทิศทางตามเข็มนาฬิกา

โปรแกรมเมอร์บางคนชอบให้เลขช่องแถวลำดับจาก 0 ถึง N-1 ไม่ใช่จาก 1 ถึง N (รูป 5-10) ขณะนี้เราสามารถจัดการการผ่านขอบเขต Q(N-1) - Q(0)

โดยใช้ modulo arithmetic ซึ่งเป็น built-in function ใน Pascal, FORTRAN และ PL/1




รูป 5-10 Circular queue space with base at array slot 0

ใน Pascal, A **mod** B หมายถึงเศษที่เป็นจำนวนเต็มซึ่งเป็นผลลัพธ์เมื่อ A เป็นตัวตั้ง และ B เป็นตัวหาร

ดังนั้น I := A **mod** B

จึงมีความหมายเหมือนกับข้อความสั่งของ COBOL ดังนี้

DIVIDE A BY B GIVING C REMAINDER I.

เพราะฉะนั้นการปฏิบัติการใส่และการลบจึงสามารถทำให้เกิดผลในทางปฏิบัติในโปรแกรม Pascal โดยแทนที่รหัส

```
if (q.rear = n)
then q.rear := 1
else q.rear := q.rear + 1
```

ด้วย

```
q.rear := (q.rear + 1) mod n;
```

ในโปรซีเดอร์ insert , ให้แทนที่รหัส

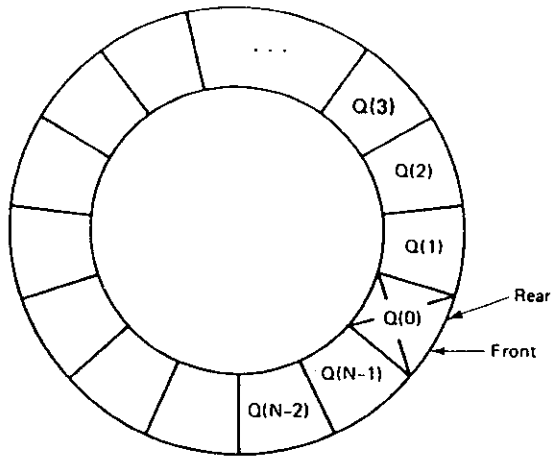
```
if (q.front = n)
then q.front := 1
else q.front := q.front + 1
```

ด้วย

```
q.front := (q.front+1) mod n;
```

ในโปรซีเดอร์ removeที่จริงแล้ว การเปลี่ยนแปลง โครงร่างของการให้หมายเลขช่องว่างดูยากมากขึ้นที่จะตรวจพบแถวคอยว่างขณะนี้ FRONT = REAR = 0 หมายถึงแถวคอยที่มีสมาชิกหนึ่งตัว อยู่ในช่องว่าง 0 (รูป 5-11) ในตอนต้นนั้นเมื่อแถวลำดับของแถวคอยอยู่ที่ ตำแหน่ง 1 แทนที่จะเป็น 0

FRONT = REAR = 0 หมายถึงแถวคอยว่าง

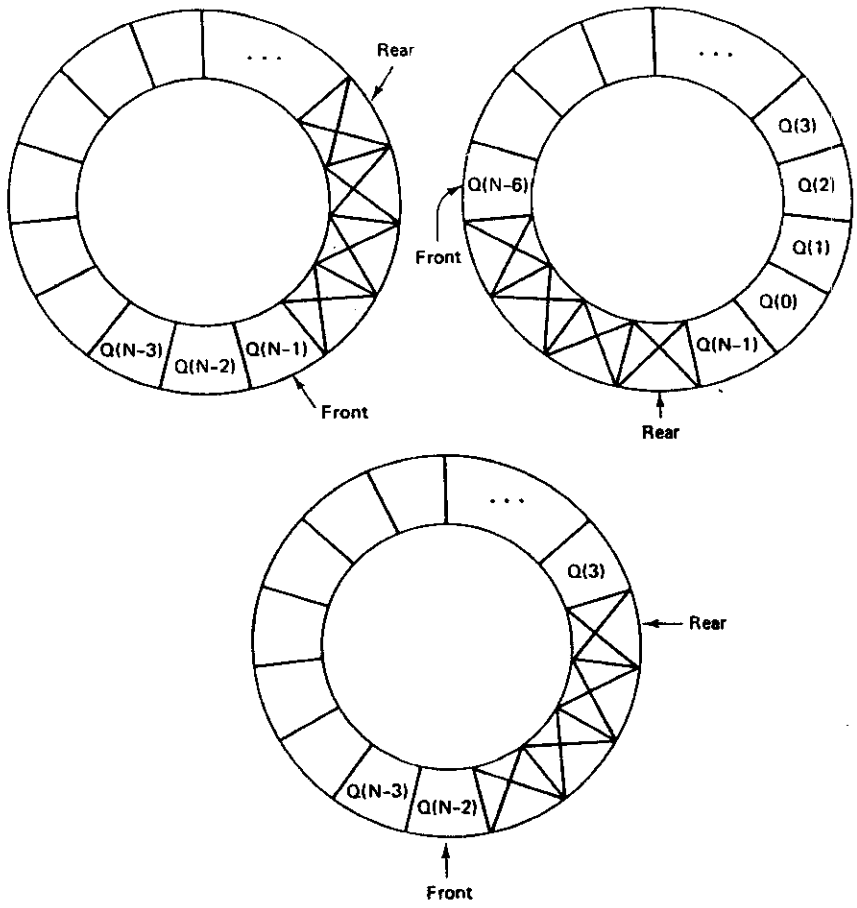



รูป 5-11 Circular queue space of Fig 5-10 with FRONT = REAR = 0

### 5.3.3 การแทนที่อีกหนึ่งทางเลือก(An Alternative Representation)

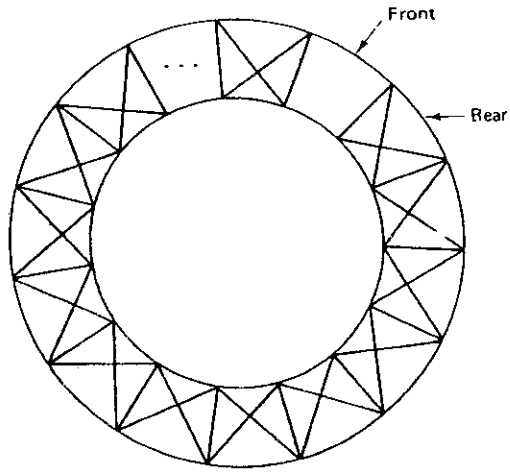
เพื่อแบ่งเบาความยากนี้ เทคนิควิธีที่สองสำหรับการแทนที่แถวคอยในแถวลำดับแบบวงกลมจึงถูกนำมาใช้ร่วมกัน วิธีนี้แทนที่พอยน์เตอร์ FRONT จะชี้ที่สมาชิกตัวหน้าของแถวคอย กลับให้ FRONT ชี้ไปยังช่องว่างหนึ่งช่องหน้าสมาชิกตัวหน้าของแถวคอยเพราะฉะนั้นแถวคอยจึงว่างเมื่อ FRONT = REAR และแถวคอยเต็มเมื่อ

REAR = (FRONT + 1) **mod** N ให้ดูรูป 5-13




รูป 5-12 Snapshots of circular queue with FRONT pointing to the slot preceding the first element and REAR pointing to the last element

จำนวนสมาชิกมากที่สุดซึ่งสามารถเก็บในแถวคอยนี้คือ N-1 ตัว ถึงแม้ว่าจะมีช่องอยู่ N ช่องก็ตาม




รูป 5-13 Full queue , using pointer convention of Fig. 5-12

ถ้าเรายอมให้สมาชิก N ตัวเข้าไปในแถวคอยเพราะฉะนั้นเงื่อนไขเต็ม FRONT = REAR จะเหมือนกับเงื่อนไขว่างสิ่งที่ขัดแย้งกันเองนี้จัดการได้โดยตรงคือ ถ้า FRONT = REAR เป็นผลลัพธ์จาการใส่สมาชิก (แถวคอยเต็ม) หรือลบสมาชิก(แถวคอยว่าง) หรือวิธีง่าย ๆ คือการเก็บค่าของ Noel (Q) ไว้

การปฏิบัติการใส่และลบของ Pascal สำหรับแถวคอยแบบวงกลมที่มีดรรชนี 0 ถึง n-1 และพอยน์เตอร์ front ชี้ที่ก่อนหน้าสมาชิกตัวแรก เขียนดังนี้

```
procedure insert (eon : integer);
begin
        if (q.front <> (q.rear + 1) mod n);
        then begin
                q.rear := (q.rear + 1) mod n
                q.quere [q.rear] := eon
             end
        else OVERFLOW
end;
```

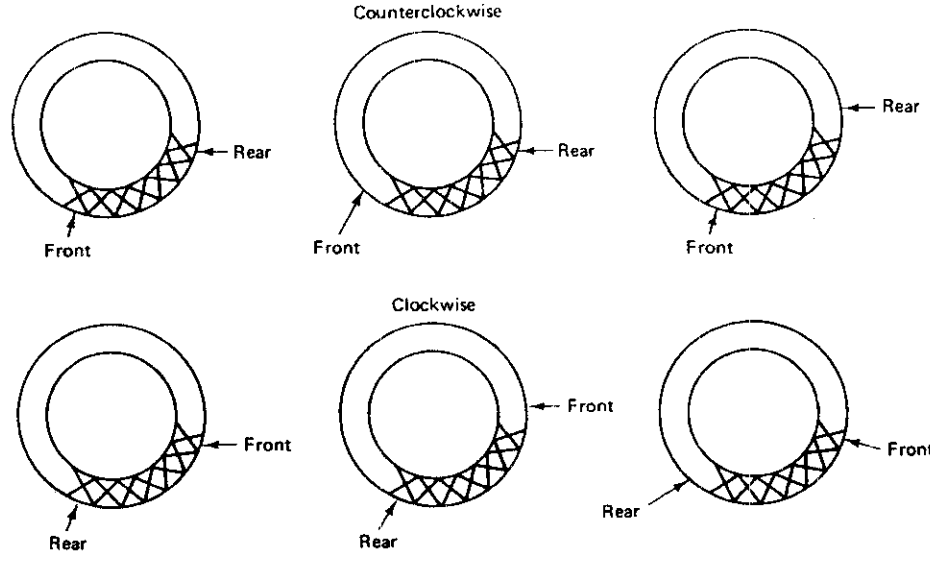
```
procedure removeq (var eoff : integer);
    begin
        if (q.rear = q.front)
        then UNDERFLOW-CONDITION
        else  begin
                q.front := (q.front + 1) mod n;
                 eoff    :=  q.queue [q.front]
              end;
    end;
```

**กิจกรรม**

ให้นักศึกษาเปรียบเทียบอัลกอริทึมเหล่านี้กับอัลกอริทึมซึ่งนำแสนอก่อนหน้านี้เมื่อพอยน์เตอร์ frontชี้ที่สมาชิกตัวแรกของแถวคอย

การแทนที่อีกหนึ่งทางเลือก คือให้พอยน์เตอร์ REAR ชี้ที่ช่องหลัง rear ของแถวคอยหนึ่งตัว ส่วนพอยน์เตอร์ FRONT ชี้ที่สมาชิกตัวแรก

ให้นักศึกษาสร้างอัลกอริทึมการใส่และการลบสมาชิกสำหรับวิธีนี้




รูป 5-14 Alternative conventions for representing a queue in a circular array

## 5.4 การปฏิบัติงานของแถวคอย (Performance of Queues)

แถวคอยนำมาใช้อย่างกว้างขวางในงานประยุกต์ซึ่งจำลองแบบ (simulate) พฤติกรรมของระบบจริงซึ่งเป็นกิจกรรมเข้าก่อนออกก่อน (FIFO activity) แถวคอยสร้างขึ้นในสถานะการณ์มากมาย : รถยนต์เข้าแถวคอยที่สัญญาณไฟจราจร,ผู้คนเข้าแถวคอยที่สำนักงานไปรษณีย์ , ลูกค้าเข้าแถวคอยที่เคาน์เตอร์รับชำระเงินในร้านขายของชำ,ข้อความเข้าแถวคอยในเครือข่ายการสื่อสาร, แฟ้มถูกเก็บพักเพื่อส่งมอบในอันดับลักษณะ FIFO เพื่อไปยัง line printer เป็นต้น

### 5.4.1 พารามิเตอร์ปฏิบัติงาน (Performance Parameters)

นักออกแบบบ่อยครั้งสนใจที่จะปรับปรุงการปฏิบัติงานของระบบเช่นนี้ เมื่อการปฏิบัติงานสามารถวัดได้ด้วยพารามิเตอร์ดังต่อไปนี้ :

. จำนวนเฉลี่ยของสมาชิกในแถวคอย
. เวลาเฉลี่ยที่ใช้ในแถวคอย
. ความน่าจะเป็นซึ่งความยาวของแถวคอยจะมากกว่าเลขบางตัว
. ความยาวน้อยที่สุดและมากที่สุดผ่านช่วงเวลาหนึ่ง
. เวลามากที่สุดและน้อยที่สุดในแถวคอยผ่านช่วงเวลาหนึ่ง

มีวิธีพื้นฐานสามวิธีที่จะคำนวณค่าของพารามิเตอร์ข้างต้น ได้แก่ การสังเกต การจำลอง และทฤษฎีแถวคอย

**การสังเกตุ (Observation)**

การสังเกตสามารถนำมาประยุกต์เพื่อศึกษาระบบที่มีอยู่ ตัวอย่างเช่น เราอาจเห็น electronic counters ที่สัญญาณไฟจราจรสังเกตรถยนต์ในแถวคอยและจำนวนรถยนต์ที่ผ่านสี่แยกไปแล้ว

การสังเกตไม่ใช่วิธีที่เหมาะสมเพื่อศึกษาระบบซึ่งยังคงอยู่ในขั้นตอนออกแบบหรือศึกษาผลกระทบของการเปลี่ยนแปลง ข้อเสนอให้กับระบบที่มีอยู่

**การจำลอง (Simulation)**

เครื่องมือที่ดีมาก (powerful tool) ซึ่งสามารถนำมาใช้เพื่อศึกษาพฤติกรรมของระบบคือการจำลอง การจำลองฝึกหัดใช้ตัวแบบของระบบภายใต้การศึกษา
(A simulation exercises a model of the system under study.)

ตัวอย่างเช่น เครื่องจำลอง (simulator) ของร้านขายของชำระบบ checkout ซึ่งสร้างการเข้ามายัง checkout queue จะจำแนกผู้เข้ามาแต่ละคนโดยจำนวนบริการที่ต้องการ ซึ่งจะแทนอัตราของการคืนบริการที่แต่ละคนยืนอยู่ เป็นต้น เครื่องจำลองจะรวบรวมสถิติระหว่างการปฏิบัติการของตัวแบบผ่านบางช่วงเวลา ตัวสถิติเหล่านี้จะบอกคุณสมบัติ พฤติกรรมของแถวคอยซึ่งพัฒนาขึ้น

ในการสร้างตัวแบบจำลอง (simulation model)จำเป็นต้องมีสถิติให้ใช้มากมาย เพื่อบอกลักษณะระบบภายใต้การศึกษา สถิติเหล่านี้ได้แก่ arrival distribution ค่าเฉลี่ย และ ค่าแตกต่างมาตรฐานของ service time เป็นต้น

ข้อมูลเหล่านี้อาจถูกรวบรวมโดยการสังเกตหรือโดยสมมติฐานสำหรับระบบ "กระดาษ" ซึ่งไม่มีอยู่เมื่อตัวแบบจำลองถูกพัฒนา พารามิเตอร์ของมันสามารถถูกปรับ (adjusted ) และทำการทดลองได้อย่างง่าย ผลกระทบของการเปลี่ยนแปลงที่มีต่อระบบสืบสวนได้ตัวอย่างเช่น เกิดอะไรขึ้นกับพฤติกรรมของระบบถ้าเราเพิ่ม additional checkstand

เกิดอะไรขึ้นถ้ามี quick–check lines สองแถวแทนที่จะเป็นหนึ่งแถว

เกิดอะไรขึ้นถ้าเราค่อย ๆ ลดอัตราการให้บริการ เช่น ต้องการให้ลูกค้าทั้งหมดตรวจทานโดยผู้จัดการร้าน

**ทฤษฎีแถวคอย** คือวิธีการวิเคราะห์เพื่อศึกษาพฤติกรรมของระบบซึ่งเกี่ยวข้องกับแถวคอย

(**Queueing theory** is an analytic approach to studying behavior of systems that involve queues.)

## แบบฝึกหัด

1. จงอธิบายความเหมือนกันและความแตกต่างกันระหว่างแถวคอยและกองซ้อน
2. ปลายด้านใดของแถวคอยซึ่งจะใส่สมาชิกและปลายด้านใดของแถวคอยซึ่งจะลบสมาชิก
3. ทำไมแถวคอยซึ่งเก็บในแถวลำดับปกติแทนที่ในแบบวงกลม ไม่ใช่แบบเชิงเส้น
4. กองซ้อนเป็นแบบวงกลมได้หรือไม่ ทำไมหรือทำไมไม่ได้
5. อัลกอริทึมเพื่อใส่สมาชิกหนึ่งตัวในแถวคอยจะบอกได้อย่างไรว่าแถวคอยเคลื่อนที่ตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา
6. เมื่อใดที่ฟังก์ชัน mod จะสะดวกในการนำมาใช้ในแถวคอยแบบวงกลม
7. จะทดสอบว่าแถวคอยว่างได้อย่างไร จงอธิบายการวางตำแหน่งพอยน์เตอร์ front
8. จงเขียนโปรซีเดอร์ใส่สมาชิกหนึ่งตัวให้กับแถวคอยแบบวงกลม เมื่อสมาชิกตัวแรกกำหนดโดย FRONT และ REAR แสดงพื้นที่หลังตัวสุดท้ายของแถวคอย
9. จงเขียนอัลกอริทึมนับจำนวน items ในแถวคอยแบบวงกลม
10. จงเขียนอัลกอริทึมลบสมาชิกหนึ่งตัวออกจากแถวคอยแบบวงกลมเมื่อ REAR ชี้ที่ rear และ FRONT ชี้ที่หนึ่งที่ข้างหน้า front
11. จงเขียนอัลกอริทึมใส่สมาชิกให้กับแถวคอยแบบวงกลมเมื่อ REAR ชี้ที่ rear และ FRONT ชี้ที่ front
12. จงวาดรูปแถวคอยที่มีสมาชิกสิบตัวแสดง front และ rearจากนั้น จงเขียนอัลกอริทึมเพื่อ ตรวจสอบว่าแถวคอยเต็มหรือไม่
13. จงเขียนอัลกอริทึมใส่แถวคอยสองชุดซึ่งความยาวผันแปร(แต่ผลรวมทั้งหมดของความยาวเท่ากับ N) ไว้ในแถวลำดับที่มี N ช่อง