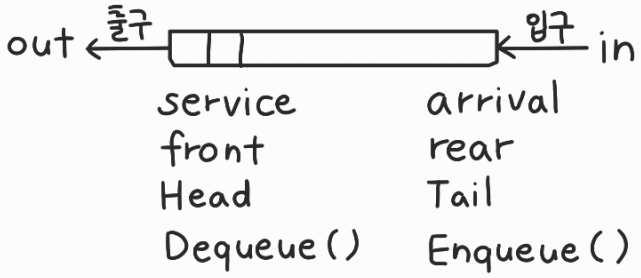
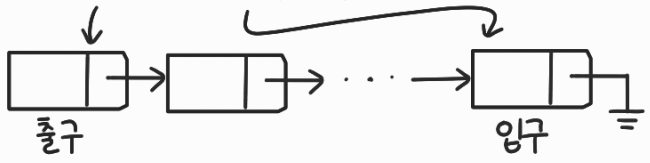
Ch 6 큐 (Queue)

Q : FIFO (First-In First-Out)



1. Link List 로 구현

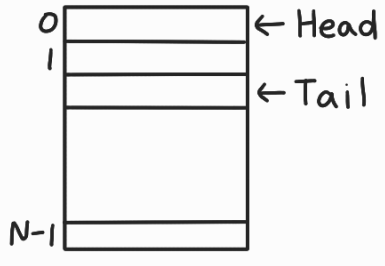
Node *Head, *Tail, *p




```
enQ(Node *p)
{
    p → Next = NULL
    if(Tail ≠ NULL)     // 맨 뒤에 삽입
    {  Tail → Next = p;  Tail = p; }
    else
        Head = Tail = p;   // empty에 추가
}
dQ()
{ 맨 앞의 원소를 삭제 및 반환 }
```

2. 배열 (array) 로 구현

Q [0... N-1]

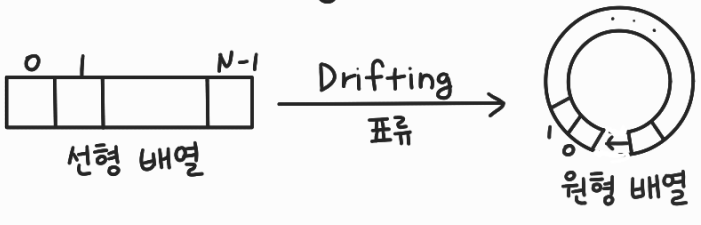


⇔

```
enQ(item)
{  Q[++Tail] = item  }
deQ()
{  ret  Q[Head++] }
```

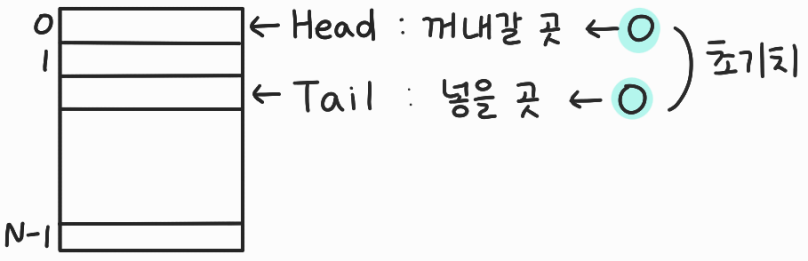
문제: 제한된 공간에서 data가 한 쪽으로만 이동하여 한계에 도달

→ Drifting (표류) 현상 발생



3. 원형배열 (circular array)

Q [0 ... N-1]



※ 원형 배열을 위해 (H = H ⊕ 1 ⇔ H = (H+1) % N
T = T ⊕ 1 ⇔ T = (T+1) % N

```
enQ(item)
{  if(FULL) ret
   Q[Tail] = item
   Tail = Tail ⊕ 1
}
```

```
deQ()
{  if(EMPTY) ret
   x = Q[Head]
   Head = Head ⊕ 1
   ret x
}
```

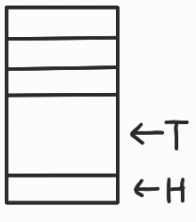
FULL ⇔ Head == Tail
EMPTY ⇔ Head == Tail
) 두 조건식이 동일하여 사용 불가!!

⇒ Head, Tail의 초기치 변경으로는 해결 불가 (확인요!)

해결책

① counter (= #q elements)를 도입 (why?)
→ Not recommendable!!
FULL ⇔ counter == N
EMPTY ⇔ counter == 0

② Buffer(Q)를 한 칸 덜 사용
→ Size N인 Q에 N-1개까지 채워지면 FULL로 간주

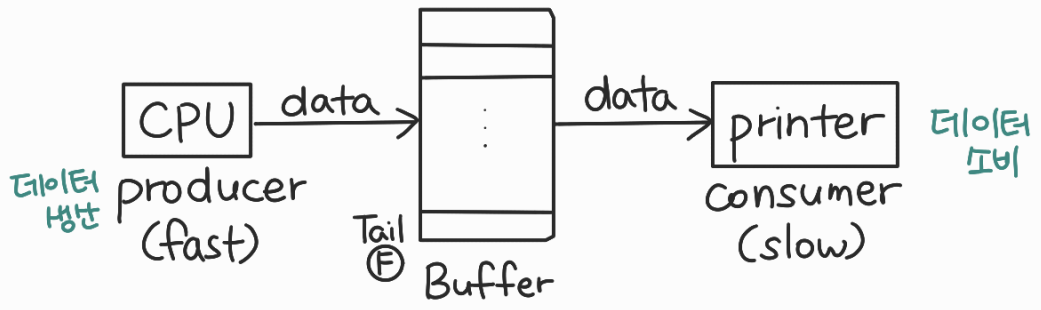


EMPTY ⇔ Head == Tail
FULL ⇔ Head == Tail ⊕ 1

③ Head, Tail 의 위치정보를 유지하면서 값을 다르게 하는 방법

→ Producer / Consumer problem 에 실제 적용
생산자 / 소비자 문제

✳ 생산자 / 소비자 문제 ↷ 큐를 버퍼로 사용



<producer>

```
while (1)
{ produce an Item   (데이터 만들기)
  while (FULL)      ↷ busy loop
    B[in % N] ← item
    in = in ⊕ 1 + N
}
```

차 있는 동안 → 넣을 수 없다

삽입 후 → in ≥ N

<consumer>

```
while(1)
{ while (EMPTY)
    Item ← B[out]
    out = out ⊕ 1
    in ← in % N
    Consume item
}
```

삭제 후 → in < N

FULL ⇔ (in + 1) % N == out in ≥ N

EMPTY ⇔ in == out in < N