HÀNG ĐỢI (Queue)

Nội dung

- Hàng đợi (Queue)
  - Khái niệm Queue
  - Các thao tác trên Queue
  - Hiện thực Queue
  - Ứng dụng Queue

Queue - Khái niệm

- Queue là một danh sách mà các đối tượng được **thêm vào** ở một đầu của danh sách và **lấy ra** ở một đầu kia của danh sách (*A queue is also a list of elements with insertions permitted at one end and deletions permitted from the other end)*
- Việc **thêm** một đối tượng vào Queue luôn diễn ra ở cuối Queue và việc **lấy** một đối tượng ra khỏi Queue luôn diễn ra ở đầu Queue
- Vì thế, việc thêm một đối tượng vào Queue hoặc lấy một đối tượng ra khỏi Queue được thực hiện theo cơ chế FIFO (First In First Out - *Vào trước ra trước*)

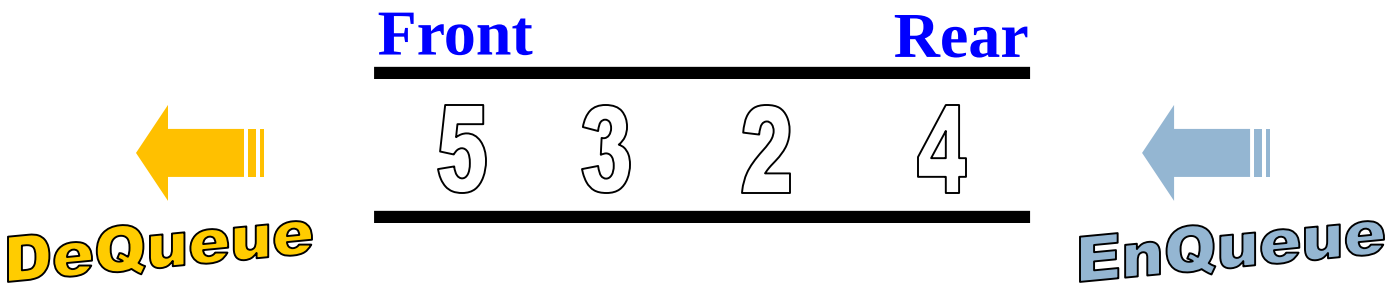
Queue - Khái niệm

Imaging

Queue – Các thao tác

- Hàng đợi hỗ trợ các thao tác:
  - **EnQueue**(): **Thêm** đối tượng vào cuối (rear) Queue
  - **DeQueue**(): **Lấy** đối tượng ở đầu (front) Queue ra khỏi Queue
- Ví dụ:

  5 3 2 - - 4



Queue – Các thao tác

- Queue còn hỗ trợ các thao tác:
  - **isEmpty**(): Kiểm tra xem hàng đợi có rỗng không
  - **Front**(): Trả về giá trị của phần tử nằm ở đầu hàng đợi mà không hủy nó. Nếu hàng đợi rỗng thì lỗi sẽ xảy ra

Queue – Hiện thực Queue
(Implementation of a Queue)

Mảng 1 chiều

Kích thước queue khi quá thiếu, lúc quá thừa

EnQueue/DeQueue hơi phức tạp

Danh sách LK

Cấp phát động!

EnQueue/DeQueue khá dễ dàng

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

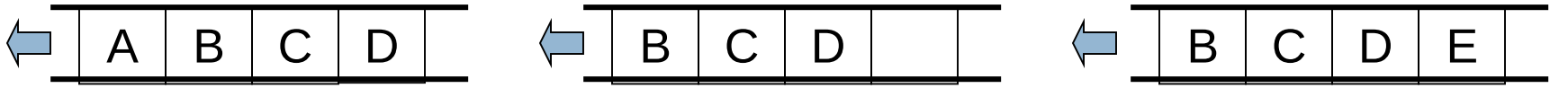
- Có thể tạo một Queue bằng cách sử dụng một mảng 1 chiều theo kiểu xoay vòng (coi phần tử a_{n-1} kề với phần tử a_0)

  $\Rightarrow$ Hàng đợi chứa tối đa N phần tử
- Phần tử ở đầu hàng đợi sẽ có chỉ số front
- Phần tử ở cuối hàng đợi sẽ có chỉ số rear

- The limitation of an array implementation is that the queue cannot grow and shrink dynamically as per the requirement

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Dùng mảng: Có xu hướng dời về cuối mảng
- Hai cách hiện thực:
  - **Khi lấy một phần tử ra thì đồng thời dời ô lên một vị trí:**

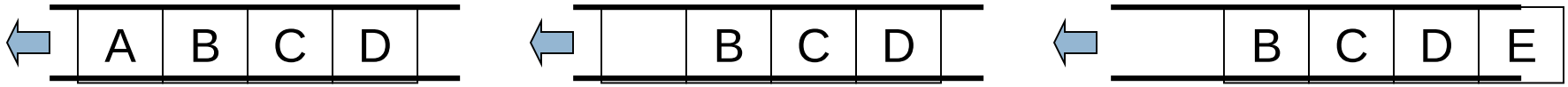


Ban đầu

Lấy ra 1 phần tử: dời tất cả về trước để trống chỗ thêm vào

Thêm vào 1 phần tử

  - **Khi lấy một phần tử ra thì không dời ô lên:**



Ban đầu

Lấy ra 1 phần tử

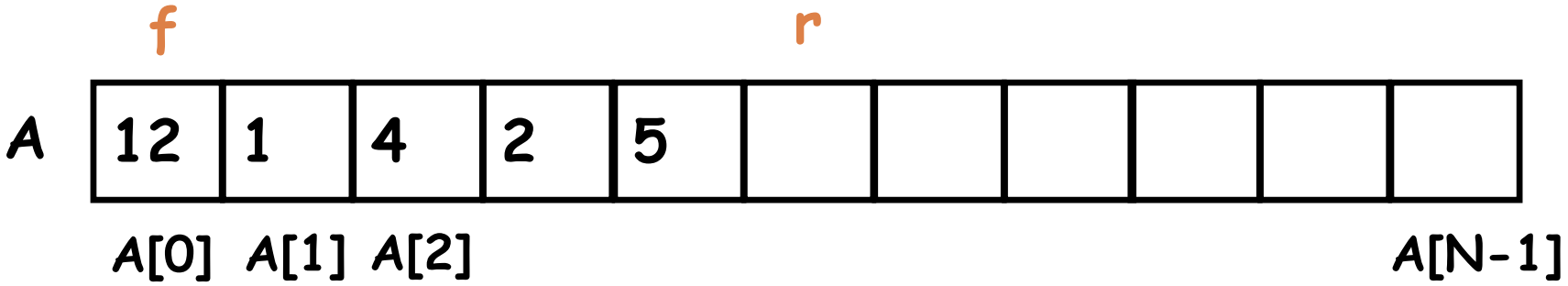
Thêm vào 1 phần tử

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Trạng thái Queue lúc bình thường:

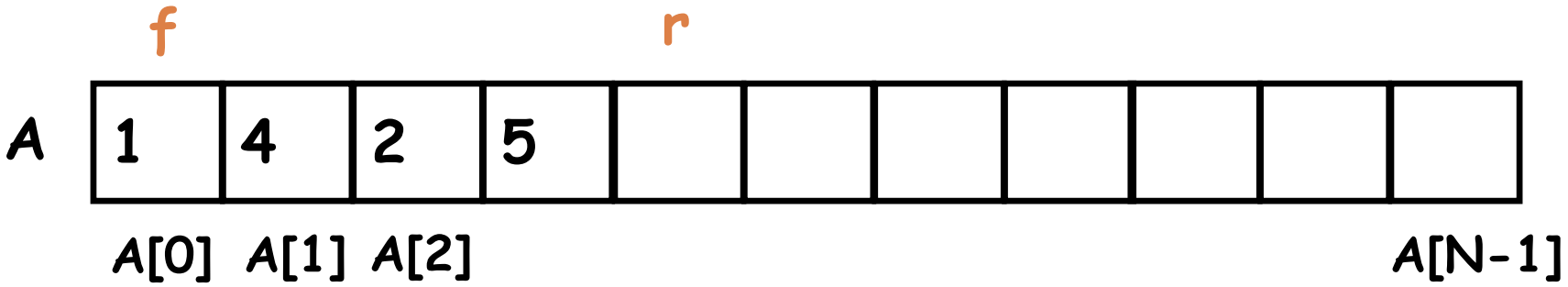
- Trạng thái Queue lúc xoay vòng:

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



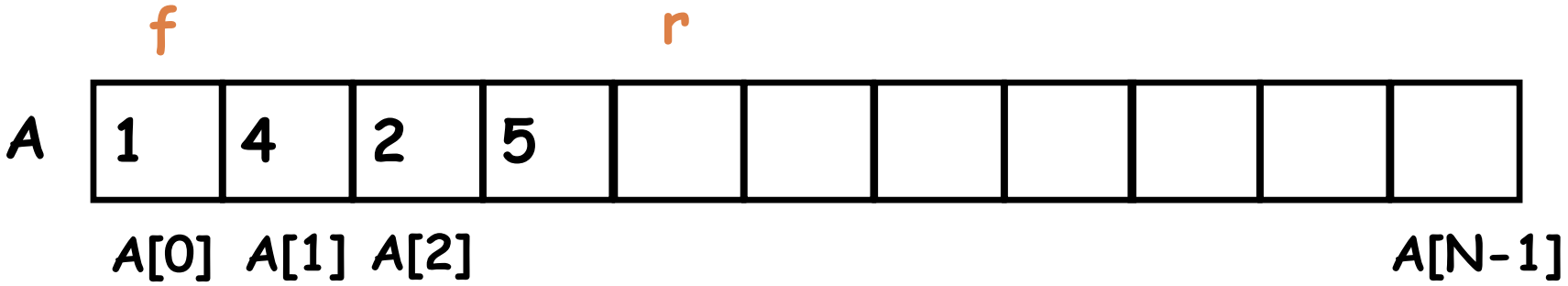
DeQueue(Q)

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



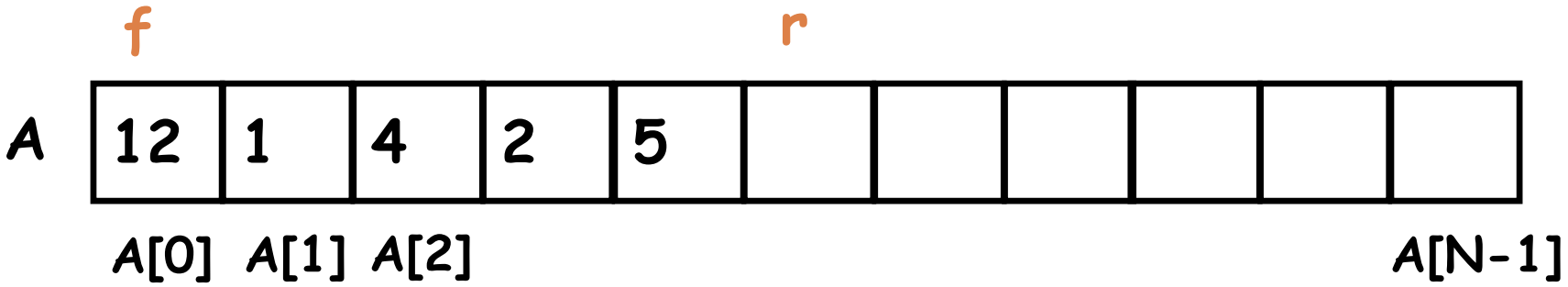
DeQueue(Q)

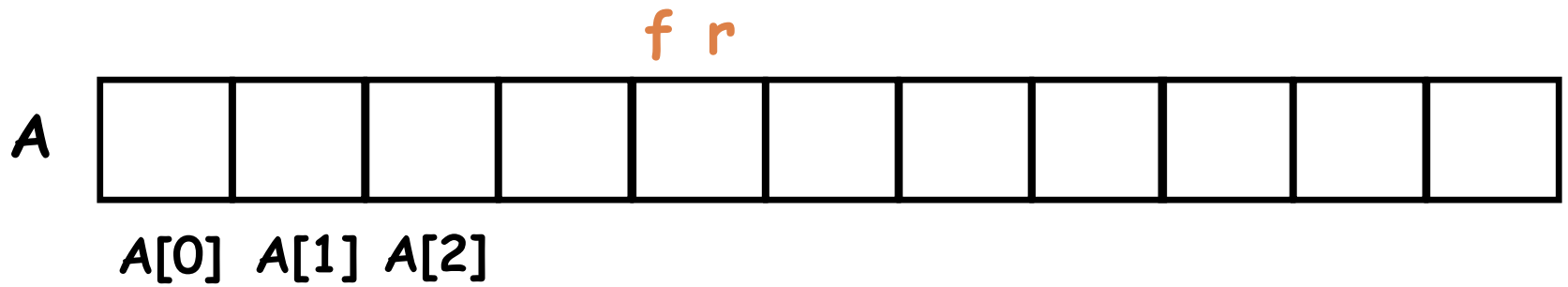
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



Cách dùng mảng 1

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

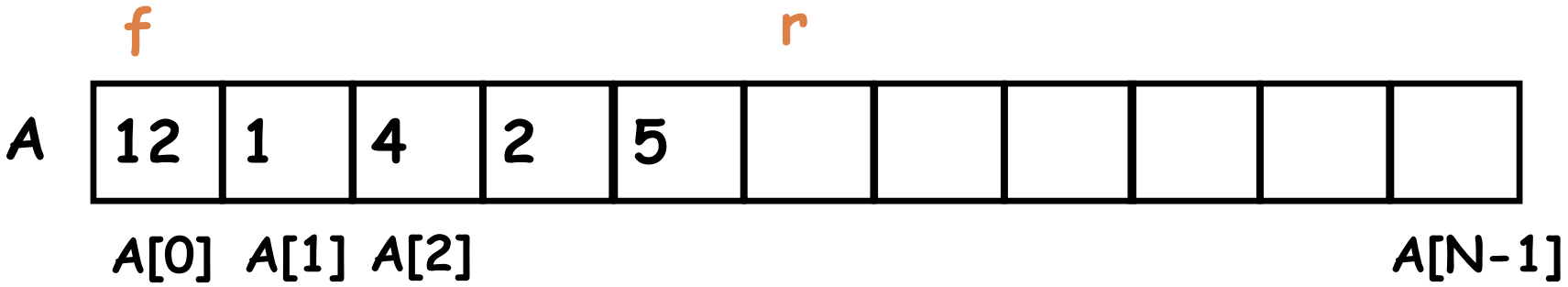




Cách dùng mảng 2

Empty queue f=r

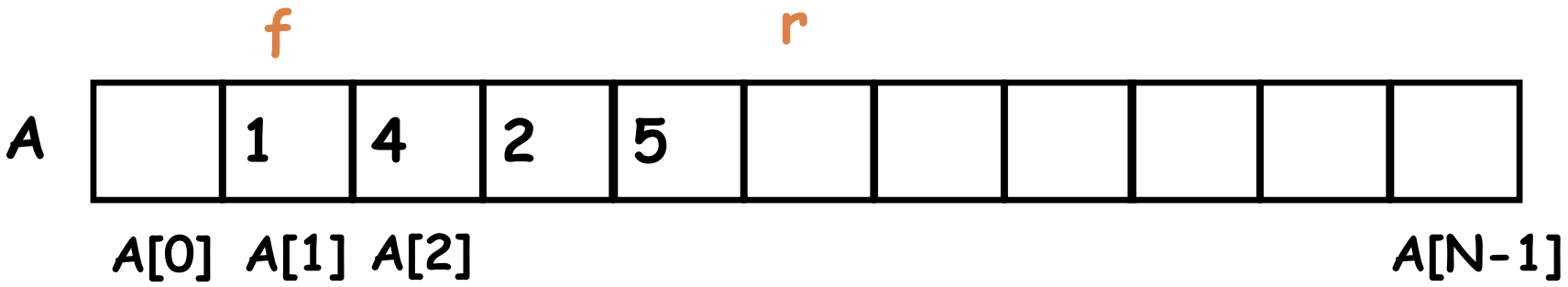
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q)

Cách dùng mảng 2

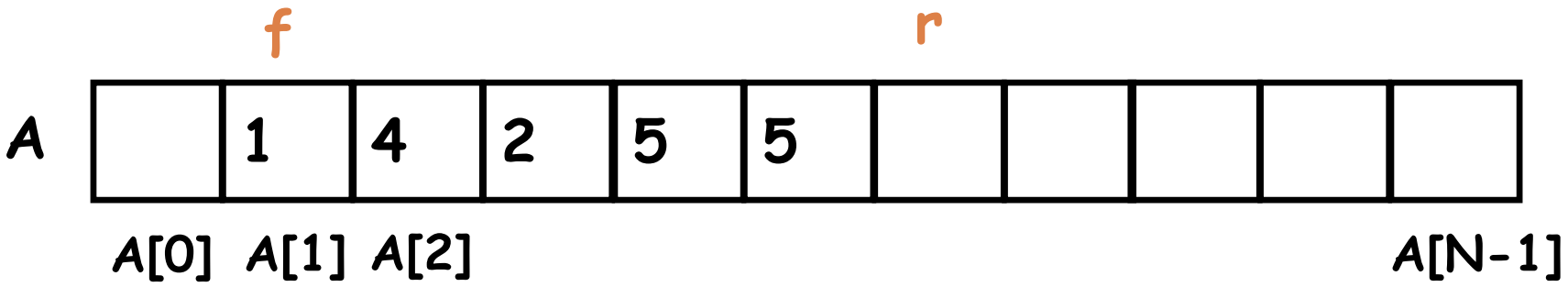
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q)
EnQueue(5,Q)

Cách dùng mảng 2

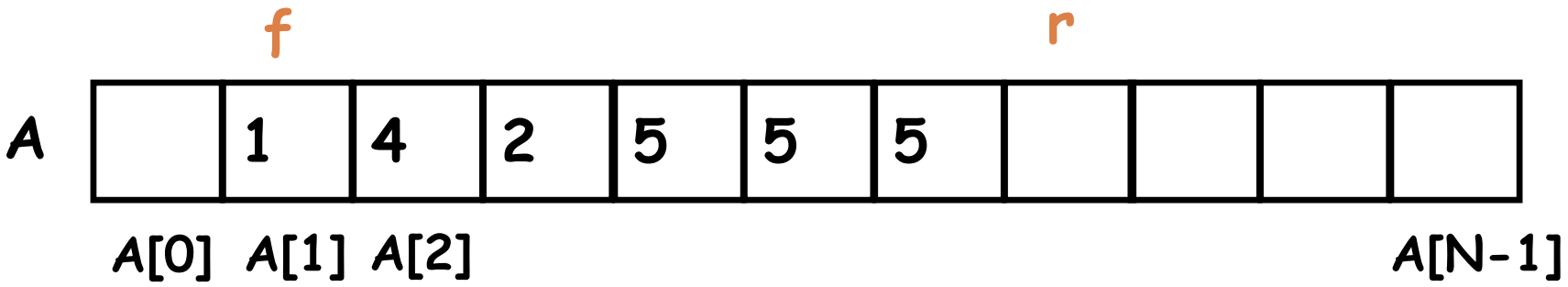
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q)
EnQueue(5,Q)
EnQueue(5,Q)

Cách dùng mảng 2

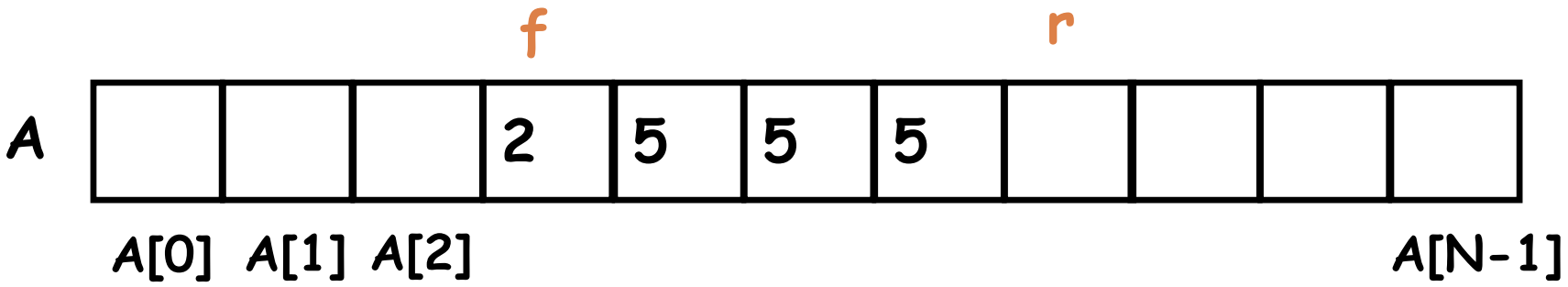
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q)
EnQueue(5,Q)
EnQueue(5,Q)
DeQueue(Q)
DeQueue(Q)

Cách dùng mảng 2

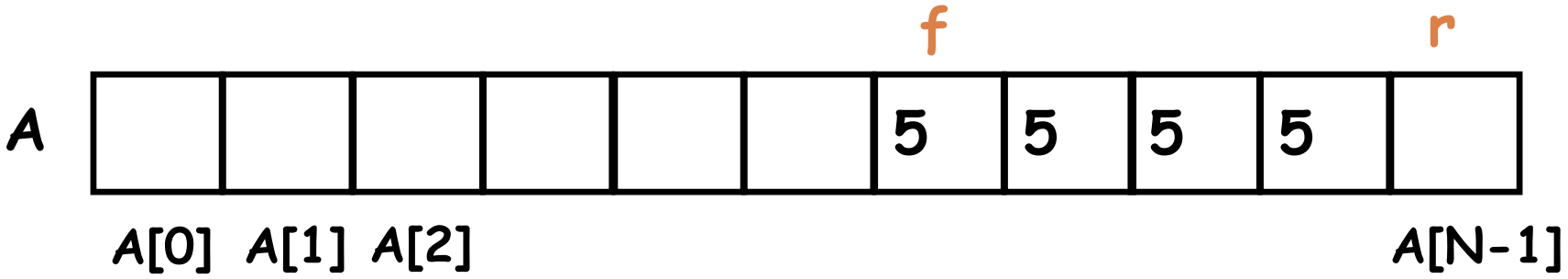
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q)
EnQueue(5,Q)
EnQueue(5,Q)
DeQueue(Q)
DeQueue(Q)
DeQueue(Q), EnQueue(5,Q), DeQueue(Q), EnQueue(5,Q),…….

Cách dùng mảng 2

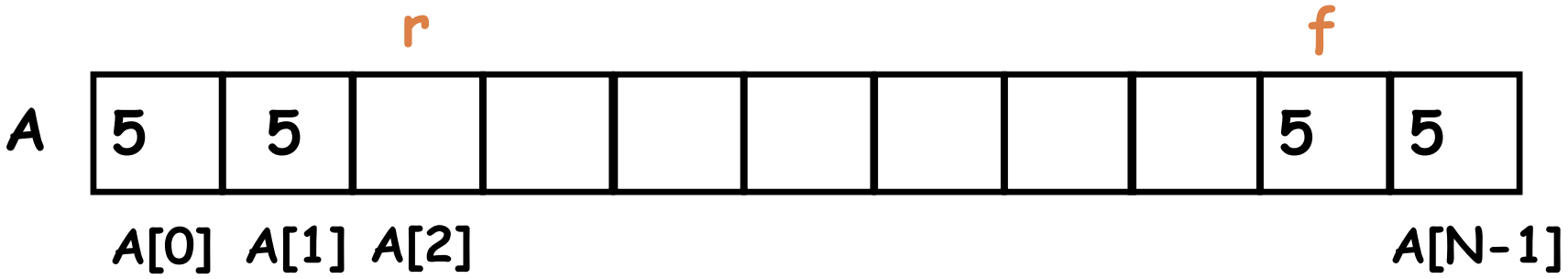
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q)
EnQueue(5,Q)
EnQueue(5,Q)
DeQueue(Q)
DeQueue(Q)
DeQueue(Q), EnQueue(5,Q), DeQueue(Q), EnQueue(5,Q),…….

Cách dùng mảng 2

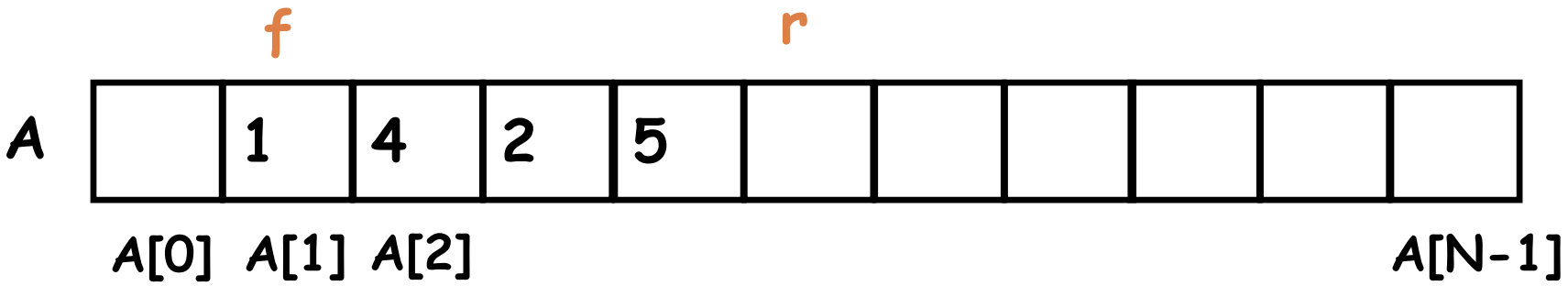
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



DeQueue(Q), EnQueue(5,Q), DeQueue(Q), EnQueue(5,Q),…….

Cách dùng mảng 2

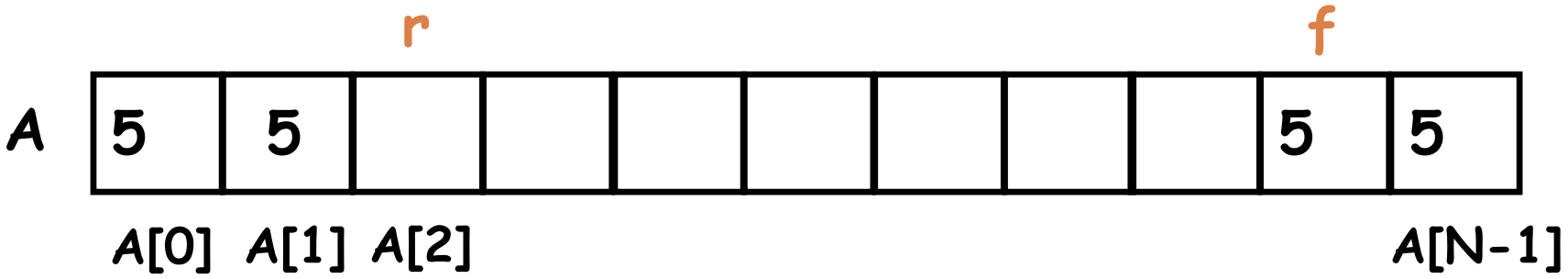
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)



size(Q): if (r >= f) then return (r-f)
else return N-(f-r)

Cách dùng mảng 2

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)




```
size(Q):  if (r >= f) then return (r-f)
          else return N-(f-r)
```

Cách dùng mảng 2

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Để khai báo một Queue, ta cần khai báo:
  - một mảng một chiều list,
  - hai biến nguyên front, rear cho biết chỉ số của đầu và cuối của hàng đợi,
  - hằng số N cho biết kích thước tối đa của Queue
- Hàng đợi có thể được khai báo cụ thể như sau:

```
struct Queue
{
    int      front, rear;
    DataType  list[N];
};
```

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Do khi cài đặt bằng mảng một chiều, hàng đợi bị giới hạn kích thước nên cần xây dựng thêm một thao tác phụ cho hàng đợi:
  - **isFull**(): Kiểm tra xem hàng đợi có đầy chưa

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- **Khởi tạo Queue:**

```
void   Init(Queue &q)
{
     q.front = q.rear = 0;
}
```

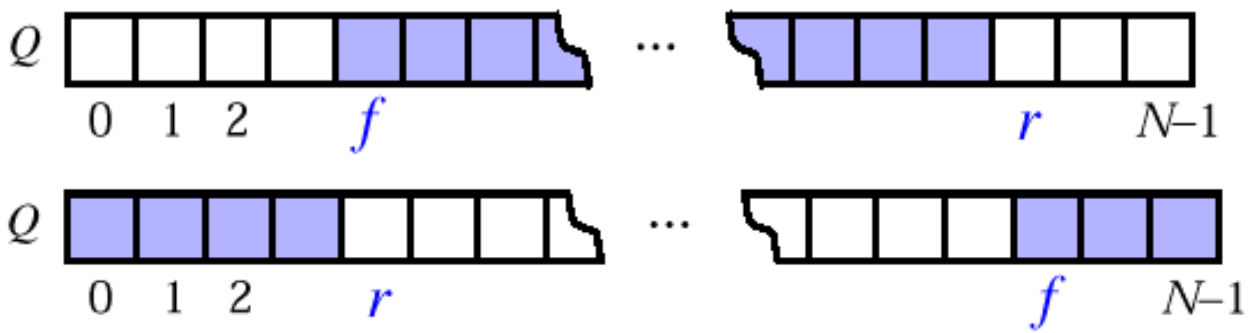
- **Kiểm tra xem Queue có rỗng không:**

```
int isEmpty(Queue q)
{
      if (q.front == q.rear == 0)
            return 1;
      if (q.front == q.rear)
            return 1;
      return 0;
}
```

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- **Kiểm tra hàng đợi đầy hay không:**

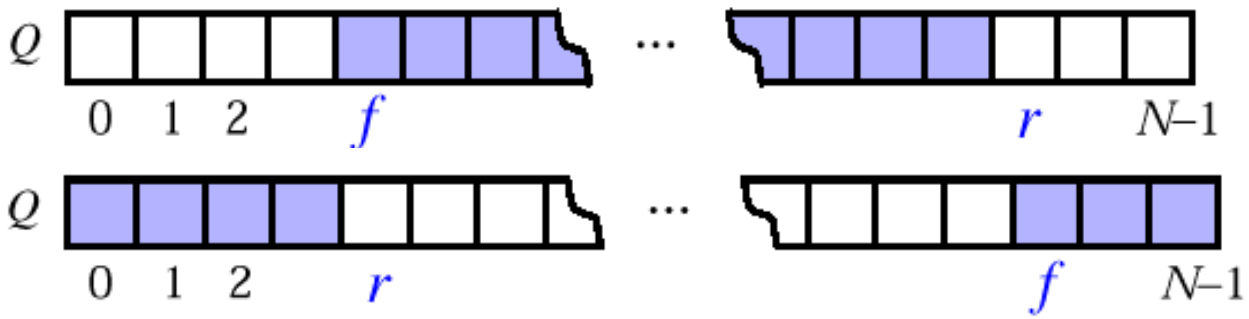
```
int isFull(Queue q)
{
  if (q.front == 0 && q.rear == N-1)
    return 1;
  if (q.front == q.rear)
    return 1;
  return 0;
}
```



Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Thêm một phần tử x vào cuối Queue:

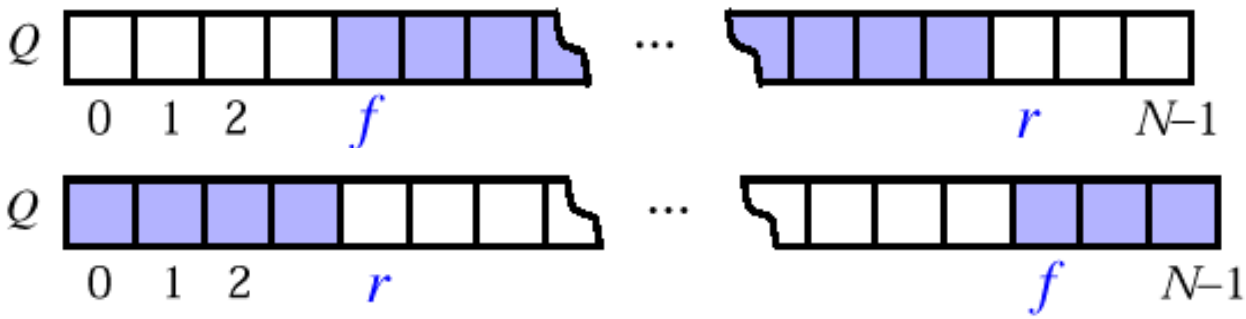
```
int  EnQueue(Queue &q, DataType x)
{
     if (isFull(q))
          return 0; // không thêm được vì Queue đầy
     q.list[q.rear] = x;
     q.rear++;
     if (q.rear == N) q.rear=0;
     return 1;
}
```



Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Trích, huỷ phần tử ở đầu hàng đợi:

```
DataType  DeQueue(Queue &q)
{
      if (Empty(q))  return NULLDATA;
      DataType t = q.list[q.front];
      q.front++;
      if (q.front == N) q.front = 0;
      return t;
}
```



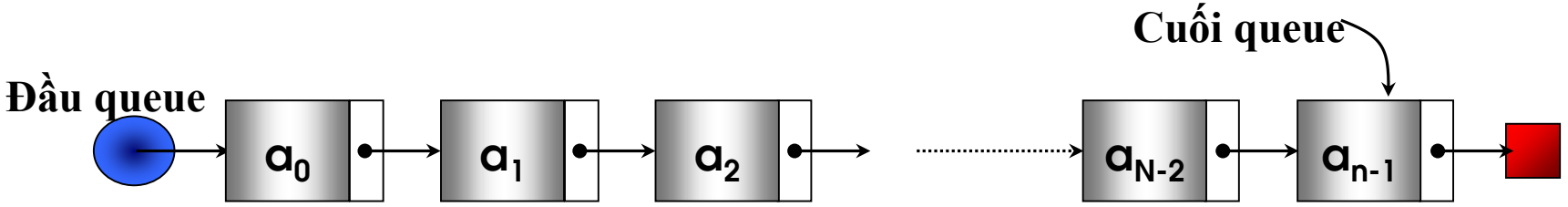
Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Xem thông tin của phần tử ở đầu Queue:

```
DataType      Front(Queue q)
{
   if (isEmpty(q)) return NULLDATA;
   return q.list[q.front];
}
```

Hiện thực Queue dùng DSLK
(Implementation of a Queue using Linked List)

- Có thể tạo một hàng đợi sử dụng một DSLK đơn
- Phần tử đầu DSKL (phead) sẽ là phần tử đầu Queue (front), phần tử cuối DSKL (ptail) sẽ là phần tử cuối Queue (rear)



Hiện thực Queue dùng DSLK
(Implementation of a Queue using Linked List)

- Khai báo các cấu trúc:

```
struct Node
{
    DataType data;
    Node *pNext;
};
struct Queue
{
    Node *front, *rear;
};
```

Hiện thực Queue dùng DSLK
(Implementation of a Queue using Linked List)

- Khởi tạo Queue rỗng:

```
void Init(Queue &q)
{
    q.front = q.rear = NULL;
}
```

- Kiểm tra hàng đợi rỗng :

```
int  isEmpty(Queue &q)
{
    if (q.front == NULL)
        return 1;
    else
        return 0;
}
```

Hiện thực Queue dùng DSLK
(Implementation of a Queue using Linked List)

- **Thêm một phần tử p vào cuối Queue:**

```
int EnQueue(Queue &q, DataType x)
{
   Node *p = new Node;
   if (p==NULL) return 0; //Khong du bo nho
   p->pNext = NULL;
   p->data = x;
   if (q.front == NULL)      // TH Queue rỗng
      q.front = q.rear  = new_node;
   else
   {
      q.rear->pNext = new_node;
      q.rear = new_node;
   }
   return 1;
}
```

Hiện thực Queue dùng DSLK
(Implementation of a Queue using Linked List)

- **Trích và huỷ phần tử ở đầu Queue:**

```
DataType DeQueue(Queue &q)
{
    if (isEmpty(q)) return NULLDATA;
    Node *p = q.front;
    p->pNext = NULL;
    q.front = q.front->pNext;
    if (q.front==NULL) q.rear = NULL;
    DataType x = p->data;
    delete p;
    return x;
}
```

Hiện thực Queue dùng mảng
(Implementation of a Queue using Array)

- Xem thông tin của phần tử ở đầu Queue:

```
DataType Front(Queue q)
{
    if (isEmpty(q)) return NULLDATA;
    return q.front->data;
}
```

Hiện thực Queue dùng DSLK
(Implementation of a Queue using Linked List)

✍ **Nhận xét:**

- Các thao tác trên Queue biểu diễn bằng danh sách liên kết làm việc với chi phí O(1)
- Nếu không quản lý phần tử cuối xâu, thao tác **Dequeue** sẽ có độ phức tạp O(n)

Queue - Ứng dụng

- Queue có thể được sử dụng trong một số bài toán:
  - Bài toán "sản xuất và tiêu thụ" (ứng dụng trong các hệ điều hành song song)
  - Bộ đệm (ví dụ: Nhấn phím ⇒ Bộ đệm ⇒ CPU xử lý)
  - Xử lý các lệnh trong máy tính (ứng dụng trong HĐH, trình biên dịch), hàng đợi các tiến trình chờ được xử lý, ….