CHƯƠNG 4. HÀNG ĐỢI

ThS. Nguyễn Chí Hiếu

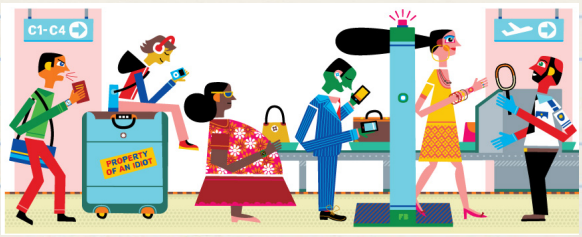
2021

NỘI DUNG

Giới thiệu hàng đợi

Hàng đợi (Queue)

- Thực hiện theo cơ chế FIFO *(First In, First Out)* vào trước ra trước.
- Dùng để lưu trữ các phần tử có thứ tự truy xuất *đúng* với thứ tự lưu trữ *(vào trước, ra trước)*.



Hình 1: Hình minh họa hàng đợi.

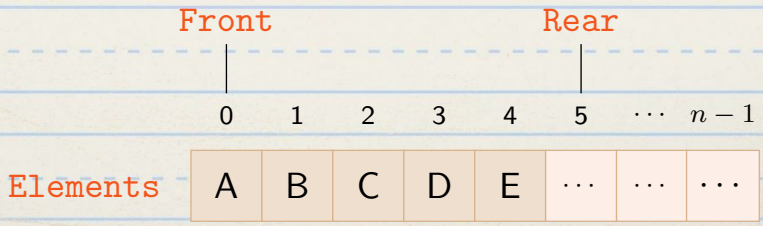
Giới thiệu hàng đợi

Các thao tác cơ bản

- EnQueue: *thêm* phần tử vào *cuối* hàng đợi.
- DeQueue: *lấy và xóa* phần tử tại *đầu* hàng đợi.
- GetFront: *xem thông tin* phần tử tại *đầu* hàng đợi.
- Kiểm tra hàng đợi rỗng, đầy.

Cài đặt hàng đợi bằng mảng

- Biến `Elements` là mảng 1 chiều kích thước n: lưu trữ phần tử từ vị trí $[0, ..., n-1]$.
- Biến `Front`, `Rear` kiểu số nguyên: cho biết vị trí đầu và cuối.
- Mặc định, hàng đợi vừa khởi tạo `Front = Rear = 0` và tất cả phần tử của mảng `Elements` gán bằng `NULL_DATA`.



Cài đặt hàng đợi bằng mảng

```
1  public class Queue
2  {
3      public int[] elements;
4      public int front;
5      public int rear;
6
7      public void InitQueue()
8      {
9          elements = new int[MAX_SIZE];
10         front = rear = 0;
11     }
12 }
```

Cài đặt hàng đợi bằng mảng

Thuật toán 1: IsEmpty(q)

- Đầu vào: hàng đợi q.
- Đầu ra: true/false.

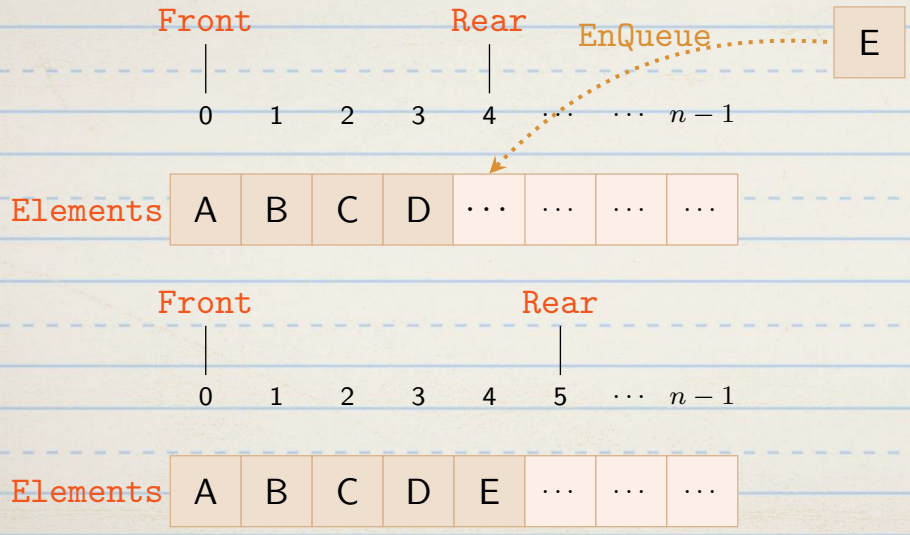
```
1 if elements[front] ≠ QUEUE_EMPTY
2     return false
3 return true
```

Thuật toán 2: IsFull(q)

- Đầu vào: hàng đợi q.
- Đầu ra: true/false.

```
1 if elements[rear] = QUEUE_EMPTY
2     return false
3 return true
```

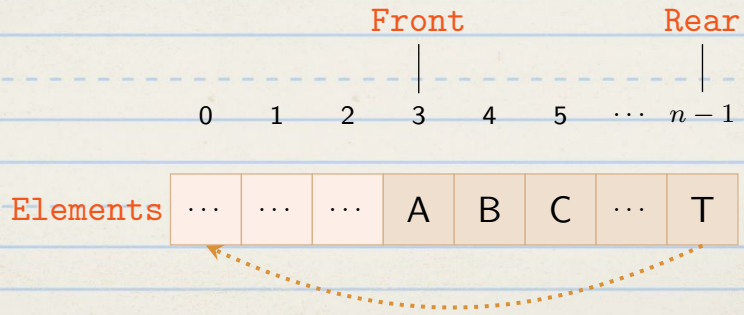

Cài đặt hàng đợi bằng mảng



Cài đặt hàng đợi bằng mảng

Xử lý vấn đề tràn giả

Sử dụng mảng như danh sách vòng.



- EnQueue: nếu đến cuối mảng, cập nhật Rear = 0.

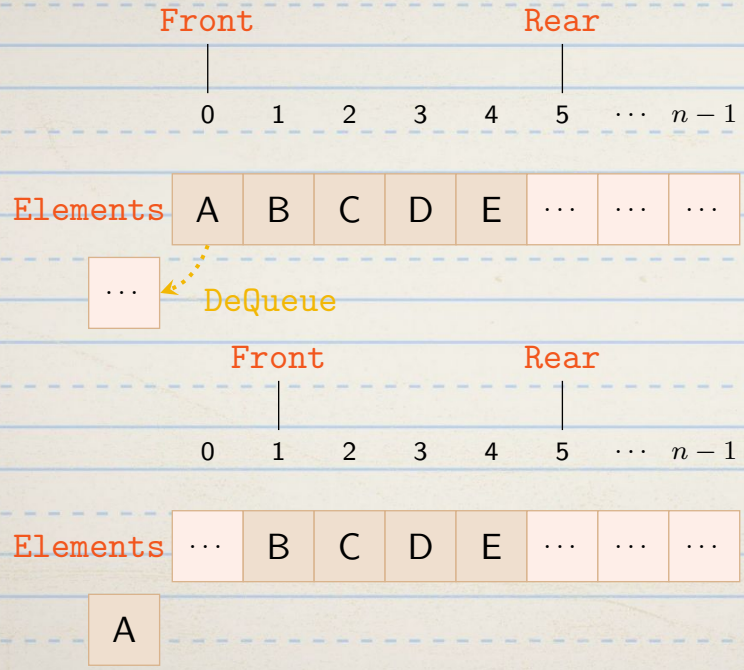
Cài đặt hàng đợi bằng mảng

Thuật toán 3: EnQueue(q, x)

- Đầu vào: hàng đợi q và phần tử x cần thêm.
- Đầu ra: hàng đợi q sau khi thêm x.

```
1 if hàng đợi chưa đầy
2     elements[rear] ← x
3     rear ← rear + 1
4     if rear = MAX_SIZE
5         rear ← 0
```

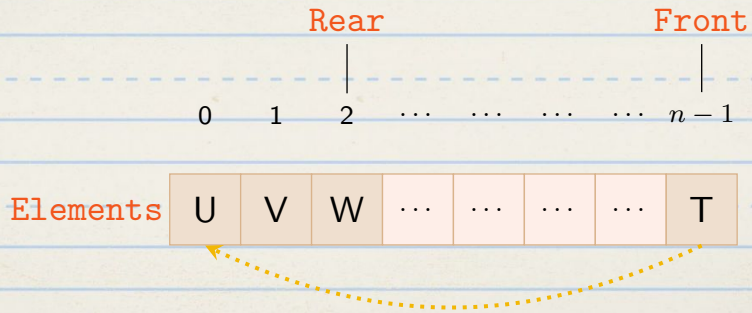
Cài đặt hàng đợi bằng mảng



Cài đặt hàng đợi bằng mảng

Xử lý vấn đề tràn giả

Sử dụng mảng như danh sách vòng.



- DeQueue: nếu đến cuối mảng, cập nhật Front = 0.

Cài đặt hàng đợi bằng mảng

Thuật toán 4: DeQueue(q)

- Đầu vào: hàng đợi q.
- Đầu ra: phần tử đầu hàng đợi hay QUEUE_EMPTY (hàng đợi rỗng).

```
1 if hàng đợi khác rỗng
2     x ← elements[front]
3     elements[front] ← QUEUE_EMPTY
4     front ← front + 1
5     if front = MAX_SIZE
6         front ← 0
7     return x
8 return QUEUE_EMPTY
```

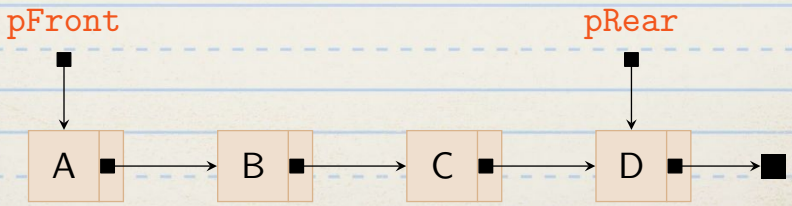

Cài đặt hàng đợi bằng mảng

Thuật toán 5: GetFront(q)

```
1 if hàng đợi khác rỗng
2     return elements[front]
3 return QUEUE_EMPTY
```

Cài đặt hàng đợi bằng danh sách liên kết

- Cấu trúc dữ liệu một phần tử của hàng đợi chứa thành phần dữ liệu và thành phần liên kết *(tương tự danh sách liên kết)*.
- Cấu trúc dữ liệu hàng đợi chứa hai con trỏ `pFront` trỏ đến phần tử *đầu* và con trỏ `pRear` trỏ đến phần tử *cuối* của hàng đợi.
- Thao tác thêm thực hiện ở cuối và thao tác xóa thực hiện ở đầu hàng đợi.



Cài đặt hàng đợi bằng danh sách liên kết

Định nghĩa cấu trúc của một phần tử trong hàng đợi và hàm khởi tạo một nút trong hàng đợi.

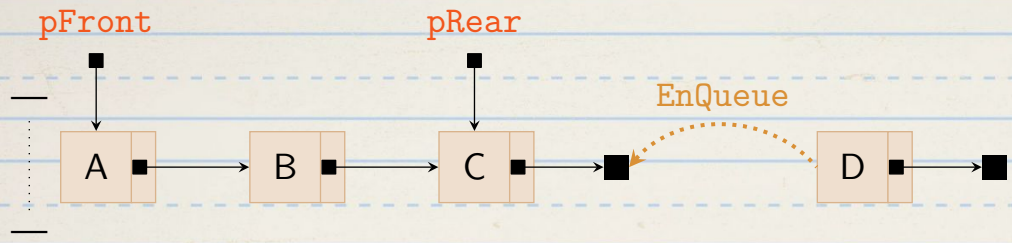
```
1  public class Node
2  {
3      public int info;
4      public Node pNext;
5
6      public void InitNode(int x)
7      {
8          info = x;
9          pNext = null;
10     }
11 }
```

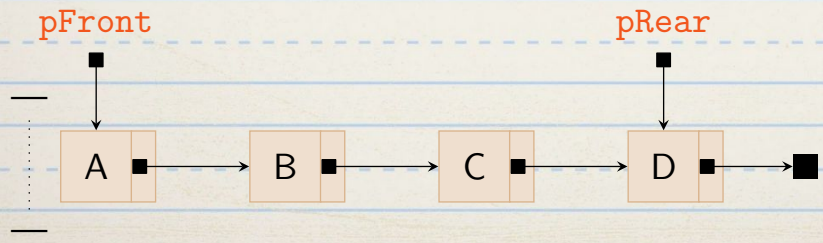
Cài đặt hàng đợi bằng danh sách liên kết

Định nghĩa cấu trúc của một hàng đợi và hàm khởi tạo hàng đợi.

```
1  public class Queue
2  {
3      public Node pFront;
4      public Node pRear;
5      public int size;
6
7      public void InitQueue()
8      {
9          pFront = null;
10         pRear = null;
11         size = 0;
12     }
13 }
```


Cài đặt hàng đợi bằng danh sách liên kết

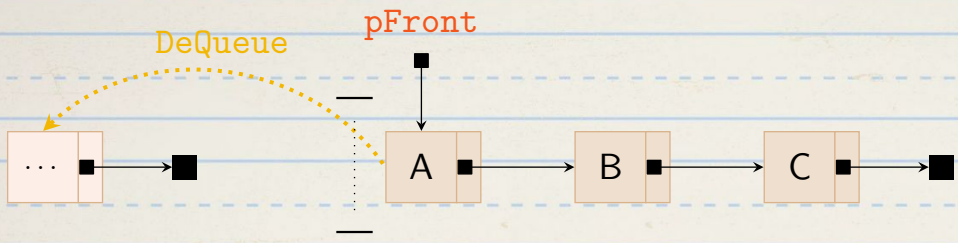


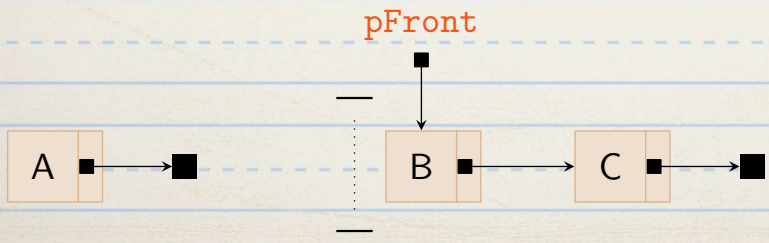


Thao tác thêm phần tử

```
1  // Thêm phần tử vào hàng đợi (thêm cuối)
2  public void EnQueue(Node p)
3  {
4      if (p == null)
5          return;
6      if (pFront == null)
7      {
8          pFront = p;
9          pRear = p;
10     }
11     else
12     {
13         pRear.pNext = p;
14         pRear = p;
15     }
16     size++;
17 }
```


Cài đặt hàng đợi bằng danh sách liên kết





Thao tác lấy phần tử

```
1  // Lấy phần tử ra khỏi đầu hàng đợi (xóa đầu)
2  public int DeQueue()
3  {
4      if (pFront == null)
5          return QUEUE_EMPTY;
6      Node p = pFront;
7      int x = p.info;
8      pFront = pFront.pNext;
9      size--;
10     p = null; // delete p
11     return x;
12 }
```


Thao tác lấy phần tử

```
1  public int GetFront()
2  {
3      if (pFront == null)
4          return QUEUE_EMPTY;
5      return pFront.info;
6  }
```

Ứng dụng của hàng đợi

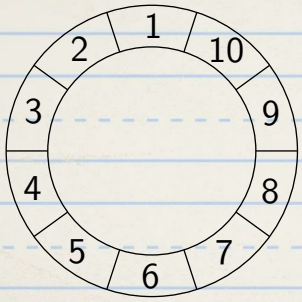
Một số ứng dụng của hàng đợi

- Trong một số thuật toán của lý thuyết đồ thị, hàng đợi được sử dụng để lưu dữ liệu khi thực hiện.
- Bài toán sản xuất và tiêu thụ.
- Quản lý bộ đệm (ví dụ: nhấn phím $\rightarrow$ bộ đệm $\rightarrow$ CPU xử lý).
- Xử lý các lệnh/tiến trình trong máy tính (ví dụ: hàng đợi máy in)

Ứng dụng của hàng đợi

Ví dụ 1 (Bài toán Josephus)

Cho n người đứng thành vòng tròn và một số nguyên m, với $m < n$.



- Giả sử
  - $n = 10$
  - $m = 3$

- Bắt đầu vị trí s, bài toán sẽ đếm từng người theo một chiều nhất định. Sau khi có $m - 1$ người được bỏ qua, người thứ m sẽ bị xử tử.
- Quy luật lặp lại đến khi còn $m - 1$ người sống sót.

Câu hỏi m - 1 người còn sống đứng vị trí nào ?

Bài toán Josephus

Thuật toán 6: Josephus(n, m)

- Đầu vào: n là số người và m là một số nguyên
- Đầu ra: in ra thứ tự người bị xử tử

```
1  for i ← 1 to n
2      EnQueue(q, i)
3
4  while q ≠ ∅
5      for i ← 1 to m - 1
6          EnQueue(q, DeQueue(q))
7      x ← DeQueue(q)
8      Print x
```


Bài toán Josephus

Giải thích

- Dòng 1 → 2: đưa tất cả người tham gia vào hàng đợi.
- Dòng 4 → 7: hàng đợi khác rỗng, bắt đầu đếm và thực hiện
  - Dòng 5 → 6: đưa $m-1$ người vào hàng đợi.
  - Dòng 7: chọn người vị trí m.
- Dòng 8: in thứ tự người bị chọn (trong đó, hai người ở vị trí cuối cùng sẽ sống sót)

Bài tập

1. Cho một hàng đợi rỗng, hãy lần lượt thực hiện các thao tác sau đây: EnQueue(1), EnQueue(5), EnQueue(2), EnQueue(7), DeQueue(), DeQueue(), EnQueue(9). Hãy vẽ ngăn xếp tương ứng với các thao tác trên.
2. Hàng đợi được cài đặt lại bằng cách sử dụng 2 ngăn xếp: ngăn xếp thứ nhất đặt tên là inStack và ngăn xếp thứ hai là outStack. *Chú ý, chỉ sử dụng các thao tác của cấu trúc ngăn xếp.*
3. Cài đặt phiên bản hàng đợi sử dụng hai thao tác *thêm đầu* và *xóa cuối* danh sách.
4. Áp dụng hàng đợi viết hàm cài đặt thuật toán Josephus.

Tài liệu tham khảo


Donald E. Knuth.
The Art of Computer Programming, Volume 3.
Addison-Wesley, 1998.


Dương Anh Đức, Trần Hạnh Nhi.
Nhập môn Cấu trúc dữ liệu và Thuật toán.
Đại học Khoa học tự nhiên TP Hồ Chí Minh, 2003.


Niklaus Wirth.
Algorithms + Data Structures = Programs.
Prentice-Hall, 1976.


Robert Sedgewick.
Algorithms in C.
Addison-Wesley, 1990.