Cấu trúc dữ liệu và thuật toán

PGS. TS. Phạm Tuấn Minh
Trường Công nghệ Thông tin, Đại học Phenikaa
minh.phamtuan@phenikaa-uni.edu.vn
https://sites.google.com/site/phamtuanminh/

Chương 2: Mảng và danh sách liên kết

- Cấu trúc lưu trữ mảng
- Danh sách liên kết
- **Hàng đợi**
- Ngăn xếp

Bài trước... bài này

Hàng đợi

- **Ví dụ về hàng đợi**
- Cấu trúc dữ liệu hàng đợi
- Cài đặt hàng đợi dùng danh sách liên kết
- Các thao tác hàng đợi
  - enqueue()
  - dequeue()
  - peek()
  - isEmptyQueue()
- Ví dụ ứng dụng

Ví dụ 1

- Xếp hàng thanh toán tại siêu thị
  - Khách hàng xếp trước được thanh toán trước
  - Khách hàng muốn thanh toán thì xếp vào cuối của hàng đợi và đợi đến lượt thanh toán

Ví dụ 2

- Trình điều khiển máy in:
  - Công việc in được gửi tới trình điều khiển máy in tại bất kì thời điểm nào
  - Công việc in phải được lưu tới khi chuyển tới máy in
  - Công việc in được chuyển tới máy in theo thứ tự yêu cầu gửi tới trước được phục vụ trước
  - Công việc in cần một khoảng thời gian để hoàn thành
  - Khi một công việc in hoàn thành, công việc in tiếp theo đang chờ sẽ được chuyển tới máy in

Hàng đợi

- Ví dụ về hàng đợi
- **Cấu trúc dữ liệu hàng đợi**
- Cài đặt hàng đợi dùng danh sách liên kết
- Các thao tác hàng đợi
  - enqueue()
  - dequeue()
  - peek()
  - isEmptyQueue()
- Ví dụ ứng dụng

Mảng, danh sách liên kết, hàng đợi

- Mảng
  - Cấu trúc dữ liệu truy cập ngẫu nhiên
  - Truy cập trực tiếp bất kì phần tử nào của mảng
    - array[index]
- Danh sách liên kết
  - Cấu trúc dữ liệu truy cập tuần tự
  - Để truy cập một phần tử phải đi qua các phần tử trước nó
    - cur->next
- Hàng đợi
  - Cấu trúc dữ liệu tuần tự truy cập có giới hạn

Cấu trúc dữ liệu hàng đợi

- Hàng đợi là một cấu trúc dữ liệu hoạt động như hàng đợi gặp trong cuộc sống
  - Ví dụ: Hàng đợi để sử dụng máy ATM, thanh toán
  - Các phần tử chỉ có thể thêm vào cuối hàng đợi
  - Các phần tử chỉ có thể được lấy ra từ đầu hàng đợi
- Nguyên tắc: First-In, First-Out (FIFO)
  - hoặc Last-In, Last-Out (LILO)
- Hàng đợi thường được xây dựng dựa trên các cấu trúc dữ liệu khác
  - Mảng, danh sách liên kết
  - Bài giảng tập trung vào cài đặt hàng đợi dựa trên danh sách liên kết

Cấu trúc dữ liệu hàng đợi

- Thao tác chính
  - Enqueue: Thêm vào một phần tử ở cuối hàng đợi
  - Dequeue: Lấy ra một phần tử ở đầu hàng đợi
- Thao tác hỗ trợ
  - Peek: Xem một phần tử ở đầu hàng đợi nhưng không lấy ra khỏi hàng đợi
  - IsEmptyQueue: Kiểm tra xem hàng đợi có phần tử nào không
- Các hàm tương ứng
  - enqueue()
  - dequeue()
  - peek()
  - isEmptyQueue()
- Trong các ví dụ cài đặt hàng đợi sẽ giả sử xử lý với số nguyên
  - Nhưng với danh sách liên kết, có thể xử lý bất kì dữ liệu nào

Hàng đợi

- Ví dụ về hàng đợi
- Cấu trúc dữ liệu hàng đợi
- **Cài đặt hàng đợi dùng danh sách liên kết**
- Các thao tác hàng đợi
  - enqueue()
  - dequeue()
  - peek()
  - isEmptyQueue()
- Ví dụ ứng dụng

Cài đặt hàng đợi sử dụng danh sách liên kết

- Chúng ta định nghĩa cấu trúc dữ liệu danh sách liên kết

```
typedef struct _linkedlist {
    ListNode *head;
    int size;
} LinkedList;
```

- Queue xây dựng dựa trên danh sách liên kết

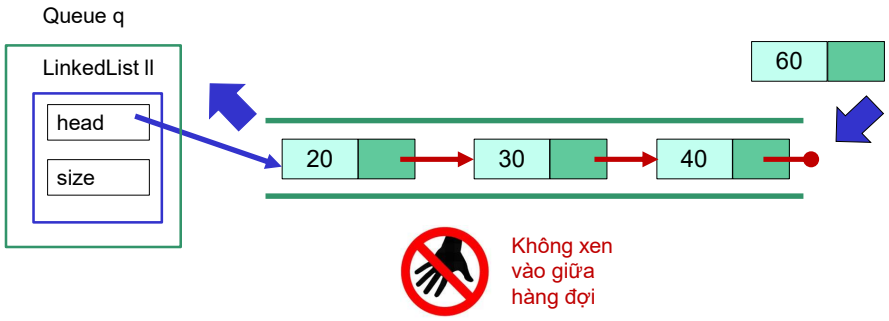
```
typedef struct _queue {
    LinkedList ll;
} Queue;
```

Cài đặt hàng đợi sử dụng danh sách liên kết

- Cấu trúc Queue

```
typedef struct _queue {
        LinkedList ll;
} Queue;
```

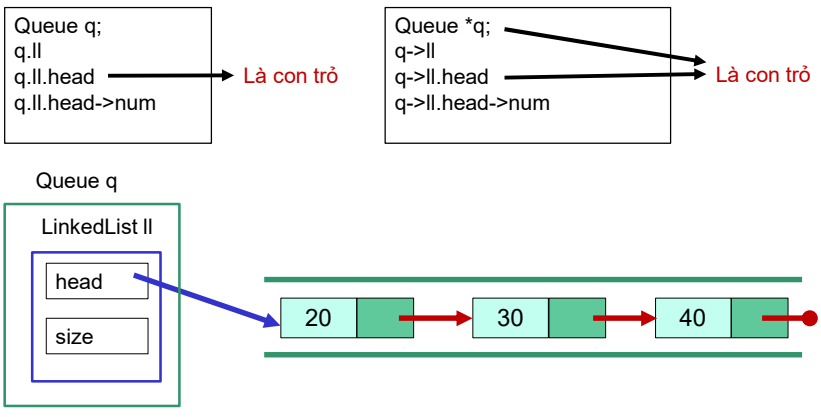
- Sử dụng danh sách liên kết để chứa dữ liệu
- So với danh sách liên kết, thao tác trong hàng đợi có thay đổi



Cài đặt hàng đợi sử dụng danh sách liên kết

- Cấu trúc Queue

```
typedef struct _queue {
        LinkedList ll;
} Queue;
```

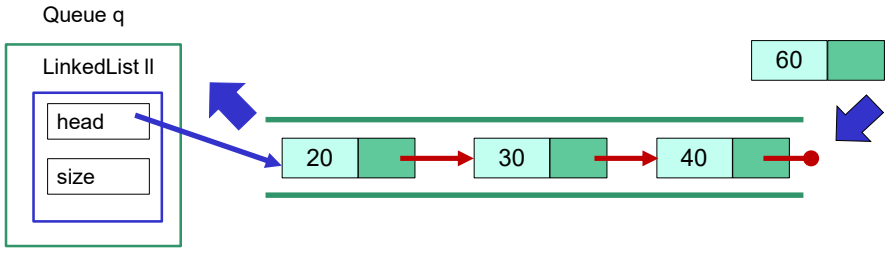


Hàng đợi

- Ví dụ về hàng đợi
- Cấu trúc dữ liệu hàng đợi
- Cài đặt hàng đợi dùng danh sách liên kết
- **Các thao tác hàng đợi**
  - enqueue()
  - dequeue()
  - peek()
  - isEmptyQueue()
- Ví dụ ứng dụng

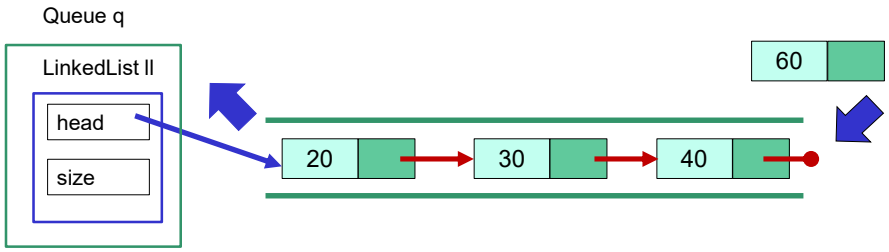
enqueue()

- enqueue() là cách duy nhất để thêm một phần tử vào cấu trúc dữ liệu hàng đợi
- enqueue() chỉ cho phép thêm một phần tử vào cuối hàng đợi
- Nút đầu tiên của danh sách liên kết là đầu của hàng đợi (hoặc nếu chọn là cuối của hàng đợi thì thay đổi code)



enqueue()

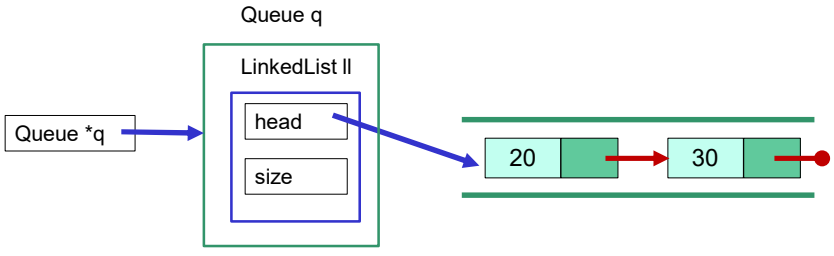
- Viết hàm enqueue()
  - Khai báo prototype
  - Cài đặt hàm
- Yêu cầu
  - Sử dụng các hàm cho LinkedList đã cài đặt
  - Chỉ thêm vào cuối hàng đợi



enqueue()

```
void insertNode(ListNode **ptrHead, int index, int value);
void enqueue(Queue *q, int item) {
    insertNode(&(q->ll.head), q->ll.size, item);
}
```

Thêm một phần tử vào hàng đợi -> thêm một nút vào cuối của danh sách liên kết



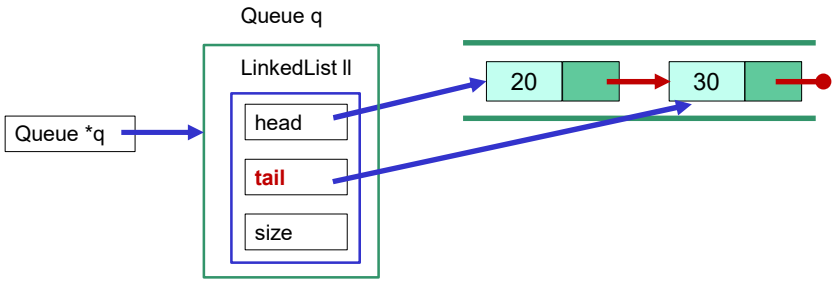
enqueue()

```
void enqueue(Queue *q, int item) {
    insertNode(&(q->ll.head), q->ll.size, item);
}
```

Không hiệu quả nếu hàng đợi dài

Độ phức tạp O(n)

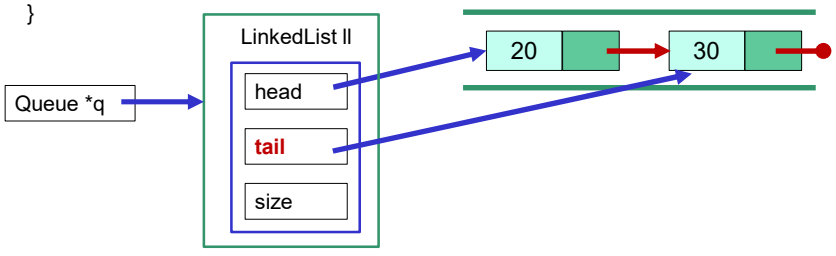
- Cần sử dụng thêm một **con trỏ tail** để thao tác hiệu quả
  - Con trỏ tail chỉ tới phần tử cuối cùng của danh sách liên kết



enqueue()

```
void enqueue(Queue *q, int item) {
    if (q->ll.tail==NULL) {
        insertNode(&(q->ll.head), 0, item);
        q->ll.tail=q->ll.head;
    } else {
        q->ll.tail->next=malloc(…);
        q->ll.tail=q->ll.tail->next;
        q->ll.tail->num = item;
        q->ll.tail->next=NULL;
        q->ll.size ++;
    }
}
```

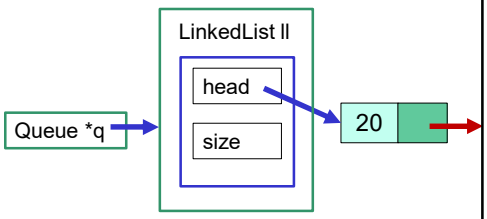
Độ phức tạp O(1)



dequeue()

- Thao tác lấy một phần tử ra khỏi hàng đợi gồm 2 bước
  - Lấy giá trị của nút ở đầu hàng đợi
  - Xóa nút đó trong danh sách liên kết
- Yêu cầu
  - Sử dụng các hàm cho LinkedList đã cài đặt
  - Chỉ lấy ra ở đầu hàng đợi

```
int dequeue(Queue *q) {
    int item;
    If (q->ll.head!=NULL) {
        item = ((q->ll).head)->num;
        removeNode(&(q->ll).head, 0);
        (q->ll).size--;
        return item;
    }
    else return NULL_VALUE;
}
```



- Cần biến item để chứa giá trị vì không thể lấy giá trị sau khi loại bỏ phần tử
- Độ phức tạp O(1)

peek()

- Không tạo sự thay đổi gì trong hàng đợi
- Lấy giá trị của nút ở đầu hàng đợi
  - Lấy giá trị của nút đầu tiên của danh sách liên kết
  - Không xóa nút này
- Độ phức tạp O(1)

```
int peek(Queue *q){
    if (q->ll.head!=NULL)
        return (q->ll).head->num;
    else return NULL_VALUE;
}
```

isEmplyQueue()

- Kiểm tra xem số phần tử trong hàng đợi có phải bằng 0 không
- Sử dụng biến size trong cấu trúc LinkedList
- Độ phức tạp O(1)

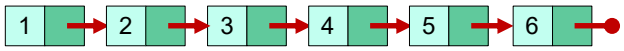
```
int isEmptyQueue(Queue *q){
    if ((q->ll).size == 0) return 1;
    return 0;
}
```

Hàng đợi

- Ví dụ về hàng đợi
- Cấu trúc dữ liệu hàng đợi
- Cài đặt hàng đợi dùng danh sách liên kết
- Các thao tác hàng đợi
  - enqueue()
  - dequeue()
  - peek()
  - isEmptyQueue()
- **Ví dụ ứng dụng**

Ứng dụng đơn giản kiểm tra hoạt động của Queue

- Ứng dụng đơn giản
  - Thêm vào hàng đợi một vài số nguyên
  - Lấy các phần tử trong hàng đợi và đưa giá trị của các phần tử đó ra màn hình




```
int main() {
    Queue q;
    q.ll.head = NULL;
    q.ll.tail = NULL;
    q.ll.size = 0;
    enqueue(&q, 1);
    enqueue(&q, 2);
    enqueue(&q, 3);
    enqueue(&q, 4);
    enqueue(&q, 5);
    enqueue(&q, 6);
    while (!isEmptyQueue(&q))
        printf("%d ", dequeue(&q));
}
```

Chương 2: Mảng và danh sách liên kết

- Cấu trúc lưu trữ mảng
- Danh sách liên kết
- Hàng đợi
- Ngăn xếp

Cấu trúc dữ liệu và giải thuật

- Nội dung bài giảng được biên soạn bởi PGS. TS. Phạm Tuấn Minh.

1-27