Cấu trúc dữ liệu và Giải thuật
Ngăn Xếp và Hàng Đợi

Bùi Ngọc Thăng

October 14, 2015

Outline

Mục tiêu bài học

1. Tìm hiểu khái niệm ngăn xếp (Stack) và hàng đợi (Queue)
2. Hiểu và cài đặt cấu trúc dữ liệu ngăn xếp và hàng đợi
3. Cài đặt được các phép toán trên ngăn xếp và hàng đợi

Ngăn xếp

1. Khái niệm ngăn xếp
2. Các phép toán trên ngăn xếp
3. Cài đặt ngăn xếp và các phép toán (C/C++)

Outline

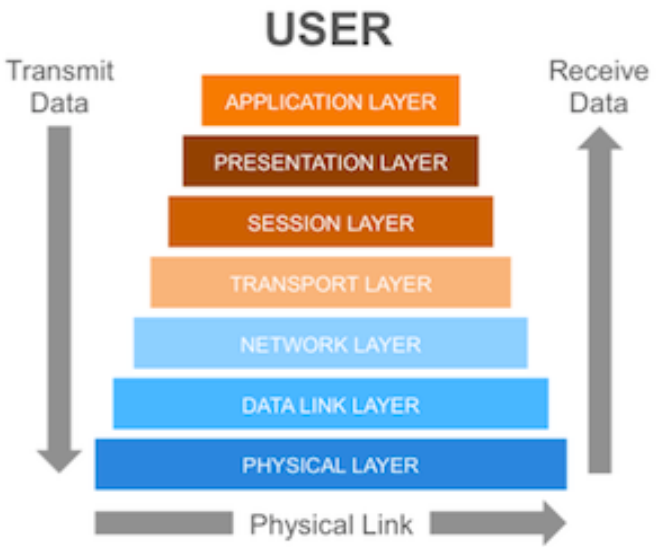
Ngăn xếp

Definition

Ngăn xếp: Là một cấu trúc dữ liệu lưu trữ và quản lý một tập các phần tử theo cơ chế LIFO (Last In First Out– Vào sau ra trước): Chèn một phần tử vào đỉnh ngăn xếp (Push) và loại một phần tử ở đỉnh của ngăn xếp (Pop).

Ngăn xếp: Các ứng dụng

1. Lưu trữ chỗng đĩa, chồng sách, các tệp văn bản; phục hồi các xâu ký tự trong một môi trường soạn thảo.
2. Tính toán các biểu thức toán, đệ quy, quy lui.
3. Tương tác giữa các tầng mạng.



Outline

Cài đặt cấu trúc dữ liệu ngăn xếp

I. Sử dụng mảng để cài đặt ngăn xếp

```
const int max =...// Số lượng tối đa các phần tử mà danh sách có thể lưu trữ
struct Stack{
        item element[max];// mảng lưu các phần tử có kiểu item
        int top;// Lưu chỉ số của phần tử nằm ở đầu (đỉnh) ngăn xếp
} ;
```

Cài đặt cấu trúc dữ liệu ngăn xếp

II. Cài đặt ngăn xếp bởi danh sách liên kết

```
struct StackNode{
        item data;//thành phần lưu dữ liệu
        StackNode *next;//Lưu địa chỉ của phần tử tiếp theo
};
struct Stack{
        StackNode *top;//Con trỏ quản lý địa chỉ nhớ của phần
tử ở đầu ngăn xếp
};
```

Outline

Các phép toán trên ngăn xếp

1. Khởi tạo ngăn xếp
2. Kiểm tra ngăn xếp rỗng hoặc đầy
3. Lấy thông tin phần tử ở đỉnh ngăn xếp
4. Chèn một phần tử ở đỉnh ngăn xếp
5. Loại một phần tử ở đỉnh ngăn xếp
6. Duyệt danh sách

Khởi tạo ngăn xếp

Số lượng phần tử của ngăn xếp được gán bằng 0 (count=0)

Mã nguồn:

```
void Initialization(Stack &S){
        S.top=0;
}
void Initialization(Stack *S){
        S→top=0;
}
```

Kiểm tra ngăn xếp rỗng hoặc đầy

I. Kiểm tra ngăn xếp rỗng:
Input: Một ngăn xếp S
Output: Nhận 02 giá trị: true– Ngăn xếp rỗng; false– ngăn xếp không rỗng.
Mã nguồn:

```
bool isEmpty(Stack S){
        if(S.top==0) return true;
        else return false;
}
```

Kiểm tra ngăn xếp rỗng hoặc đầy

II. Kiểm tra ngăn xếp đầy:
Input: Một ngăn xếp S
Output: Nhận 02 giá trị: true– Ngăn xếp đầy; false– Ngăn xếp không đầy.
Mã nguồn:

```
bool isFull(Stack S){
        if(S.top==max) return true;
        else return false;
}
```

Lấy thông tin phần tử ở đỉnh ngăn xếp

Input: Một ngăn xếp S
Output: trả về phần tử tại đỉnh ngăn xếp nếu ngăn xếp không rỗng và thông báo lấy thành công (hàm trả về giá trị **true**); hàm trả về giá trị **false** nếu ngẫn xếp rỗng
Mã nguồn:

```
bool GetTop(Stack S, item &x){
        if((isEmpty(S)==true)) return false;
        x=S.element[top]; return true;
}
```

Loại phần tử ở đỉnh ngăn xếp

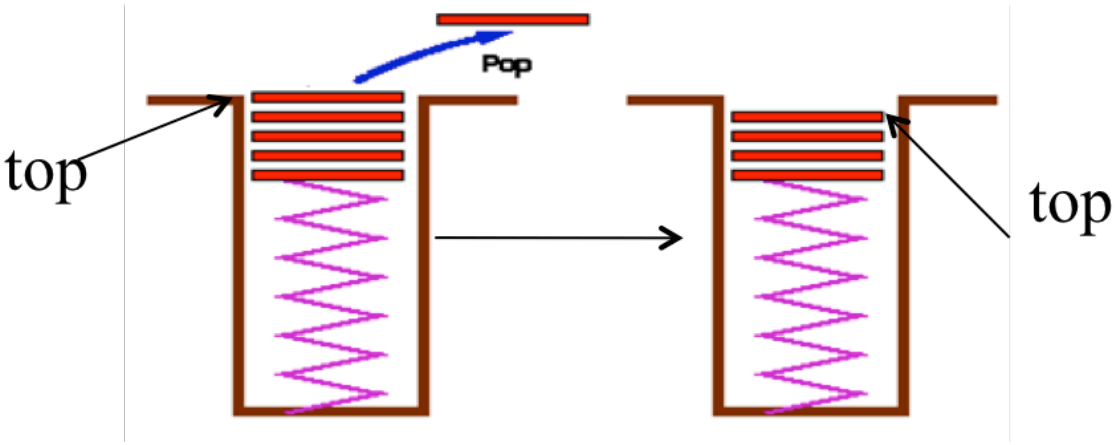


Figure: Loại phần tử ở đỉnh ngăn xếp

Input: Một danh sách S
Output: Một danh sách S sau khi loại bỏ phần tử ở đỉnh ngăn xếp
Phương pháp:
Bước 1: Kiểm tra S có rỗng hay không, nếu không thực hiện
Bước 2.
Bước 2: Lưu lại phần tử ở đỉnh. Giảm số lượng phần tử (top)

Loại phần tử ở đỉnh ngăn xếp

Input: Một danh sách S
Output: Một danh sách S sau khi loại bỏ phần tử ở đỉnh ngăn xếp
Mã nguồn:

```
bool Pop(Stack &S, item &x){
        if((isEmpty(S)==true)) return false;
        x= S.element[top];
        S.top=S.top-1; return true;
}
```

Chèn một phần tử vào vị trí p

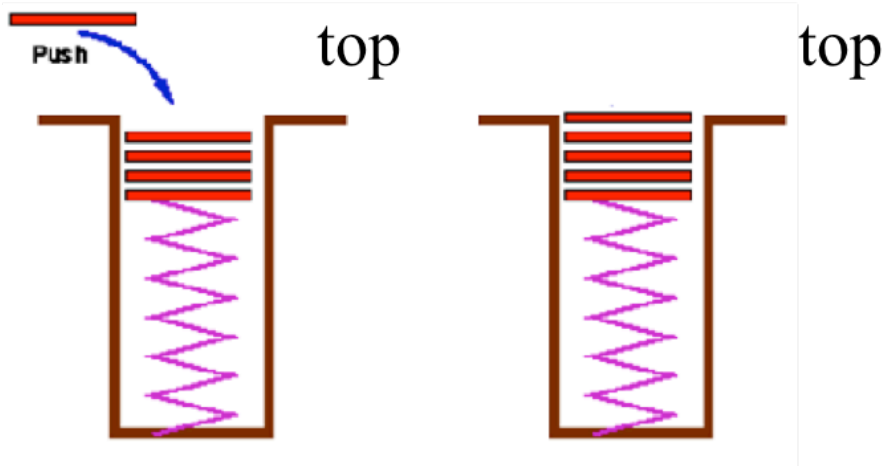


Figure: Chèn một phần tử ở đỉnh ngăn xếp

Input: Một danh sách S, một phần tử x, và phần tử x cần chèn vào đỉnh

Output: Một danh sách S sau khi chèn x

Chèn một phần tử vào vị trí p

Input: Một danh sách S, một phần tử x, và phần tử x cần chèn vào đỉnh
Output: Một danh sách S sau khi chèn x
Phương pháp:
Bước 1: Kiểm tra S có đầy hay không, nếu không thực hiện **Bước 2**.
Bước 2: Tăng giá trị của *top* lên một đơn vị.
Bước 3: Gán thông tin của x tại cho phần tử tại vị trí *top* trong mảng *element*

Danh sách: Các phép toán trên danh sách

1. Khởi tạo danh sách
2. Kiểm tra danh sách rỗng hoặc đầy
3. Lấy thông tin phần tử trong danh sách tại vị trí p
4. Thay đổi thông tin của phần tử tại vị trí p
5. Loại một phần tử tại vị trí p
6. Chèn một phần tử ở vị trí p
7. Duyệt danh sách

Outline

Độ phức tạp của các phép toán trên ngăn xếp

Table: Độ phức tạp của các phép toán

Phép toán	**Độ phức tạp**
Khởi tạo danh sách	$\mathcal{O}(1)$
Kiểm tra danh sách rỗng hoặc đầy	$\mathcal{O}(1)$
Lấy thông tin phần tử ở đầu ngăn xếp	$\mathcal{O}(1)$
Chèn phần tử vào đầu ngăn xếp	$\mathcal{O}(1)$
Loại phần tử đầu ngăn xếp	$\mathcal{O}(1)$
Duyệt ngăn xếp	$\mathcal{O}(n)$

Ngăn xếp và hàng đợi: Các vấn đề còn lại

- Hàng đợi và các phép toán trên hàng đợi (buổi tiếp theo)
- Đánh giá độ phức tạp của các phép toán trên hàng đợi (buổi tiếp theo)

Thanks for your attention!