Cấu trúc dữ liệu và giải thuật

Chương 2: Các cấu trúc tuyến tính

Phần 2: Cấu trúc Danh Sách

Các nội dung chính

- Cấu trúc danh sách
  - Mô tả
  - Cấu trúc vào sau ra trước (LIFO) (Stack-Ngăn xếp)
  - Cấu trúc vào trước ra trước (FIFO) (Queue-Hàng đợi)
  - Một số ứng dụng của ngăn xếp và hàng đợi

Giới thiệu - *Mô tả cấu trúc*

- Danh sách tuyến tính: CTDL gồm một hay nhiều phần tử cùng kiểu dữ liệu và tồn tại một trật tự tuyến tính giữa các phần tử.
- Kí hiệu: $L = <x_1,x_2,...,x_n>$
  - $n \geq 1$ và $x_1,x_2,...,x_n$ là các phần tử của danh sách,
  - x_1 được gọi là phần tử đầu tiên (đầu) của danh sách
  - x_n được gọi là phần tử cuối cùng (đuôi) của danh sách
- Trật tự tuyến tính: trật tự trước-sau giữa các phần tử, tức là với mọi cặp phần tử $<x_i,x_j>$ ($1 \leq i,j \leq n$ và $i \neq j$) trong tập các phần tử này luôn có duy nhất một trật tự trước sau.
- Quy ước: trường hợp đặc biệt khi danh sách không có phần tử nào, gọi là danh sách rỗng, kí hiệu ∅ ($L = \emptyset$).

Giới thiệu – *Đặc trưng:*

- Kích thước hay độ dài danh sách: số phần tử của danh sách. Kích thước của danh sách rỗng bằng 0.
  - Chú ý kích thước danh sách không cố định mà biến đổi trong quá trình xử lý, thao tác và nó là đại lượng mà ta thường không biết trước được.
- Kiểu dữ liệu của các phần tử: có một kiểu dữ liệu duy nhất cho các phần tử của danh sách. Kiểu dữ liệu cho các phần tử luôn luôn cố định.
- Trật tự tuyến tính trong danh sách: một danh sách luôn có hai phía, một phía chúng ta quy ước là đầu, còn phía kia là đuôi của danh sách. Trật tự trước-sau là trật tự từ đầu đến cuối.

Giới thiệu – *Các thao tác cơ bản trên danh sách*

- **Khởi tạo danh sách**
  - Định ra cấu trúc danh sách, xác định các đặc trưng của danh sách.
  - Sau thao tác khởi tạo, ta thường thu được một danh sách rỗng (danh sách chưa có nội dung, mà mới chỉ có phần khung).
  - Trong các ngôn ngữ lập trình, thao tác khởi tạo thường là việc khai báo cấu trúc lưu trữ thích hợp để biểu diễn cho cấu trúc danh sách yêu cầu.

Giới thiệu – *Các thao tác cơ bản trên danh sách*

- **Bổ sung một phần tử mới vào danh sách**
  - Trước tiên cần phải xác định vị trí trong danh sách mà phần tử mới sẽ được đưa vào.
  - Thông thường, vị trí bổ sung thường là đầu hay cuối của danh sách. Tuy nhiên, ta cũng có thể chèn phần tử mới vào giữa danh sách.
  - Mỗi thao tác bổ sung này làm tăng kích thước danh sách lên 1.
  - Chú ý, điều kiện trước tiên (hay tiền điều kiện) để thực hiện được thao tác bổ sung là danh sách chưa "*đầy*" hay chưa bão hoà.

Giới thiệu – *Các thao tác cơ bản trên danh sách*

- **Loại bỏ một phần tử khỏi danh sách**
  - Đây là thao tác ngược lại của thao tác bổ sung.
  - Việc loại bỏ này làm giảm kích thước của danh sách đi 1.
  - Chú ý là điều kiện trước tiên để loại bỏ một phần tử là danh sách phải không rỗng, tức là nó phải còn ít nhất một phần tử.
  - Nói chung, việc xác định trạng thái rỗng của danh sách là một công việc đơn giản, vì đây cũng chính là trạng thái của danh sách sau thao tác khởi tạo.

Giới thiệu – *Các thao tác cơ bản trên danh sách*

- **Tìm kiếm một phần tử trong danh sách:** tìm kiếm sự xuất hiện hay không của một hay một số phần tử trong danh sách theo một điều kiện tìm kiếm, và nếu xuất hiện thì ở vị trí nào trong danh sách. Thao tác tìm kiếm thường đi đôi với các thao tác bổ sung và loại bỏ. Chúng ta sẽ dành một chương "*các giải thuật tìm kiếm*" để nói về thao tác này.
- **Sắp xếp danh sách:** là thao tác sắp xếp các phần tử của danh sách theo một trật tự nhất định Các giải thuật sắp xếp sẽ được trình bầy chi tiết trong chương "*Các giải thuật sắp xếp*"
- **Nối hai hay nhiều danh sách con** để thành một danh sách
- **Tách danh sách** thành hai hay nhiều danh sách con

Giới thiệu – *Các cấu trúc danh sách thông dụng*

- Cấu trúc vào sau ra trước (Last In, First Out - LIFO) hay cấu trúc ngăn xếp – Stack
- Cấu trúc vào trước ra trước (First In, First Out - FIFO) hay cấu trúc *hàng đợi* – Queue

Giới thiệu – *Các cấu trúc danh sách thông dụng*

- Cấu trúc *hàng đợi có ưu tiên* - Priority queue
  - Ở đây mỗi phần tử có thêm một thuộc tính gọi là độ ưu tiên.
  - Với các phần tử có độ ưu tiên như nhau thì danh sách hoạt động giống như nguyên tắc hàng đợi.
  - Khi có hai phần tử bất kỳ có độ ưu tiên khác nhau thì danh sách luôn ưu tiên phần tử có độ ưu tiên cao hơn, nên dù phần tử có độ ưu tiên cao hơn có được bổ sung vào sau phần tử có độ ưu tiên thấp hơn thì nó vẫn được loại bỏ ra trước. Có thể nói cấu trúc này kết hợp cả hai nguyên tắc hoạt động của ngăn xếp và hàng đợi không ưu tiên.

Giới thiệu – *Các phương pháp cài đặt*

- Cài đặt bằng cấu trúc lưu trữ tuần tự (hay mảng một chiều): đây là cách cài đặt đơn giản và nhanh nhất. Tuy nhiên, cách cài đặt này cũng có một số hạn chế do sự khác nhau cơ bản giữa cấu trúc danh sách và cấu trúc mảng, giữa một bên là cấu trúc động, một bên là cấu trúc tĩnh.
- Cài đặt bằng cấu trúc lưu trữ móc nối: đây là một cấu trúc lưu trữ động với kích thước và tổ chức lưu trữ có thể biến đổi linh hoạt theo yêu cầu của cấu trúc dữ liệu, nên rất thích hợp để tổ chức và lưu trữ các cấu trúc dữ liệu động.

Nguyên tắc chung

- Khi sử dụng một CTLT để cài đặt cho một cấu trúc dữ liệu, chúng ta phải lưu ý một số nguyên tắc sau:
  - *Tính đầy đủ*: CTLT phải có các đặc trưng thích hợp để biểu diễn đầy đủ các đặc tính và khả năng của CTDL. Nó thể hiện ở hai khía cạnh
    - **Khả năng lưu trữ:** là kích thước của CTLT được cài đặt. Đại lượng này được quyết định bởi miền giá trị của cấu trúc dữ liệu mà nó cài đặt. Việc cài đặt cấu trúc dữ liệu kiểu số nguyên là một ví dụ sinh động về nguyên tắc cài đặt này.
    - **Khả năng xử lý:** được cụ thể hoá bằng tập các thao tác được cài đặt trên CTLT đã chọn.
  - *Tính tối ưu*: hiệu quả cao nhất về lưu trữ và xử lý.
  - *Tính mở*
  - *Tính che dấu*

Cài đặt danh sách bằng cấu trúc lưu trữ tuần tự

- Nguyên tắc chung:
  - Chọn cấu trúc lưu trữ phù hợp
  - Bố trí hợp lý các phần tử của danh sách trong CTLT đã chọn

Cài đặt danh sách bằng cấu trúc lưu trữ tuần tự

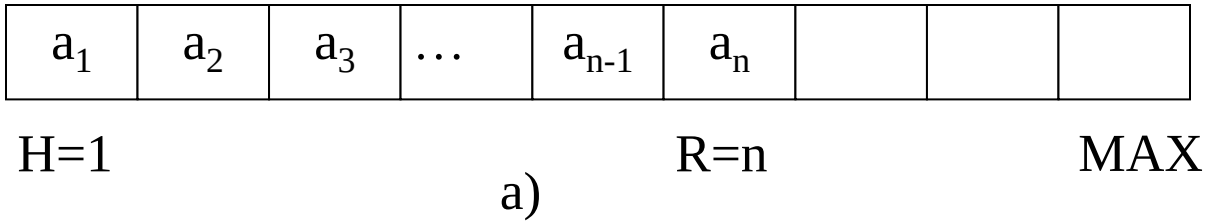
- Nguyên tắc chung:
  - Chọn cấu trúc lưu trữ phù hợp
    - Xác định kích thước tối đa MAX của danh sách
    - Xác định kiểu dữ liệu cho mỗi phần tử của danh sách
    - Chọn CTLT là mảng một chiều kích thước và kiểu dữ liệu như đã chọn

Cài đặt danh sách bằng cấu trúc lưu trữ tuần tự

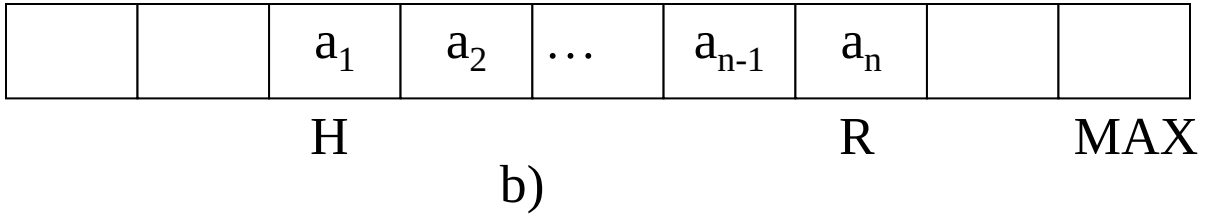
- Nguyên tắc chung:
  - Bố trí hợp lý các phần tử của danh sách trong CTLT đã chọn theo một số nguyên tắc:
    - Bố trí mỗi phần tử trong một ngăn nhớ
    - Bố trí các phần tử lần lượt theo trật tự tuyến tính của chúng
    - Bố trí các phần tử ở các vị trí kề nhau để đảm bảo số thông tin cần quản lý danh sách là ít nhất.
    - Chọn cách bố trí thích hợp để các thao tác cơ bản thực hiện trên danh sách là hiệu quả nhất.

Cài đặt danh sách bằng cấu trúc lưu trữ tuần tự

- Một số cách bố trí các phần tử của danh sách trong mảng
  a) Bố trí liền từ đầu mảng H=1, R=n, chỉ cần một thông tin là n
  b) Bố trí dãy liên tục, cần hai trong ba thông tin H, R, n.



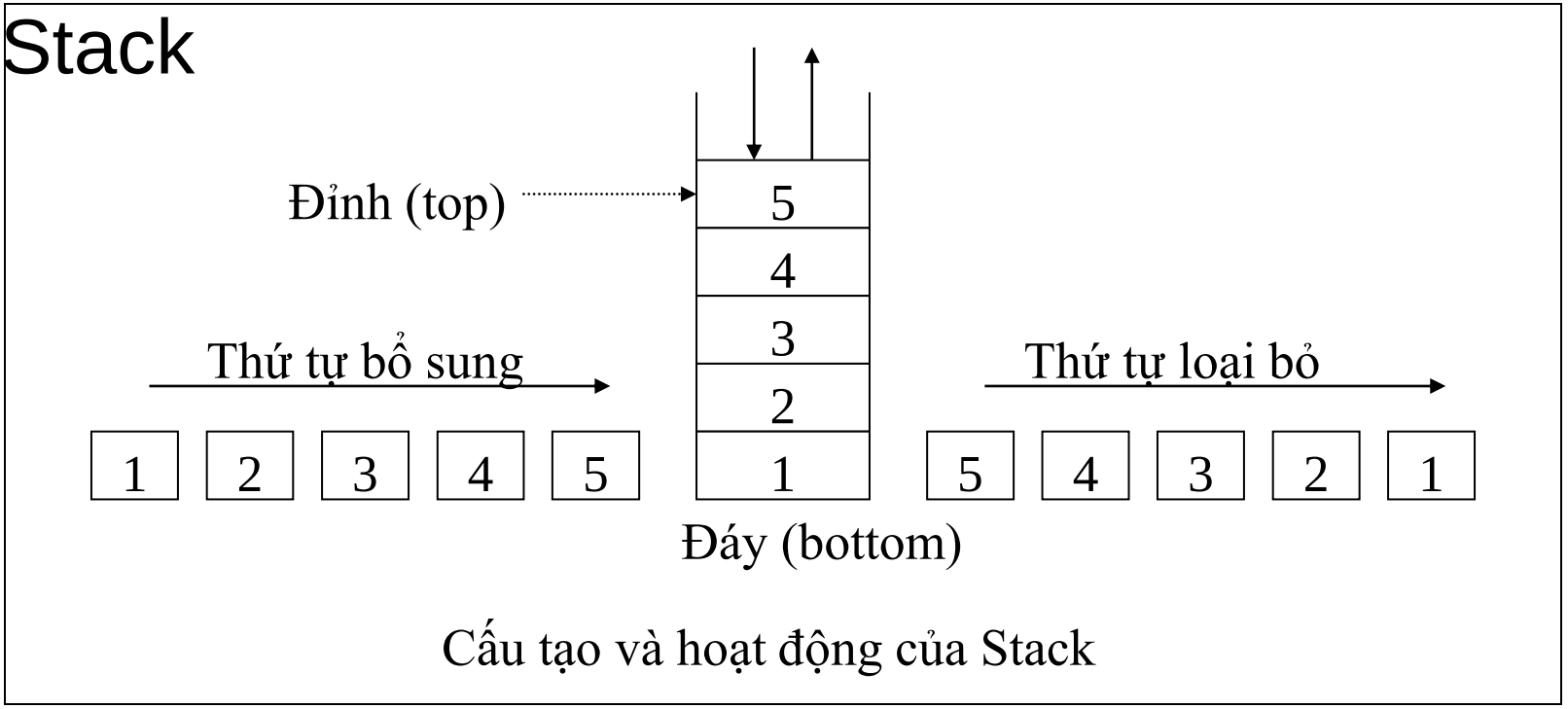
a)



b)

Cấu trúc LIFO (Stack)

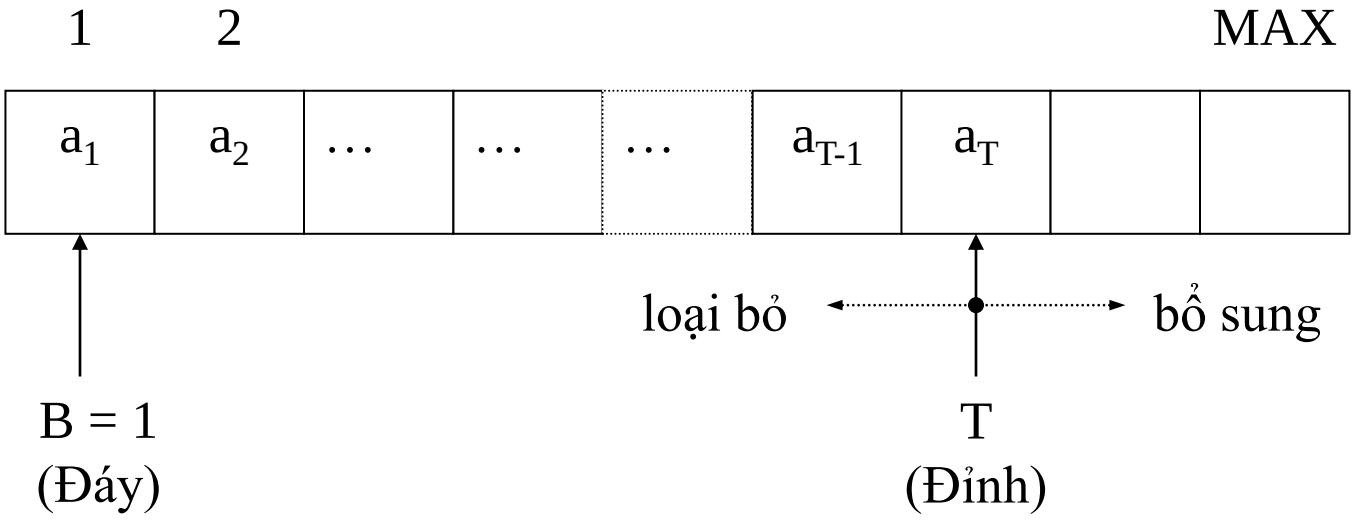
- Mô tả: cấu trúc vào sau ra trước (Last In, First Out - LIFO) hay cấu trúc ngăn xếp – Stack



Cấu tạo và hoạt động của Stack

Cài đặt Stack bằng cấu trúc lưu trữ tuần tự

- Nguyên tắc cài đặt



Biểu diễn ngăn xếp bằng CTLT tuần tự

```
typedef  struct  {
   Type  info [MAX];
   unsigned int n;            //Số phần tử của Stack
} Stack;
```

- CTLT tuần tự được thể hiện qua mảng *info*,
- *Type* là kiểu dữ liệu của danh sách
- n là số phần tử của ngăn xếp.
- Đỉnh của ngăn xếp sẽ ở vị trí n-1

```
typedef  struct  {
   Type  info [MAX];
   unsigned int n;            //Số phần tử của Stack
} Stack;
```

- Trạng thái hiện thời của ngăn xếp:
  - Rỗng (*empty*): n=0
  - Đầy (*full*): thông thường, giá trị MAX là kích thước tối đa của ngăn xếp (nếu có thể xác định chính xác) hoặc kích thước tối đa mà CTLT tuần tự này được cho phép, nên ta sẽ quy ước ngăn xếp đầy khi n=MAX.
  - Bình thường (*normal*): trạng thái không rỗng mà cũng không đầy. Khi ngăn xếp ở trạng thái này, nó có thể thực hiện các thao tác bổ sung và loại bỏ.

- Các thao tác cơ bản:

```
void Initialize (Stack & S){
    S.n =0;
}
```

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

```
bool IsFull (Stack S){
    if (S.n == MAX) return true;
    else return false;
}
```

- Các thao tác cơ bản:

```
//Bổ sung phần tử K vào đỉnh của ngăn xếp
void Push (Type K, Stack & S){
    if (IsFull(S)) return;
    S.info[S.n] = K;
    S.n++;
}
```

```
//Lấy ra phần tử ở đỉnh của ngăn xếp
Type Pop (Stack & S){
    if (IsEmpty(S)) return NULL;
    S.n--;
    return S.info[S.n];
}
```

- Các thao tác cơ bản:

```
//Trả về phần tử ở đỉnh của ngăn xếp (không lấy phần tử đó ra)
Type Top (Stack  S){
      if (IsEmpty(S)) return NULL;
      return S.info[S.n - 1];

}
```

Bài tập

- **Bài 9.1**: Cài đặt cấu trúc FIFO (Queue) bằng cấu trúc lưu trữ tuần tự
- **Bài 9.2**: Cài đặt danh sách tổng quát bằng CTLT tuần tự (với danh sách tổng quát, việc bổ sung và lấy ra một phần tử có thể thực hiện ở một vị trí bất kỳ trong danh sách)
- **Bài 9.3**: Viết chương trình chuyển đổi 1 số từ cơ số 10 sang cơ số 2, mà có sử dụng cấu trúc Stack.

Thank you!