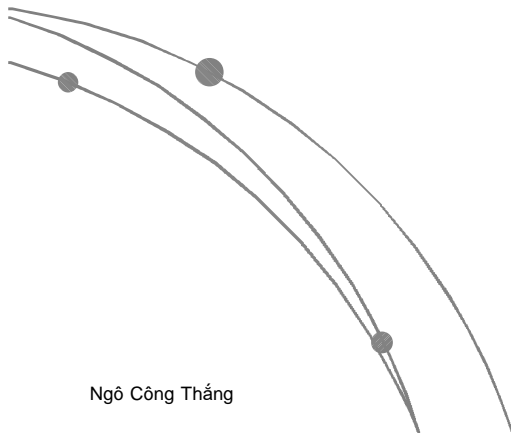


CHƯƠNG 2

MẢNG VÀ DANH SÁCH

1. Mảng

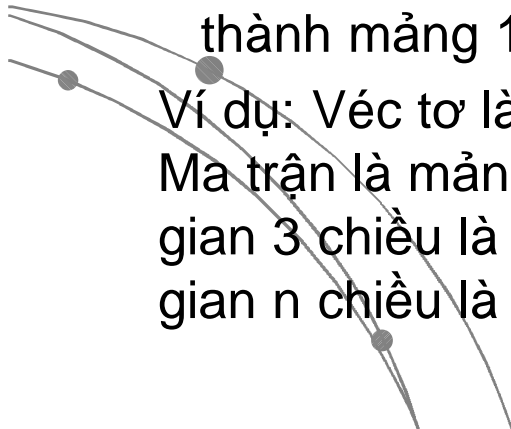
2. Danh sách



1. Mảng

- | Mảng là một tập hợp có thứ tự gồm một số cố định các phần tử cùng kiểu.
- | Một phần tử mảng được chỉ ra bởi chỉ số, thể hiện thứ tự của phần tử trong mảng.
- | Các phần tử của mảng có thể được tổ chức thành mảng 1 chiều, 2 chiều, 3 chiều...

Ví dụ: Véc tơ là mảng 1 chiều có 1 chỉ số (i).
Ma trận là mảng 2 chiều có 2 chỉ số (i, j). Không gian 3 chiều là mảng 3 chiều có 3 chỉ số. Không gian n chiều là mảng n chiều có n chỉ số.



1. Mảng

- | Mảng chỉ dùng được cấu trúc lưu trữ kế tiếp, để cho phép truy nhập trực tiếp các phần tử.
- | Dùng vec tơ lưu trữ V có n ô nhớ liên tiếp với chỉ số từ 1 đến n để lưu trữ các phần tử dữ liệu của mảng.
- | Với mảng 1 chiều, phần tử a_i được lưu trữ ở ô nhớ $V[i]$
- | Với mảng 2 chiều, các phần tử được lưu trữ lần lượt, hết hàng 1 đến hàng 2... Phần tử a_{ij} được lưu trữ ở ô nhớ $V[k]$, $k = (i-1)*n + j$

1. Mảng

- | Mảng 2 chiều có $m = 2$ hàng, $n = 3$ cột. Tính chỉ số k truy nhập vào ô nhớ chứa phần tử a_{ij} .

4 5 9

7 10 1

4 5 9 7 10 1 $\Rightarrow k = (i-1)*n + j$

- | Có các phép tạo lập mảng, tìm kiếm 1 phần tử từ mảng, truy nhập một phần tử mảng.
- | Không có phép bổ sung hoặc loại bỏ một phần tử mảng.

2. Danh sách

2.1. Khái niệm

- I Danh sách là một tập hợp có thứ tự gồm một số biến động các phần tử cùng kiểu.
- I Phép loại bỏ, bổ sung 1 phần tử là phép thường xuyên tác động lên danh sách.
- I Ví dụ: Tập hợp người đến khám bệnh cho ta một danh sách. Người đến xếp hàng khám bổ sung ở phía sau, người được khám sẽ ra khỏi hàng (loại bỏ) ở phía trước.

2.1. Khái niệm

- I Danh sách tuyến tính: Một danh sách mà quan hệ lân cận giữa các phần tử được xác định rõ ràng thì được gọi là danh sách tuyến tính. Véc tơ là một danh sách tuyến tính.
- I Danh sách tuyến tính hoặc rỗng (không có phần tử nào) hoặc có dạng (a_1, a_2, \dots, a_n) với $a_i, 1 \leq i \leq n$ là các phần tử.
- I Trong danh sách tuyến tính tồn tại phần tử đầu là a_1 , phần tử cuối là a_n , phần tử thứ i là a_i . Với a_i bất kỳ $1 \leq i \leq n$ thì a_{i+1} gọi là phần tử sau a_i ; $2 \leq i \leq n$ thì phần tử a_{i-1} là phần tử trước của a_i .

2.1. Khái niệm

- | n là độ dài (kích thước) của danh sách, n có thể thay đổi.
- | Một phần tử trong danh sách thường là một bản ghi (gồm một hoặc nhiều trường).
- | Ví dụ 1: Danh mục điện thoại là một danh sách tuyến tính, mỗi phần tử của nó là một thuê bao gồm 3 trường: Họ tên chủ hộ, địa chỉ, số điện thoại.
- | Ví dụ 2: Tập(File) là một danh sách có kích thước lớn được lưu trữ ở bộ nhớ ngoài.

2.2. Các phép toán trên danh sách

- | Phép bổ sung: Có thể bổ sung phần tử vào danh sách.
- | Phép loại bỏ: có thể loại bỏ một phần tử ra khỏi danh sách.
- | Phép ghép: có thể ghép hai hay nhiều danh sách thành một danh sách.
- | Phép tách: có thể tách một danh sách thành nhiều danh sách.
- | Phép cập nhật: cập nhật giá trị cho các phần tử của danh sách.

2.2. Các phép toán trên danh sách

- | Phép sao chép: có thể sao chép một danh sách.
- | Phép sắp xếp: Có thể sắp xếp các phần tử của danh sách theo một thứ tự nhất định.
- | Phép tìm kiếm: Tìm kiếm trong danh sách một phần tử mà một trường nào đó có giá trị ấn định.

Ví dụ 1: Minh họa cho các phép toán trên danh sách được cài đặt trên mảng. Cho danh sách các số nguyên, thêm vào 1 số nguyên và loại bỏ một số nguyên.

2.3. Lưu trữ kế tiếp cho danh sách tuyến tính

- | Dùng mảng một chiều làm cấu trúc lưu trữ danh sách tuyến tính. Tức là dùng vector lưu trữ (V_i) với $1 \leq i \leq m$ để lưu trữ một danh sách tuyến tính (a_1, a_2, \dots, a_n) . Phần tử a_i được chứa ở V_i .

a_1	$a_2 \dots$	$a_i \dots$	a_n
V_1	$V_2 \dots$	$V_i \dots$	$V_n \dots V_m$

2.3. Lưu trữ kế tiếp cho danh sách tuyến tính

- I Do số phần tử của danh sách tuyến tính luôn biến động, tức là kích thước n luôn thay đổi, do vậy $m = \max(n)$.
- I Mặt khác, n không thể xác định chính xác mà chỉ dự đoán. Bởi vậy, nếu $\max(n)$ lớn sẽ lãng phí bộ nhớ cũng như lãng phí thời gian để thực hiện các thao tác để dồn các phần tử xuống khi ta thêm một phần tử vào danh sách tuyến tính.

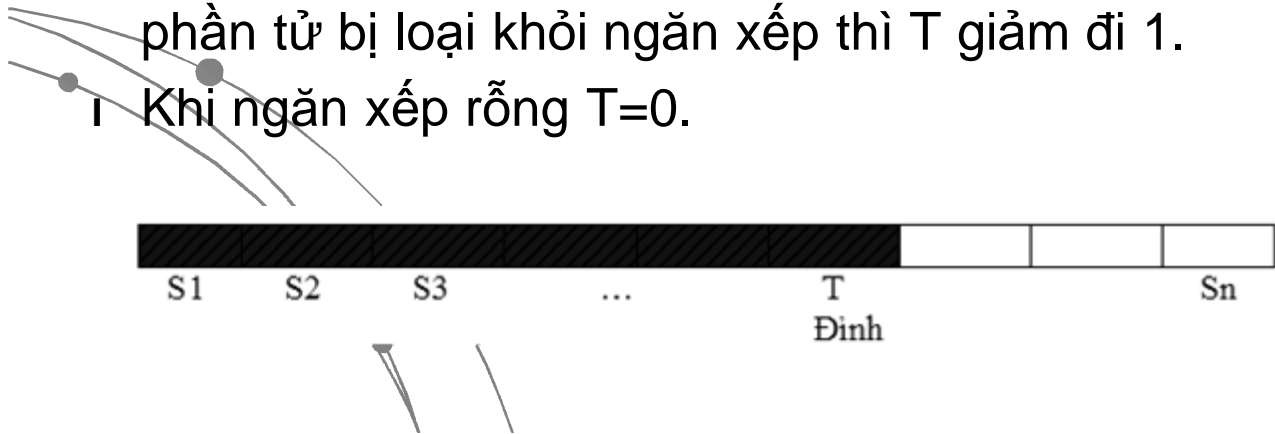
3. Cấu trúc ngăn xếp (Stack)

3.1. Định nghĩa

- I Ngăn xếp là một kiểu danh sách tuyến tính đặc biệt mà phép bổ sung và phép loại bỏ luôn luôn thực hiện ở một đầu gọi là đỉnh (Top).
- I Phép bổ sung và loại bỏ phần tử của ngăn xếp được thực hiện theo nguyên tắc "Vào sau ra trước" (Last in - First out, viết tắt là LIFO).
- I Ngăn xếp có thể rỗng.

3.2. Lưu trữ kế tiếp

- I Dùng vector lưu trữ S có n ô nhớ kế tiếp nhau với chỉ số từ 1 đến n để lưu trữ các phần tử dữ liệu.
- I Dùng biến T để lưu trữ chỉ số phần tử đỉnh của ngăn xếp, T sẽ thay đổi khi ngăn xếp hoạt động. Khi bổ sung 1 phần tử T sẽ tăng lên 1. Khi 1 phần tử bị loại khỏi ngăn xếp thì T giảm đi 1.
- I Khi ngăn xếp rỗng $T=0$.



3.3. Các phép toán trên ngăn xếp

- I Bổ sung một phần tử vào stack
 - Vào: phần tử x , ngăn xếp (S, T)
 - Ra: không có

{Thủ tục này bổ sung phần tử x vào ngăn xếp được lưu trữ bởi véc tơ S có kích thước là n , có chỉ số đỉnh là T .}

Thủ tục bổ sung một phần tử vào stack

Procedure push(Var S, T; x)

1) {Kiểm tra ngăn xếp đã đầy chưa?}

If $T = n$ then Begin Write('Stack đã đầy')

Return

End;

2) {Thay đổi chỉ số}

$T := T + 1$

3) {Bổ sung phần tử mới x}

$S[T] := x$

Return

Ngô Công Thắng

Bài giảng Cấu trúc dữ liệu và giải thuật - Chương 03

3.15

3.3. Các phép toán trên ngăn xếp

I Loại bỏ một phần tử ra khỏi stack

- Vào: Ngăn xếp (S, T)

- Ra: giá trị phần tử loại bỏ (đỉnh)

{Hàm này thực hiện việc loại bỏ phần tử ở đỉnh ngăn xếp (S,T) và trả về phần tử này.}

Hàm loại bỏ phần tử khỏi ngăn xếp

Function pop(Var S, T)

1) {Kiểm tra xem stack có rỗng?}

If T = 0 then Begin

Write('Stack rỗng')

Return;

End

2) Tg := S[T]; {Giữ lại phần tử đỉnh}

3) {Thay đổi chỉ số đỉnh}

T := T-1

4) {Trả về phần tử bị loại}

Pop := Tg;

Return

Ngô Công Thắng

Bài giảng Cấu trúc dữ liệu và giải thuật - Chương 02

2.17

Hàm kiểm tra ngăn xếp rỗng

Function isEmpty(S,T)

If T = 0 then Empty := TRUE

Else Empty := FALSE;

Return

Function isFull(S,T)

If T = n then Full := TRUE

Else Full := FALSE;

Return

Ngô Công Thắng

Bài giảng Cấu trúc dữ liệu và giải thuật - Chương 02

2.18

Hàm trả về phần tử đỉnh ngăn xếp

Function top(S,T)

1) {Kiểm tra xem stack có rỗng?}

If T = 0 then Begin

Write('Stack rỗng')

Return;

End

2) {Trả về phần tử đỉnh}

Top := S[T];

Return

Ví dụ về ứng dụng của ngăn xếp

- I Viết giả mã có sử dụng ngăn xếp để đổi số nguyên hệ 10 sang hệ 2.
- I Giải thuật: Lấy số hệ 10 chia nguyên liên tiếp cho 2, kết quả là phần dư của phép chia lấy theo thứ tự ngược lại. Áp dụng cơ chế vào sau ra trước, mỗi lần chia ta lấy số dư của phép chia đẩy vào stack (thủ tục Push). Khi đã kết thúc phép chia, kết quả lấy các số dư từ stack ra (hàm loại bỏ phần tử khỏi stack, Pop).

Ví dụ về ứng dụng của ngăn xếp

- Vào: n

- Ra: số nhị phân

Procedure chuyen_doi (n);

While n <> 0 do Begin

R:=n mod 2

Call Push(S,T,R);

n= n div 2

End;

While S <> NULL do Begin

R:=POP(S,T); {lay R tu dinh T cua Stack S }

Write(R)

End;

Return;

Ngô Công Thắng

Bài giảng Cấu trúc dữ liệu và giải thuật - Chương 02

3.21

Bài tập

- 1 Ứng dụng ngăn xếp chuyển biểu thức trung tố hành hậu tố. Biết rằng biểu thức trung tố có dấu ngoặc đầy đủ.

Ngô Công Thắng

Bài giảng Cấu trúc dữ liệu và giải thuật - Chương 02

3.22

4. Cấu trúc hàng đợi (Queue)

4.1. Định nghĩa

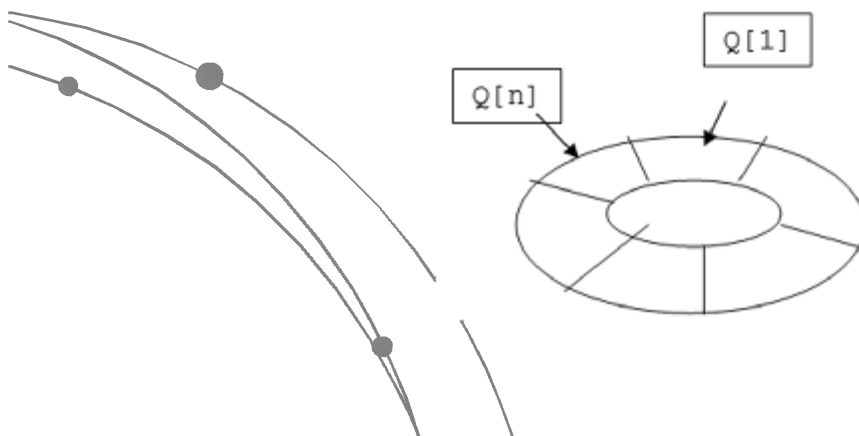
- | Hàng đợi (queue) là kiểu danh sách tuyến tính mà phép bổ sung được thực hiện ở một đầu, gọi là lối sau (rear) và phép loại bỏ thực hiện ở một đầu khác, gọi là lối trước (front).
- | Phép bổ sung và loại bỏ phần tử của hàng đợi được thực hiện theo nguyên tắc vào trước ra trước (First in - First out, viết tắt là FIFO).
- | Hàng đợi có thể rỗng.

4.2. Lưu trữ kế tiếp

- | Dùng vector lưu trữ Q có n ô nhớ với chỉ số từ 1 đến n để lưu trữ các phần tử dữ liệu của hàng đợi.
- | Dùng biến R chứa chỉ số của phần tử lối sau và F cho lối trước.
- | Khi bổ sung 1 phần tử vào hàng đợi thì R tăng lên 1, còn khi loại bỏ một phần tử khỏi hàng đợi thì F tăng lên 1.
- | Khi hàng đợi rỗng thì $R=F=0$.

4.2. Lưu trữ hàng đợi bằng mảng

- I Để sử dụng lại các ô nhớ chứa phần tử dữ liệu đã loại bỏ, người ta sử dụng các ô nhớ của vector lưu trữ Q theo kiểu quay vòng, tức là tiếp theo ô nhớ n là ô nhớ 1 .



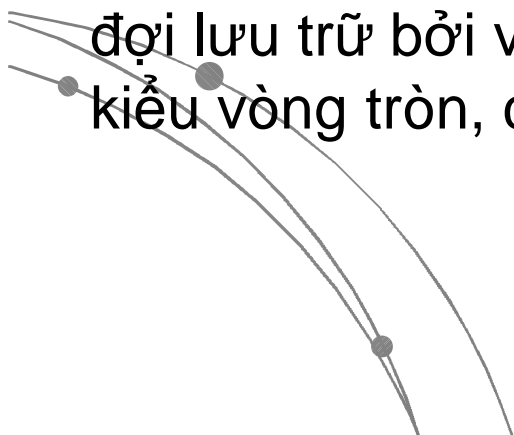
4.3. Các phép toán trên Queue

a) Bổ sung một phần tử vào queue

- I Vào: $x, (Q, F, R)$

- I Ra: Không có

{Thủ tục này bổ sung phần tử x vào hàng đợi lưu trữ bởi vector Q có n ô nhớ theo kiểu vòng tròn, có biến chỉ số F, R }



Procedure CQInsert (Var Q,F,R; x)

1) {Kiểm tra đầy}

If $(F=1)$ and $(R=n)$ or $(R+1=F)$ then Begin

Write('Hàng đợi đã đầy');

Return;

End;

2) {Thay đổi chỉ số R}

If $F=R=0$ Then $F:=R:=1$

Else If $R=n$ Then $R:=1$

Else $R:= R+1$;

3. {Bổ sung x vào}

$Q[R]:=x$;

Return {kết thúc}

4.3. Các phép toán trên Queue

b) Loại bỏ phần tử ra khỏi queue

- Vào: Hàng đợi (Q,F,R)

- Ra: Trả về phần tử loại bỏ

{Hàm này loại bỏ phần tử ở lối trước của hàng đợi (Q,F,R) và trả về phần tử loại bỏ}

Thủ tục loại bỏ phần tử khỏi hàng đợi
Function CQDelete(Var Q,F,R)

1) {Kiểm tra rỗng}

If $F=R=0$ then Begin

Write('Hàng đợi đã rỗng');

Return;

End;

2) {Lưu lại phần tử loại bỏ}

Tg:=Q[F]

Thủ tục loại bỏ phần tử khỏi hàng đợi
Function CQDelete(Var Q,F,R)

3) {Thay đổi chỉ số F}

If $F=R$ then $F:=R:=0$

Else If $F=n$ then $F:=1$

Else $F:=F+1$;

4) CQDelete := Tg;

Return

Kiểm tra hàng đợi rỗng

Function CQIsEmpty(Q,F,R)

If F=R=0 then

CQIsEmpty:= TRUE

Else

CQIsEmpty:=FALSE;

Return

