CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

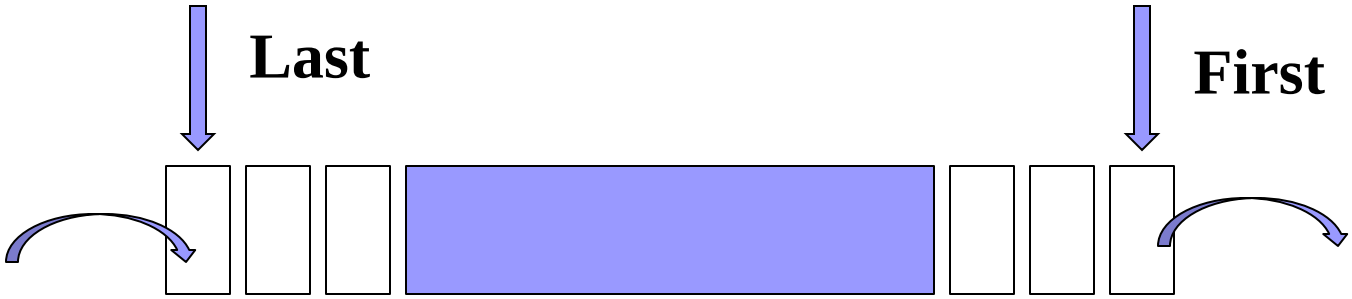
CHƯƠNG 4:HÀNG ĐỢI VÀ NGĂN XẾP

Nội dung

- 4.1. Cấu trúc DL kiểu hàng đợi (Queue)
- 4.2. Ứng dụng của hàng đợi
- 4.3. Cấu trúc DL kiểu ngăn xếp (Stack)
- 4.4. Ứng dụng của ngăn xếp

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- 1. Định nghĩa:
  Hàng đợi là một CTDL dạng danh sách FIFO (First In First Out).
- Để quản lý DS cần 2 con trỏ chạy cho đầu DS và cuối DS:
  First và Last.
- Việc thêm vào DS thực hiện ở Last, việc lấy ra thực hiện ở First.



DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- 2. Cài đặt Queue bằng mảng 1 chiều:

```
struct Queue
    { int First;
    int Last;
    Data Element[MaxQueue];
    };
    Queue Q;
```

- 3. Các thao tác trên hàng đợi:

a) Khởi tạo 1 hàng đợi rỗng:

```
void Create(Queue &Q)
    {Q.First=0;Q.Last=-1;}
```

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- 3. Các thao tác trên hàng đợi:

b) Kiểm tra hàng đợi rỗng:

```
int Empty(Queue Q)
      {  return Q.First>Q.Last?1:0; }
```

c) Đưa 1 phần tử vào hàng:

```
void Add(Data x, Queue &Q)
  {
    if(Q.Last==MaxQueue-1) return;
    else
  {
    Q.Last++;
    Q.Element[Q.Last]=x;
  }
  }
```

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- 3. Các thao tác trên hàng đợi:

d) Lấy 1 phần tử ra khỏi hàng

```
void Remove(Data &x,Queue &Q)
{
    if(Empty(Q)) return;
    x=Q.Element[Q.First];
    Q.First++;
}
```

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- 4. Ứng dụng của hàng đợi

a) Tìm số lần biến đổi ít nhất của một dãy biến đổi.
Bài toán 1: Cho một số nguyên dương x và 1 số nguyên dương y. Có ba phép biến đổi: Giảm 1 đơn vị, nhân đôi và chia đôi lấy phần nguyên. Hãy tìm số phép biến đổi ít nhất để đổi x thành y.

- Vd: x = 5; y = 11
chúng ta có thể có nhiều cách để biến đổi từ 5 về 11 theo 3 phép biến đổi trên. Chẳng hạn: 5 -> 4 ->3->6->12->11.
Tuy nhiên điều cần làm là "Số phép biến đổi ít nhất"

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- Giải thuật:

1. Đưa x vào hàng đợi. Đánh dấu x đã được tạo ra.
2. Lấy u ra khỏi hàng: Tạo ra u_1, u_2, u_3
3. Nếu một trong các số đó là y: Dừng lại

   ngược lại: Nếu số nào chưa có trong hàng(chưa được đánh dấu):

   - Đưa số đó vào hàng.
   - Đánh dấu u là "tiền bối" của số đó.

   Lặp lại 2.
4. Lần ngược từ y về x ta dựa vào dãy "tiền bối", ta sẽ thu được dãy biến đổi it nhất.

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- Bài toán 2: Có N thẻ được đánh số từ 1 đến N và một máy đổi thẻ. Bỏ 1 thẻ vào máy có thể cho ra 1 số thẻ khác. Cho 2 thẻ có số hiệu u và v. Hãy tìm số lần đổi thẻ ít nhất để từ u thu được v.

  Vd: n= 5

  1 bỏ vào cho ra 2,4

  2 bỏ vào cho ra 2

  3 bỏ vào cho ra 1,4

  4 bỏ vào cho ra 2,3,5

  5 bỏ vào cho ra 3,4.

Cho u = 5, v= 1.

- Câu hỏi: Khi nào ta có khẳng định: Từ u ko thể đổi ra v?
- Khi thuật toán dừng do hàng đợi rỗng mà v chưa tạo ra thì ta khẳng định từ u ko thể đổi ra v.

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

b) Tìm đường đi với số cung ít nhất (Đường đi ngắn nhất)

- Bài toán 1: Tìm đường trong mê cung:
  Một mê cung có N phòng, các phòng có thể được nối với nhau bởi 1 hành lang. (Hành lang có thể chỉ có 1 chiều). Một người muốn đi từ phòng u đến phòng v.
  - Có thể đi đến ko?
  - Nếu có thể, hãy chỉ ra đường qua ít hành lang nhất.
- Bài toán 2: Mê cung hai chiều

  Một lưới ô vuông có mxn ô, mỗi ô ghi số 1 hoặc số 0. Ô số 1 là ô có thể đi vào, ô số 0 là ko thể đi vào. Một người đứng ở tọa độ (x,y) muốn thoát khỏi mê cung. Hãy chỉ ra đường đi ngắn nhất để thoát.

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- vd: n=5,m=6

  10110

  00111

  11100

  00101

  11101

  00110

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

- Một số bài tập tham khảo

Bài 1: Cho một bàn cờ vua có nxn ô. Tại ô (xm,ym) đặt một quân mã và tại ô (xh,yh) đặt một quân Hậu khác màu. Hãy tìm cách di chuyển quân Mã đến vị trí (xd,yd) sao cho: Ko bị quân Hậu bắt và đến (xd,yd) nhanh nhất.

Bài 2: Bài toán rót nước

Cho 3 bình có dung tích = nhau và = 100 lít. Ở 2 bình đầu tiên đều có vạch chia theo số lít. Các vạch này là tùy ý nhưng đều ở số nguyên lít. Ban đầu bình 1 đựng đầy nước. Hãy tìm cách đổ nước từ bình nọ sang bình kia sao cho cuối cùng ở bình 3 thu được 1 lít nước, biết rằng: Mỗi lần rót hoặc là rót hết hoặc là rót đến 1 vạch nào đó và số lần rót là ít nhất.

DỮ LIỆU KIỂU HÀNG ĐỢI (QUEUE)

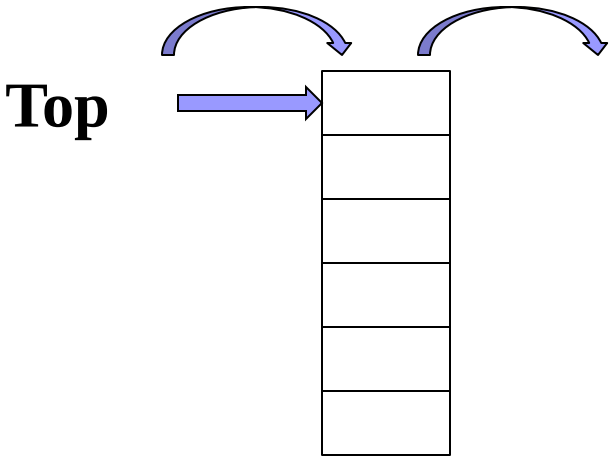
Một số kiểu hàng đợi khác

- Hàng đợi mô phỏng quầy bán vé: Cố định First = 0; Ta chỉ cần Last chính là số phần tử trong hàng-1;
  - Hàng rỗng ⇔ Last = -1;
  - Đưa vào hàng: Q.Element[Q.Last]=x; Q.Last++;
  - Lấy ra khỏi hàng:
    - x=Q.Element[0];
    - Đẩy các phần tử phía sau lên: for(i=0;i<Q.Last-1;i++) Q.Element[i]=Q.Element[i+1];Last--;
- Hàng đợi vòng tròn: Để tiết kiệm bộ nhớ. Giữa First và Last có 1 vị trí trống để phân biệt hàng đầy và hàng rỗng. Sau 1 vòng vượt qua Maxqueue thì Last và First lại quay về 0. Do đó để có Last và First sau mỗi lần tăng giảm: Last = (Last+1) % Maxqueue; First = (First+1) % Maxqueue;
- (Đọc thêm phần này)

DỮ LIỆU KIỂU NGĂP XẾP (STACK)

1. Định nghĩa:

- Ngăn xếp là một CTDL dạng danh sách LIFO (Last In First Out).
- Để quản lý DS cần 1 con trỏ ở đầu DS gọi là Top.
- Việc thêm vào và lấy ra khỏi DS đều thực hiện ở Top



DỮ LIỆU KIỂU NGĂN XẾP(STACK)

2. Cài đặt Stack bằng mảng 1 chiều:

Kiểu dữ liệu

```
struct Stack
        { int Top;
        Data Element[MaxStack];
        };
```

Biến ngăn xếp: Stack S;

3. Các thao tác trên ngăn xếp:

a) Khởi tạo 1 ngăn xếp rỗng:

```
void Init(Stack  &S)
        {S.Top=0;} hoặc {S.Top=-1}
```

DỮ LIỆU KIỂU NGĂN XẾP(STACK)

3. Các thao tác trên ngăn xếp:

b) Kiểm tra ngăn xếp rỗng:

```
int Empty(Stack S)
      {  return S.Top==0 ?1:0; }
```

c) Đẩy 1 phần tử vào ngăn xếp:

```
void  Push(Data x, Stack &S)
  {
    if(S.Top==MaxStack-1) return;
    else
  {
    S.Top++;
    S.Element[S.Top]=x;
  }
  }
```

DỮ LIỆU KIỂU NGĂN XẾP(STACK)

- 3. Các thao tác trên ngăn xếp:

d) Lấy 1 phần tử ra khỏi ngăn xếp

```
void Pop(Data &x,Stack  &S)
{
    if(Empty(S)) return;
    x=S.Element[S.Top];
    S.Top--;
}
```

DỮ LIỆU KIỂU NGĂN XẾP(STACK)

- 4. Các ứng dụng của stack:

  a) Bài toán đổi cơ số

  Bài toán: Đổi một số nguyên ở cơ số 10 ra cơ số R.

  Giải thuật:

  Do khi đổi ra cơ số R ta phải chia dần số cần đổi cho R đến khi thương = 0. Số dư lấy theo chiều ngược lại sẽ được số ở cơ số R.

Ở đây ta thấy số dư nào xuất hiện trước trong phép chia sẽ xuất hiện sau trong biểu diễn ở cơ số R. Do đó ta sử dụng Stack để lưu số dư.

```
Init(S);
while (N>0)
{ Push(N%R,S);
N=N/R;
}
```

```
while (!Empty(S))
{ Pop(i,S);
printf("%d",i);
}
```

DỮ LIỆU KIỂU NGĂN XẾP(STACK)

- 4. Các ứng dụng của stack:

  a) Bài toán đổi cơ số

  Ta cần chú ý đến R, nếu R>9 thì biểu diễn N ở cơ số R có thể có chữ cái. Vd: Đổi ra cơ số 16 (Hexa) nếu khi chia N cho 16 ta có thể nhận được các số dư từ 10 đến 15, khi đó các số dư này phải biểu diễn bằng chữ cái A đến F

DỮ LIỆU KIỂU NGĂN XẾP(STACK)

4. Các ứng dụng của stack:

b) Bài toán Ký pháp nghịch đảo Balan (Reserve Polish Notation – RPN)

Bài toán: Chuyển đổi một biểu thức thông thường thành dạng không có các dấu đóng mở ngoặc.

Để đơn giản cho việc minh họa, ta giả định rằng chuỗi biểu thức mà ta nhận được từ bàn phím chỉ bao gồm: các dấu mở ngoặc/đóng ngoặc; 4 toán tử cộng, trừ, nhân và chia (+, -, *, /); các toán hạng đều chỉ là các con số nguyên từ 0 đến 9; không có bất kỳ khoảng trắng nào giữa các ký tự.

Reserve Polish Notation

- **Thế nào là ký pháp nghịch đảo Ba Lan?**
  Cách viết biểu thức thông thường rất khó tính toán trong máy tính.
  Xét một biểu thức đơn giản: 3+4, chúng ta có thể viết +34 hay 34+.
  Dựa vào vị trí của toán tử (dấu +) ta gọi các biểu thức đó là:
  Prefix (tiền tố), Infix (Trung tố) và Postfix (Hậu tố).
- RPN là bài toán chuyển Infix sang Postfix.
- Ký pháp nghịch đảo Ba Lan được phát minh vào khoảng giữa thập kỷ 1950 bởi Charles Hamblin – một triết học gia và khoa học gia máy tính người Úc – dựa theo công trình về ký pháp Ba Lan của nhà Toán học người Ba Lan Jan Łukasiewicz. Hamblin trình bày nghiên cứu của mình tại một hội nghị khoa học vào tháng 6 năm 1957 và chính thức công bố vào năm 1962.

Reserve Polish Notation

- **Chuyển đổi từ trung tố sang hậu tố**
  Thuật toán chuyển đổi này được phát minh bởi vị giáo sư người Đức nổi tiếng Edsger Dijkstra. Thuật toán này cũng dựa theo cơ chế ngăn xếp.
- Các khái niệm: Toán hạng là các chữ số hoặc chữ cái (biến)
  Toán tử là các phép toán +,-.*,/ và dấu (
- Độ ưu tiên các toán tử: ‘(’<{‘+’,’-‘} < {‘*’,’/’}
- Đầu vào: Biểu thức Infix, đầu ra: biểu thức Postfix
  + Khởi tạo một Stack S rỗng và một chuỗi Postfix rỗng
- .

Reserve Polish Notation

- **Chuyển đổi từ trung tố sang hậu tố**

  Giải thuật: Duyệt biểu thức Infix từ trái sang phải:

  - Nếu gặp toán hạng: Đưa vào Postfix
  - Nếu gặp dấu mở ngoặc, đưa nó vào stack
  - Nếu gặp một toán tử (gọi là o_1), thực hiện hai bước sau:
    - Nhìn toán tử o_2 ở đỉnh ngăn xếp: Nếu độ ưu tiên của o_1 **nhỏ hơn hay bằng** độ ưu tiên của o_2 thì lấy o_2 ra khỏi ngăn xếp và ghi vào Postfix.
    - Push o_1 vào ngăn xếp
  - Nếu gặp dấu đóng ngoặc thì cứ lấy các toán tử trong ngăn xếp ra và ghi vào Postfix cho đến khi lấy được dấu mở ngoặc ra khỏi ngăn xếp. (Không ghi dấu mở ngoặc vào Postfix).
  - Khi đã duyệt hết biểu thức Infix, lần lượt lấy tất cả toán hạng (nếu có) từ ngăn xếp ra và ghi vào chuỗi Postfix.

Chuyển đổi: **3+4*2/(1-5) ra Postfix**

Ký tự	Thao tác	Stack	Chuỗi hậu tố
3	Ghi 3 vào Postfix		3
+	Push +	+	
4	Ghi 4 vào Postfix		3 4
*	Push *	+ *	
2	Ghi 2 vào Postfix		3 4 2
/	Lấy * ra khỏi stack, ghi vào Postfix, push /	+ /	3 4 2 *
(	Push (	+ / (	3 4 2 *
1	Ghi 1 vào Postfix	+ / (	3 4 2 * 1
-	Push -	+ / (–	3 4 2 * 1
5	Ghi 5 vào Postfix	+ / (–	3 4 2 * 1 5
)	Pop cho đến khi lấy được (, ghi các toán tử pop được ra Postfix	+ /	3 4 2 * 1 5 -
	Pop tất cả các toán tử ra khỏi ngăn xếp và ghi vào Postfix		3 4 2 * 1 5 – / +

Tính toán một biểu thức hậu tố

- Quá trình tính toán giá trị của biểu thức hậu tố khá tự nhiên đối với máy tính.
- Ý tưởng là đọc biểu thức từ trái sang phải:
  - nếu gặp một toán hạng (con số hoặc biến): push toán hạng này vào ngăn xếp;
  - nếu gặp toán tử: lấy hai toán hạng ra khỏi ngăn xếp (stack), tính kết quả, đẩy kết quả trở lại ngăn xếp. Khi quá trình kết thúc thì con số cuối cùng còn lại trong ngăn xếp chính là giá trị của biểu thức đó.
  - Lưu ý phép trừ và phép /. Ví dụ: 3/4 đổi thành 34/, nếu không cẩn thận ta sẽ nhầm thành 4/3!!!

Tính toán một biểu thức hậu tố

Ví dụ: biểu thức trung tố : **5 + ((1 + 2) * 4) + 3**

được biểu diễn lại dưới dạng hậu tố là: **5 1 2 + 4 * + 3 +**

Ký tự	Thao tác	Trạng thái stack
5	Push 5	5
1	Push 1	5, 1
2	Push 2	5, 1, 2
+	Tính 1 + 2 Push 3	5, 3
4	Push 4	5, 3, 4
*	Tính 3 * 4 Push 12	5, 12
+	Tính 12 + 5 Push 17	17
3	Push 3	17, 3
+	Tính 17 + 3 Push 20	20

ĐƠN GIẢN HÓA STACK VÀ QUEUE

- Chúng ta có thể cài đặt Queue và Stack đơn giản bằng một mảng một chiều Q và S.
- Với queue Q chúng ta sử dụng 2 chỉ số First và Last như trước:
  - Thêm vào: Last++;Q[Last]=x;
  - Lấy ra: x = Q[First];First++;
  - Empty = First>Last;
- Với stack S: số phần tử N coi như Top.
  - Init: N=0
  - Push: S[N]=x;N++;
  - Pop: x=S[N];N--;
  - Empty = N==0;

STACK VÀ QUEUE LIÊN KẾT

I. Stack cài đặt bằng DS liên kết đơn

- Do stack chỉ vào ra ở một đầu nên chỉ cần 1 con trỏ Top để quản lý DS.

1.1 Khai báo: Xem lại khai báo LIST ở DS liên kết đơn.

Chỉ cần khai báo thêm: LIST Top;

1.2 Các thao tác:

a) Khởi tạo Stack rỗng: Top=NULL;

b) Kiểm tra stack rỗng: return Top==NULL?1:0

c) Đẩy vào Stack phần tử x(Push): p= new LIST; p.Info=x;
Móc nối p vào Stack: p->pNext=Top; Top=p;

d) Lấy 1 phần tử ra khỏi stack (Pop): p=Top; x=p->Info;
Top=Top->pNext; Delete p;

STACK VÀ QUEUE LIÊN KẾT

- II. Cài đặt hàng đợi bằng DS liên kết đơn.

  2.1 Khai báo hai con trỏ như DS đơn: First và Last (Như 2 con trỏ pHead và pTail đã học)

  2.2 Các thao tác:

  a) Khởi tạo: First=Last =NULL;

  b) Kiểm tra Queue rỗng: return First==NULL?1:0;

  c) Thêm 1 phần tử vào hàng: p= new LIST; p->Info = x;

  p->pNext=NULL; Last->pNext = p; Last = p; (Chú ý xét DS rỗng!!!)

  d) Lấy 1 phần tử ra khỏi hàng:

  ```
  x=First->Info; p=First;
  First=First->pNext;
  delete p;
  ```