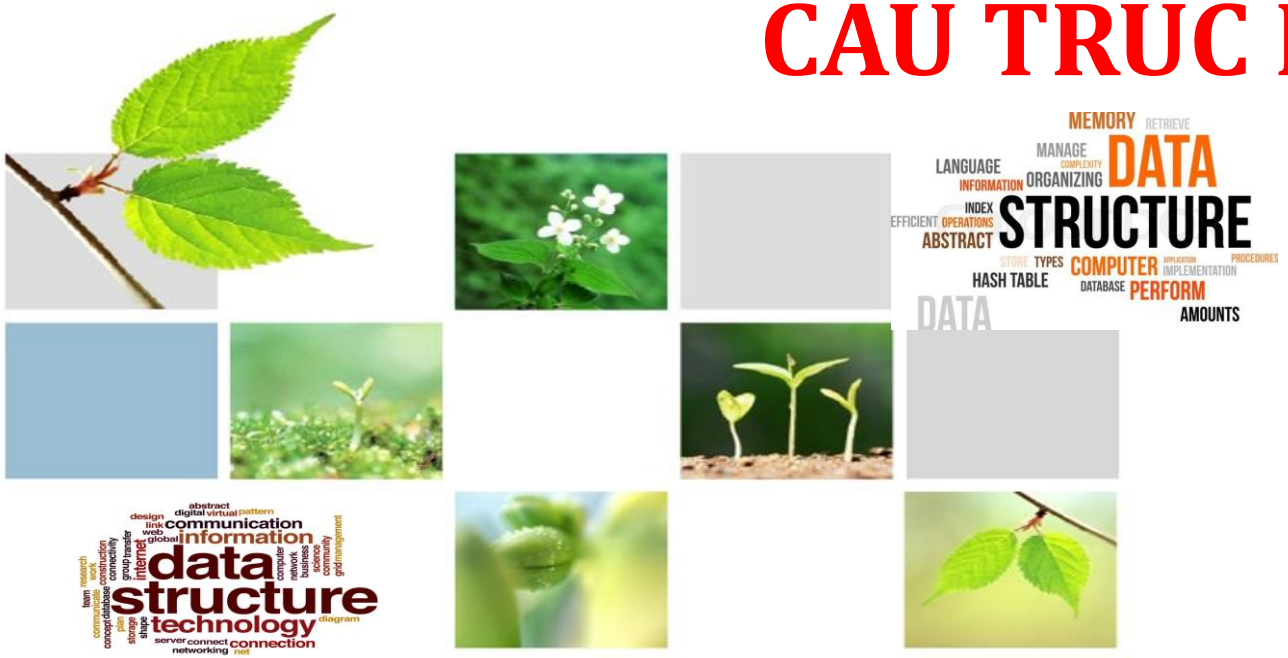
BUỔI 6: NGĂN XẾP (STACK) – HÀNG ĐỢI (QUEUE)



LÝ THUYẾT

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

IT003.N210

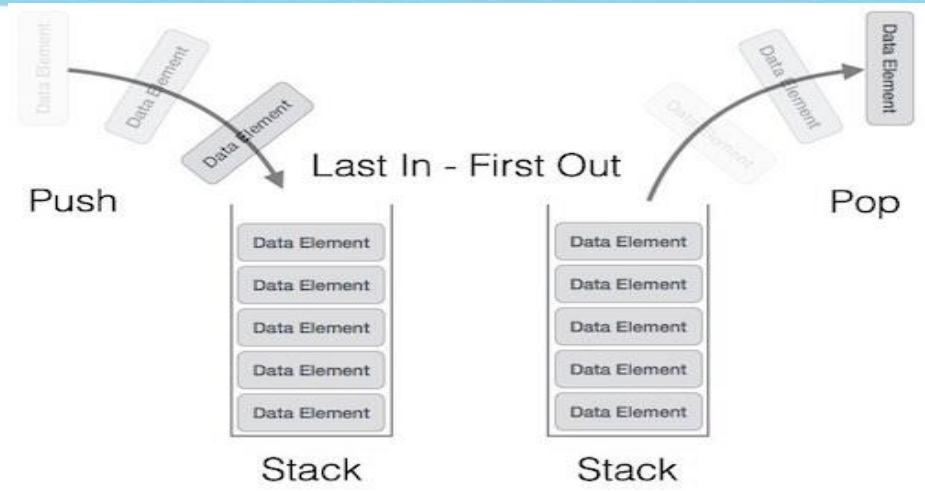
Giảng viên: Ths Đặng Văn Em

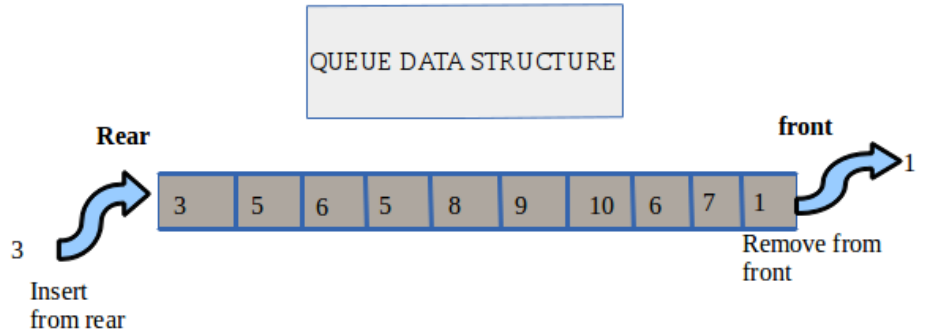
Email: vanem@uit.edu.vn

Điện thoại: 0966661006

BUỔI 6: NGĂN XẾP (STACK) – HÀNG ĐỢI (QUEUE)

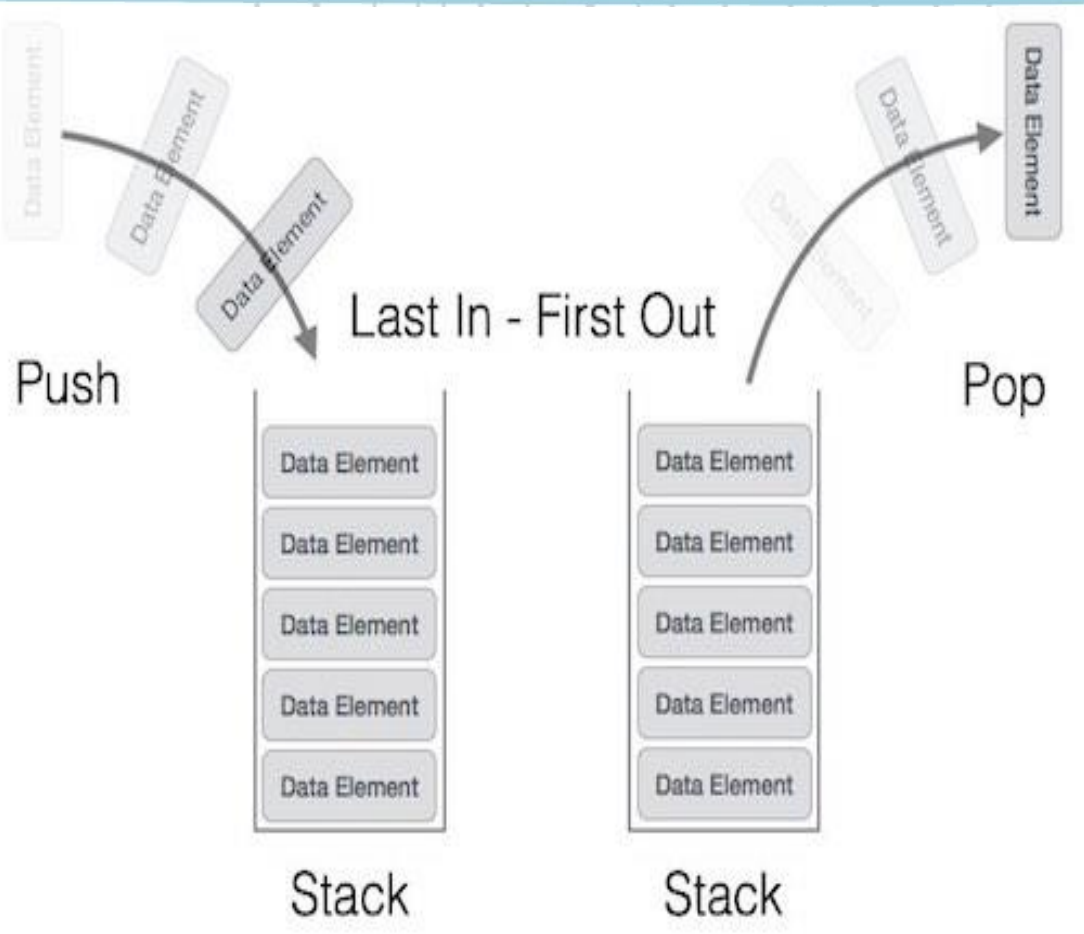
1. Ngăn xếp (Stack).
2. Hàng đơi (Queue).
3. Ví dụ minh họa về Stack và Queue.
   - ❖ Bài toán Tháp Hà nội.
   - ❖ Các Phép toán biểu thức số học
4. Bài tập chương





Stack (ngăn xếp):

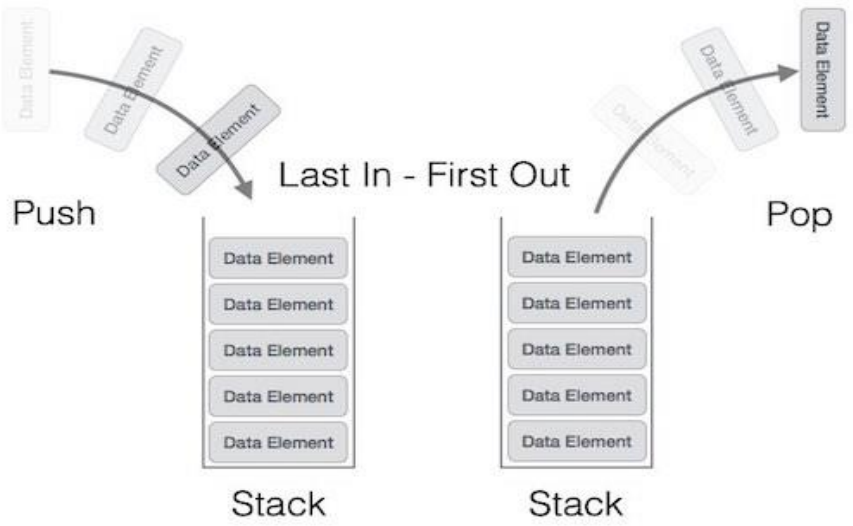
Stack (ngăn xếp): là 1 vật chứa các đối tượng làm việc theo cơ chế **LIFO** (Last In First Out), tức việc thêm 1 đối tượng vào Stack hoặc lấy 1 đối tượng ra khỏi Stack được thực hiện theo cơ chế "***vào sau ra trước***"



BUỔI 6: NGĂN XẾP (STACK) – HÀNG ĐỢI (QUEUE)

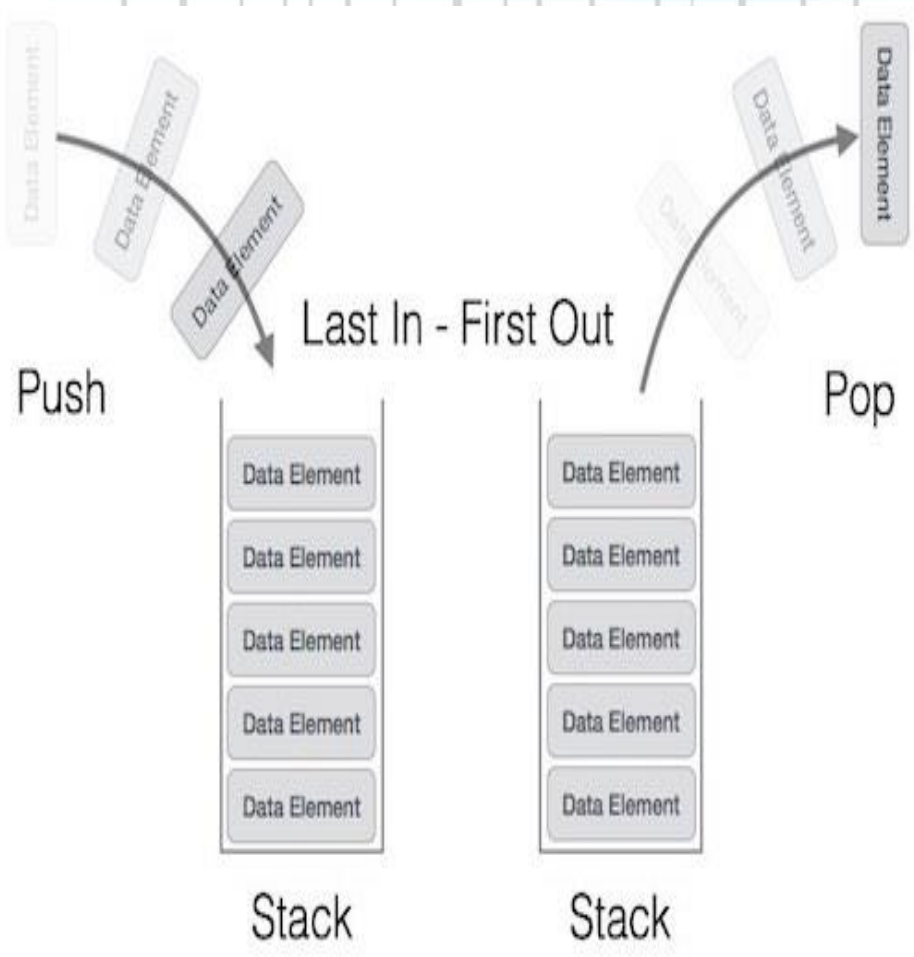
Ứng dụng trên Stack (ngăn xếp):

- ❖ Lưu vết các quá trình quay lui, vét cạn.
- ❖ Khử đệ quy đuôi.
- ❖ Đảo ngược chuỗi (theo cơ chế **LIFO**).
- ❖ Đổi một số n từ hệ cơ số thập phân sang hệ nhị phân.
- ❖ Nút Back trong trình duyệt web hay Nút Undo trong Word.
- ❖ Cài đặt/thực hiện lời gọi hàm trong trình biên dịch.



Các thao tác trên Stack (ngăn xếp):

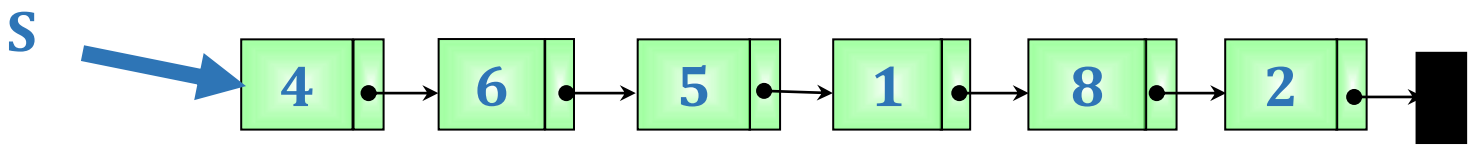
- **Push(p)**: Thêm đối tượng p vào Stack.
- **Pop(p)**: Lấy đối tượng p ra khỏi Stack.
- **isEmpty()**: Kiểm tra Stack có rỗng hay không?
- **IsFull ()**:Kiểm tra Stack đã đầy hay chưa.
- **Top()**: Trả về giá trị của phần tử nằm đầu của Stack mà không hủy nó khỏi Stack.



❖ **Cài đặt Stack (ngăn xếp) dùng mảng một chiều:**


```
Data   S [N];
int    t;
```

❖ **Cài đặt Stack (ngăn xếp) dùng danh sách liên kết:**



List S

?? Thêm và hủy cùng phía

Cài đặt Stack (ngăn xếp) dùng mảng một chiều:

❖ **Cấu trúc dữ liệu của Stack:**

```
typedef struct tagStack
{
    int a[max];
    int t;
}Stack;
```

❖ **Khởi tạo Stack:**

```
void CreateStack(Stack &s)
{
    s.t=-1;
}
```

Cài đặt Stack (ngăn xếp) dùng mảng một chiều:

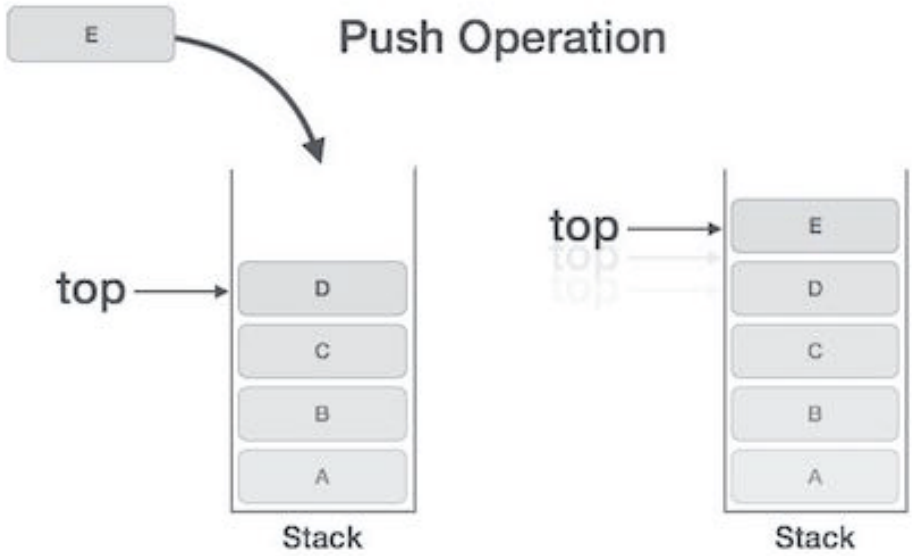
❖ Kiểm tra tính rỗng và đầy của Stack:

```
int IsEmpty(Stack s)
{
        if(s.t==-1)
                return 1;
        else
                return 0;
}
```

```
int IsFull(Stack s)
{
        if(s.t>=max)
                return 1;
        else
                return 0;
}
```

Cài đặt Stack (ngăn xếp) dùng mảng một chiều:

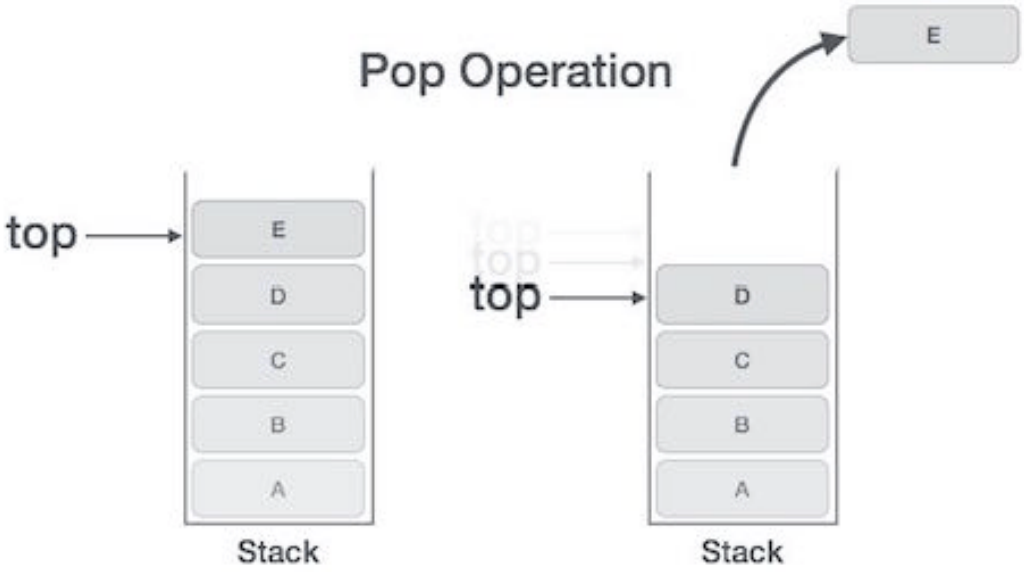
❖ **Thêm một phần tử vào Stack:**




```
int Push(Stack &s, int x)
{
        if(IsFull(s)==0)
        {
                s.t++;
                s.a[s.t]=x;
                return 1;
        }
        else
                return 0;
}
```

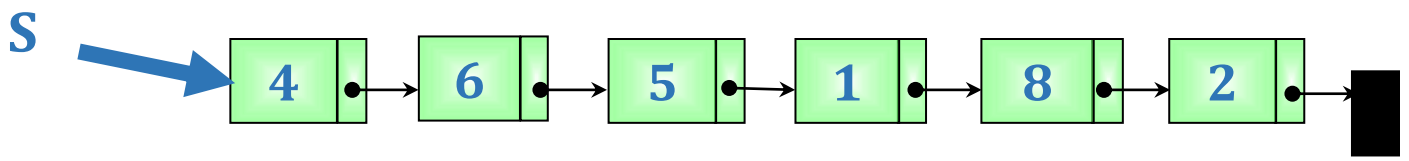
Cài đặt Stack (ngăn xếp) dùng mảng một chiều:

❖ **Lấy một phần tử ra khỏi Stack:**




```
int Pop(Stack &s, int &x)
{
        if(IsEmpty(s)==0)
        {
                x=s.a[s.t];
                s.t--;
                return 1;
        }
        else
                return 0;
}
```

❖ Cài đặt Stack (ngăn xếp) dùng danh sách liên kết:



❖ Cấu trúc dữ liệu của Stack:

```
typedef struct node
{
   Item  infor;
   node *next;
}node;
```

❖ Khởi tạo Stack:

```
void create(node *s)
{
   s = NULL;
}
```

Cài đặt Stack (ngăn xếp) dùng danh sách liên kết:

❖ **Khởi tạo Stack rỗng:**

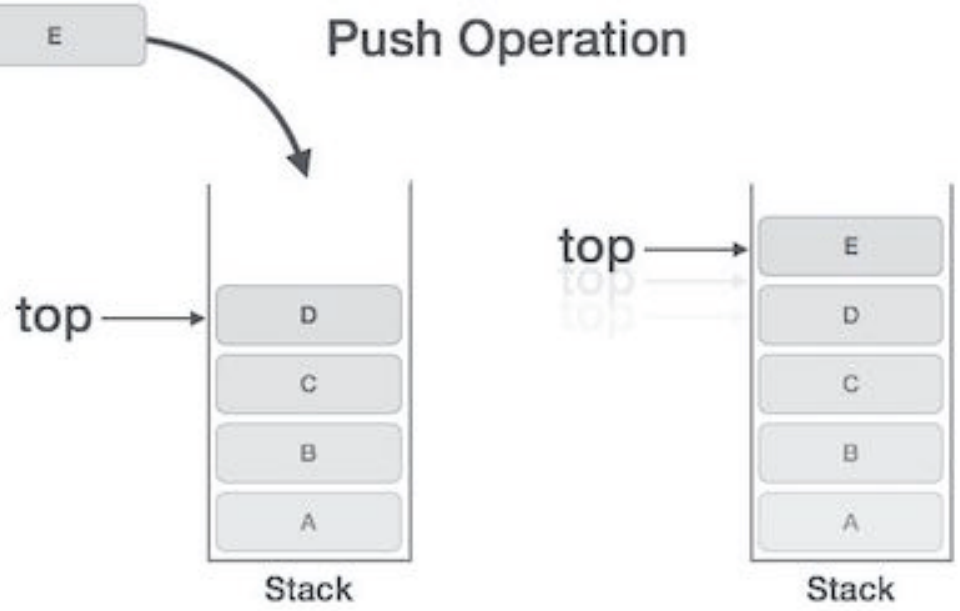
```
void CreatStack(node *top)
{
   top = NULL;
}
```

❖ **Kiểm tra tính rỗng của Stack**

```
int Empty(node *s)
{
   return (s == NULL);
}
```

Cài đặt Stack (ngăn xếp) dùng danh sách liên kết:

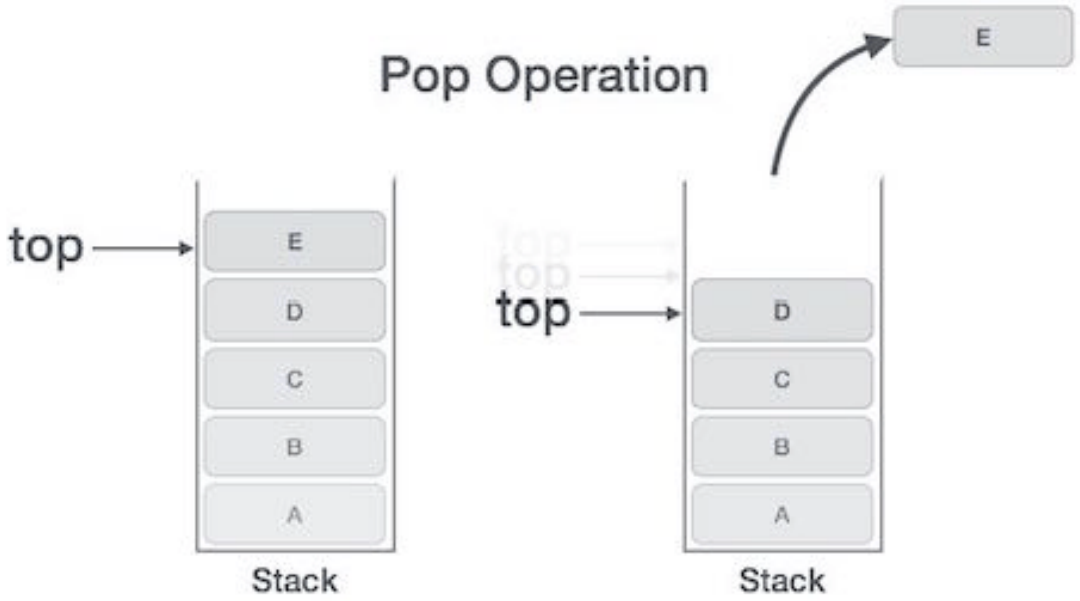
❖ **Thêm một phần tử vào Stack:**




```
void Push(struct node *s, item x)
{
   node *p;
   p= new node;
   p->infor= x;
   p->next= NULL;
   if (s==NULL)  s= p;
   else
   {
      p->next = s;
      s= p;
   }
}
```

Cài đặt Stack (ngăn xếp) dùng danh sách liên kết:

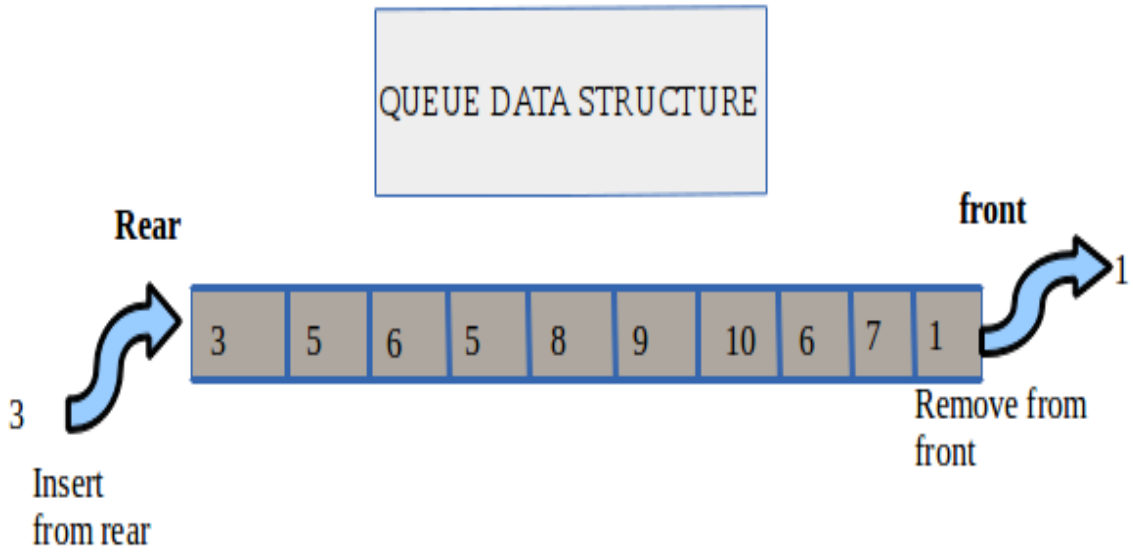
❖ **Lấy một phần tử ra khỏi Stack:**




```
void Push(node *s, item x)
{
   node *p;
   p= new node;
   p->infor= x;
   p->next= NULL;
   if (s==NULL)  s= p;
   else
   {
      p->next = s;
      s= p;
   }
}
```

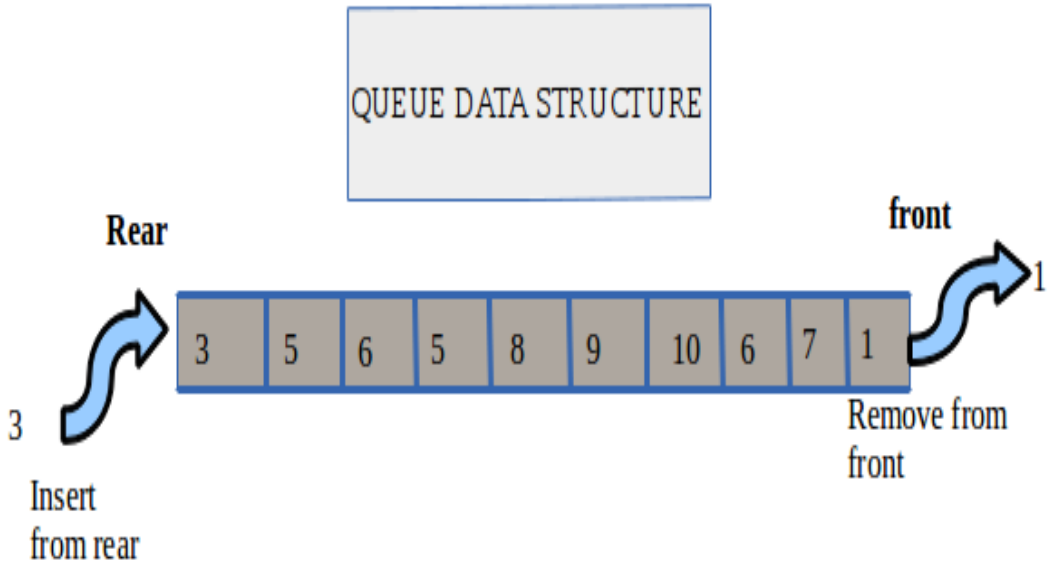
Queue (hàng đơi):

Queue (hàng đợi): Là 1 vật chứa các đối tượng làm việc theo cơ chế **FIFO (First In First Out)**: tức việc thêm 1 đối tượng vào hàng đợi hay lấy 1 đối tượng ra khỏi hàng đợi thực hiện theo cơ chế **"vào trước ra trước"**



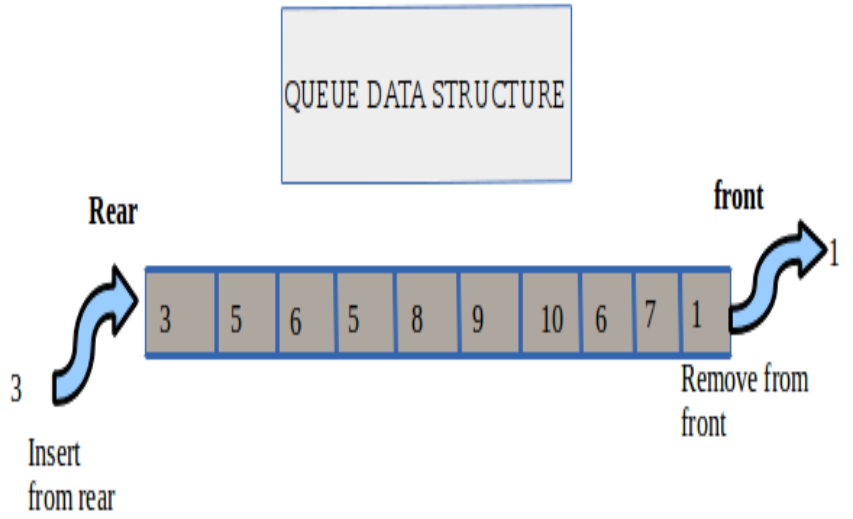
Ứng dụng Queue (hàng đơi):

- Bộ đệm dữ liệu (data buffer). Cơ chế spooling.
- Xử lý trong các trung tâm chăm sóc khách hàng.
- Xếp hành thanh toán trong siêu thị.
- Hàng đợi được sử dụng khi lập lịch CPU.



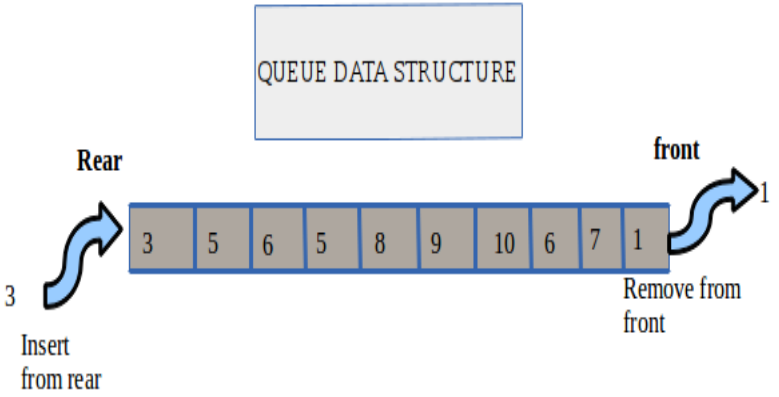
Các thao tac trên Queue (hàng đơi):

- **EnQueue(p)**: Thêm đối tượng p vào cuối hàng đợi.
- **DeQueue()**: Lấy đối tượng ở đầu hàng đợi.
- **isEmpty()**: Kiểm tra hàng đợi có rỗng hay không?
- **Front()**: Trả về giá trị của phần tử nằm đầu hàng đợi mà không hủy nó.

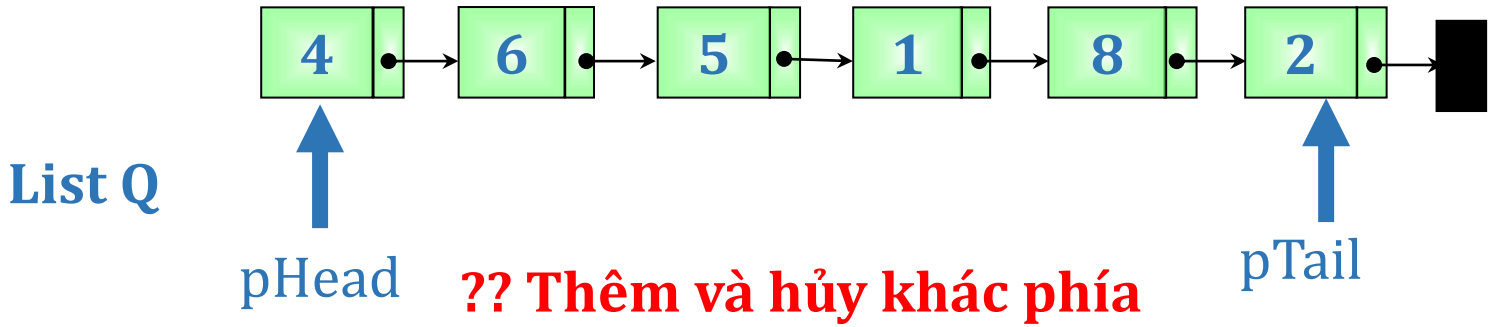


❖ **Cài đặt Queue (hàng đợi) dùng mảng một chiều:**


```
Data    Q[N];
int     f,r;
```



❖ **Cài đặt Queue (hàng đợi) dùng danh sách liên kết:**



?? Thêm và hủy khác phía

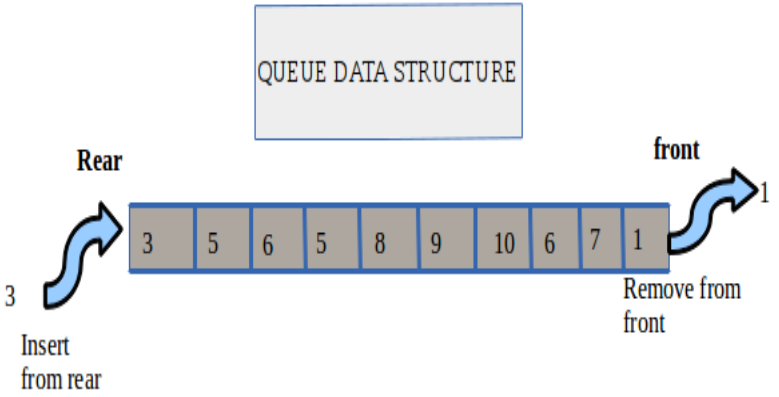
❖ Cài đặt Queue (hàng đợi) dùng mảng một chiều:

❖ Khai báo cấu trúc Queue

```
typedef struct tagQueue
{
    int a[100];
    int Front;
    int Rear;
}Queue;
```

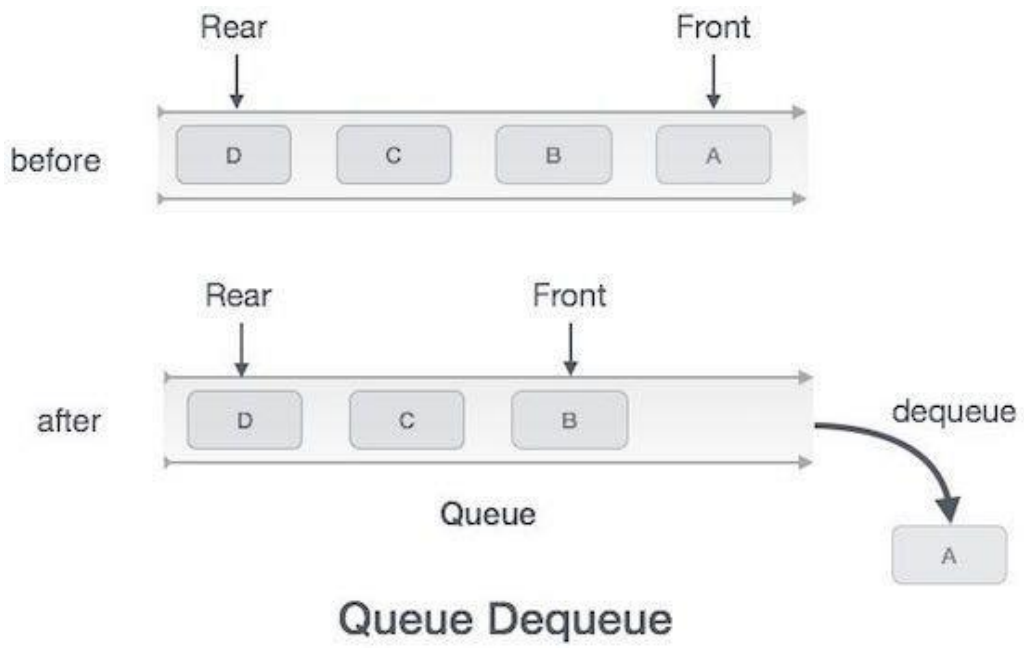
❖ Khởi tạo Queue rỗng

```
void CreateQueue(Queue &q)
{
  q.Front=-1;
  q.Rear=-1;
}
```



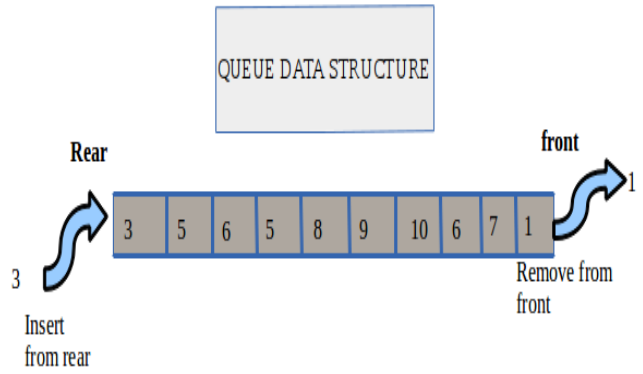
Cài đặt Queue (hàng đợi) dùng mảng một chiều:

❖ Lấy phần tử từ Queue:



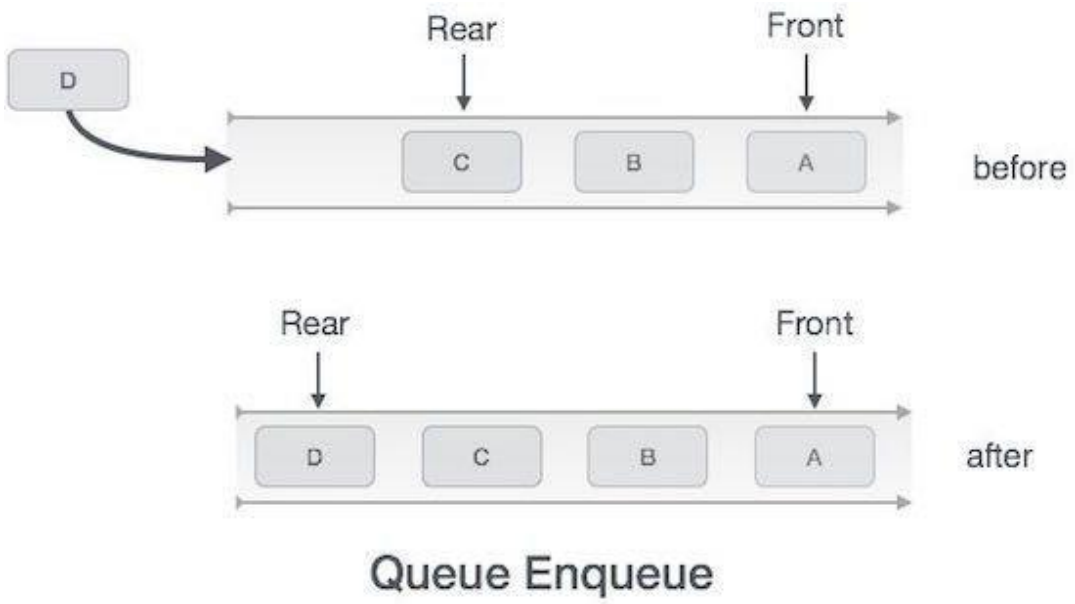
Queue Dequeue

```
int DeQueue(Queue &q,int &x)
{
   if(q.Front!=-1)
   {
      x=q.a[q.Front];
      q.Front++;
      if(q.Front>q.Rear)
      {
         q.Front=-1;
         q.Rear=-1;
      }
      return 1;
   }
   else
   {
      printf("Queue rong");
      return 0;
   }
}
```



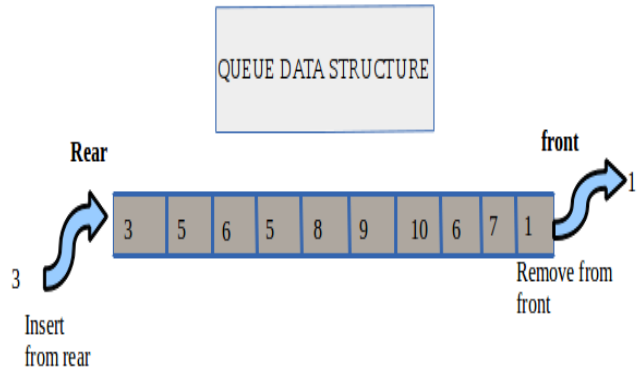
Cài đặt Queue (hàng đợi) dùng mảng một chiều:

❖ Thêm phần tử từ Queue:

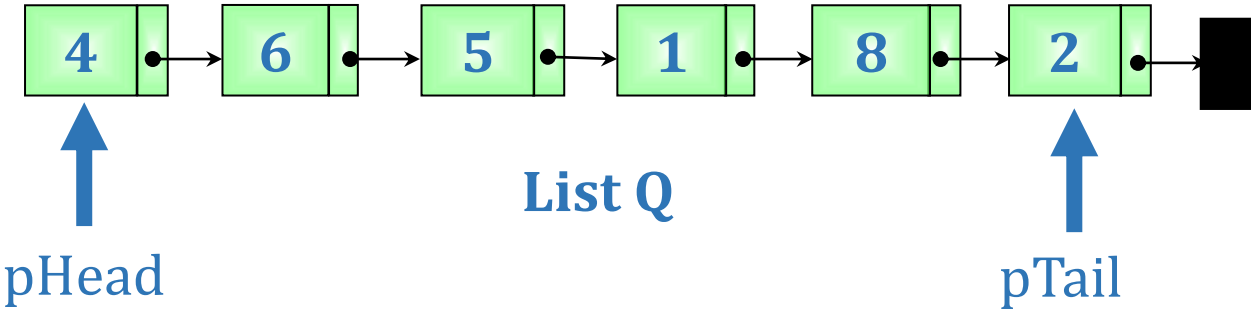


Queue Enqueue

```
void EnQueue(Queue &q, int x)
{
  int i,f,r;
  if(q.Rear-q.Front+1==N)
    printf("Queue is full");
  else
  {
    if(q.Front==-1)
    {
      q.Front=0;
      q.Rear=-1;
    }
    if(q.Rear==N-1)
    {
      f=q.Front;
      r=q.Rear;
      for(i=f;i<=r;i++)
        q.a[i-f]=q.a[i];
      q.Front=0;
      q.Rear=r-f;
    }
    q.Rear++;
    q.a[q.Rear]=x;
  }
}
```

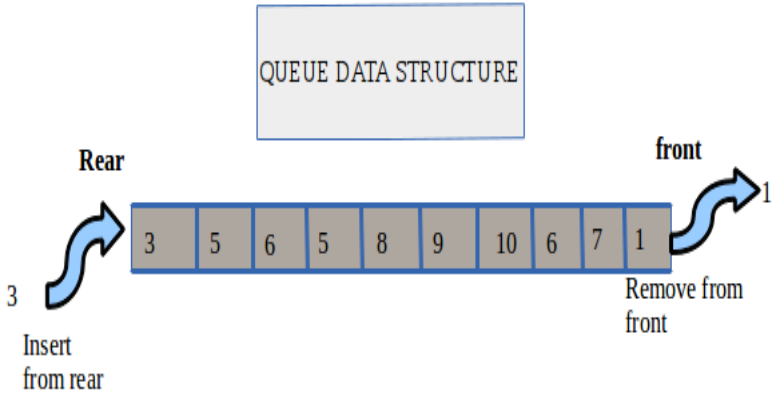


❖ **Cài đặt Queue (hang đợi) dùng danh sách liên kết:**



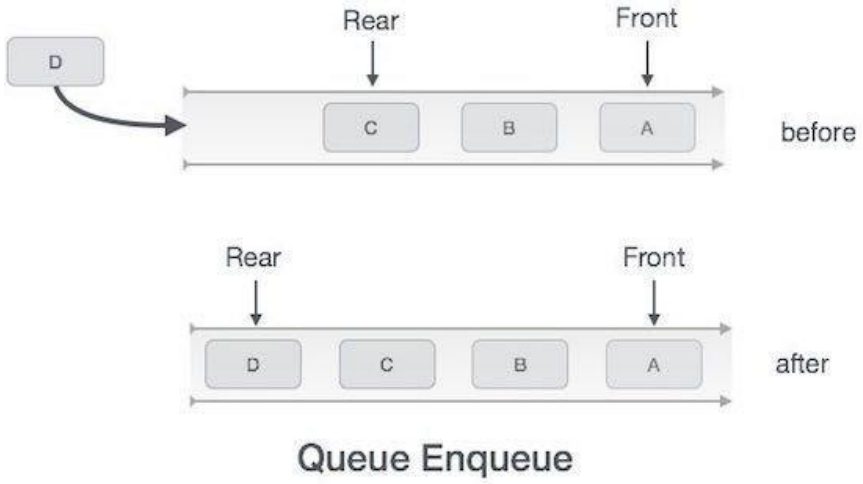
❖ **Kiểm tra Queue rỗng:**

```
int IsEmpty(List &Q)
 {
    if(Q.pHead==NULL)
      return 1;
    else
      return 0;
 }
```



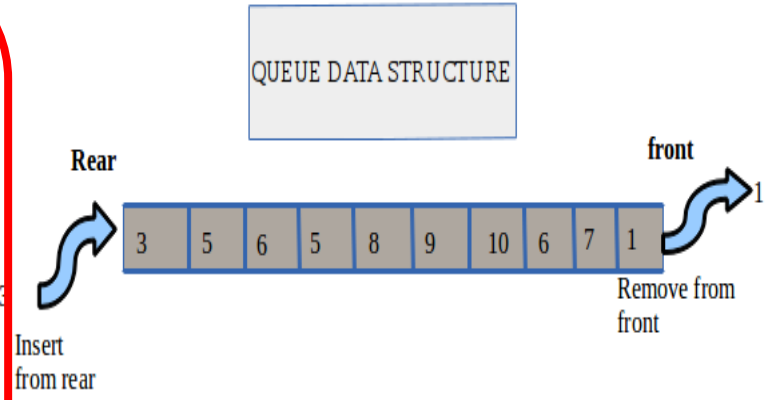
Cài đặt Queue (hang đợi) dùng danh sách liên kết:

❖ Thêm một phần tử vào Queue:



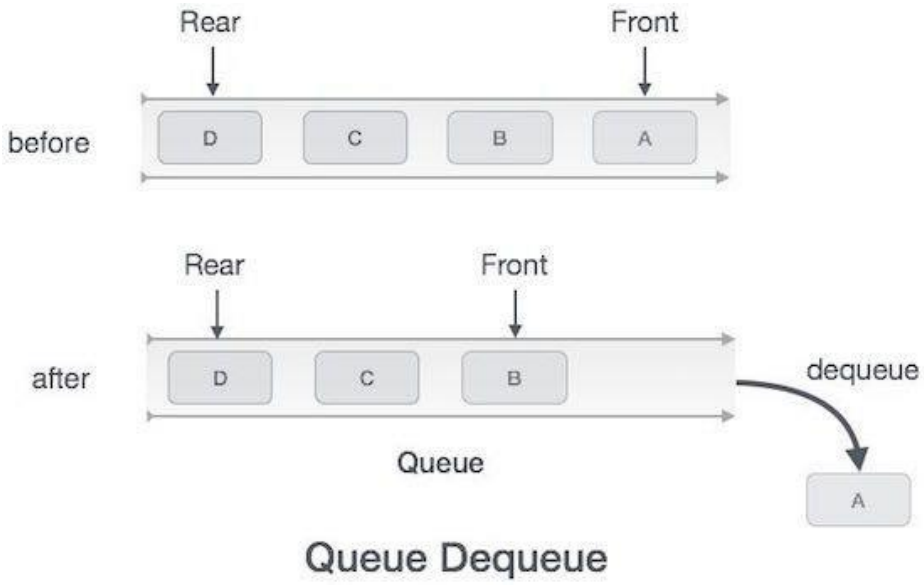
Queue Enqueue

```
void EnQueue(List &Q, Node *Tam)
{
  if(Q.pHead==NULL)
  {
    Q.pHead=Tam;
    Q.pTail=Tam;
  }
  else
  {
    Q.pTail->Next=tam;
    Q.pTail=tam;
  }
}
```



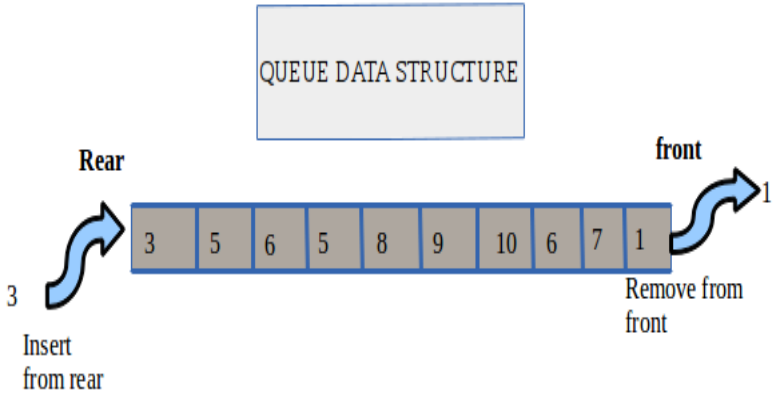
Cài đặt Queue (hang đợi) dùng danh sách liên kết:

❖ Lấy một phần tử ra khỏi Queue:



Queue Dequeue

```
int DeQueue(List &Q,int &trave)
{
  Node *p;
  if(IsEmpty(Q)!=1)
  {
    if(Q.pHead!=NULL)
    { p=Q.pHead;
      trave=p->Info;
      Q.pHead=Q.pHead->Next;
      if(Q.pHead==NULL)
        Q.pTail=NULL;
      return 1;
      delete p;
    }
  }
  return 0;
}
```

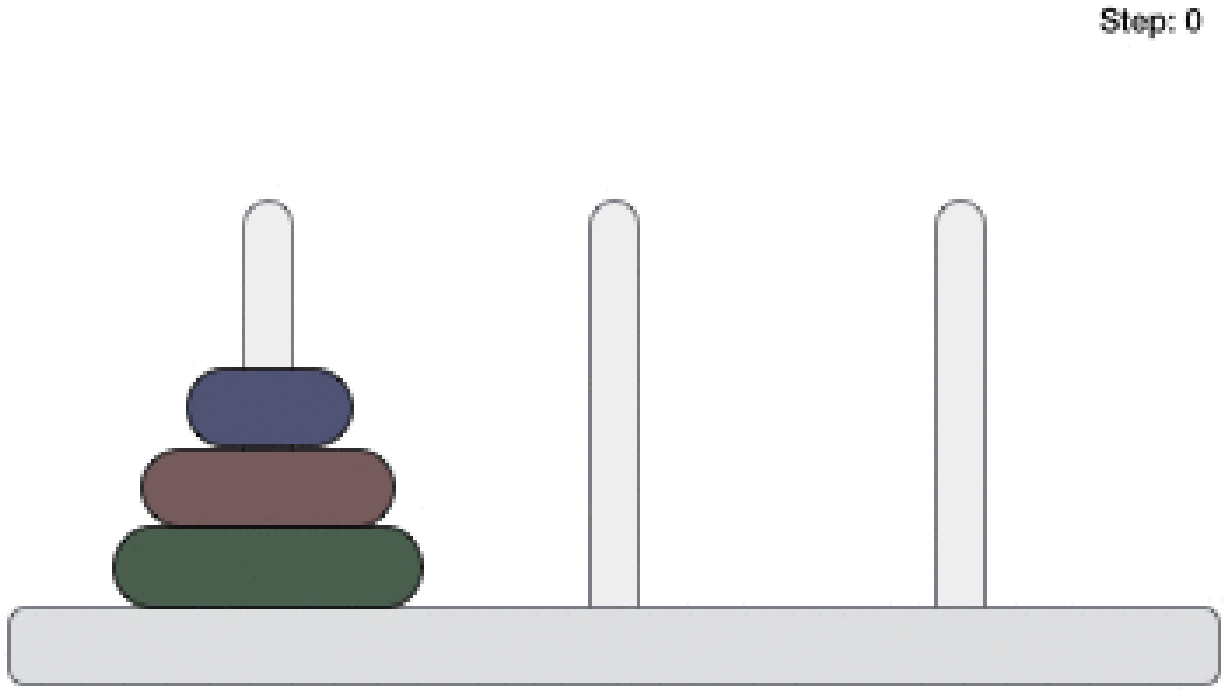


Ví dụ về Stack và Queue:

❖ Tháp Hà Nội (Tower of Hanoi) là gì ?

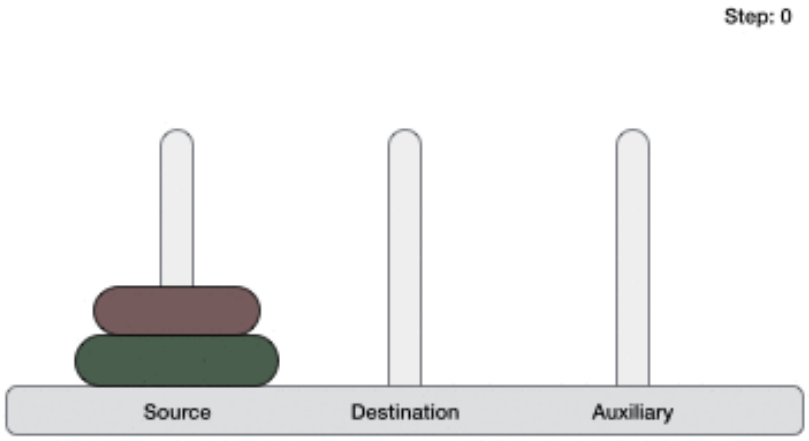
Bài toán Tháp Hà Nội (Tower of Hanoi) là một trò chơi toán học bao gồm 3 cột và với số đĩa nhiều hơn 1.

Bài toán Tháp Hà Nội với số đĩa là n có thể được giải với số bước tối thiểu là 2^n-1. Do đó, với trường hợp 3 đĩa, bài toán Tháp Hà Nội có thể được giải sau $2^3-1 = 7$ bước



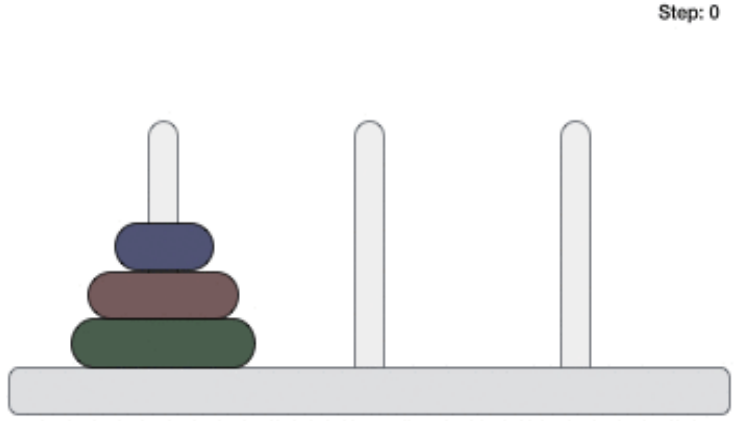
Thuật giải cho bài toánTháp Hà Nội:

- Để viết giải thuật cho trò chơi toán học Tháp Hà Nội (Tower of Hanoi), đầu tiên chúng ta cần tìm hiểu cách giải bài toán với số đĩa là 1 và 2. Chúng ta gán 3 cột với các tên là:
  - ✓ **cotNguon**: cột ban đầu chứa các đĩa
  - ✓ **cotDich**: cột cần di chuyển các đĩa tới
  - ✓ **cotTrungGian**: cột trung gian có mục đích làm trung gian trong quá trình di chuyển đĩa



Thuật giải cho bài toánTháp Hà Nội:

- Mục đích của chúng ta là di chuyển đĩa thứ n từ **cotNguon** tới **cotDich** và sau đó đặt tất cả (n-1) đĩa còn lại lên trên nó. Bây giờ chúng ta có thể tưởng tượng ra cách giải bài toán trên dựa vào đệ qui theo các bước sau:



Bước 1: Di chuyển n-1 đĩa từ **cotNguon** tới **cotTrungGian**

Bước 2: Di chuyển đĩa thứ n từ **cotNguon** tới **cotDich**

Bước 3: Di chuyển n-1 đĩa từ **cotTrungGian** về **cotDich**

Thuật giải cho bài toánTháp Hà Nội:

Bắt đầu giải thuật Tháp Hà nội Hanoi (disk, **cotNguon**, **cotDich**, **cotTrungGian**)

IF disk == 0, thì

di chuyển đĩa từ **cotNguon** tới **cotDich**

ELSE

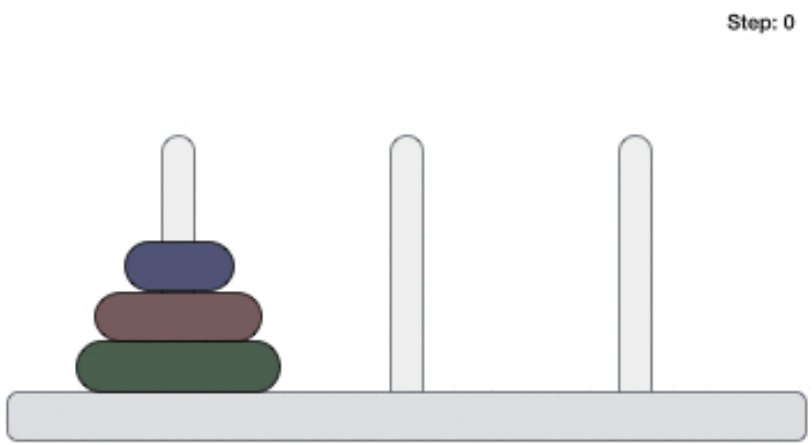
Hanoi(disk - 1, **cotNguon**, **cotTrungGian**, **cotDich**) // Bước 1

di chuyển đĩa từ **cotNguon** tới **cotDich** // Bước 2

Hanoi(disk - 1, **cotTrungGian**, **cotDich**, **cotNguon**) // Bước 3

Kết thúc IF

Kết thúc giải thuật



Câu hỏi Bài tập Danh sách liên kết kép

Bài tập 1: Viết chương trình cài đặt Stack (ngăn xếp) trên số nguyên dương theo cấu trúc mảng và cấu trúc danh sách liên kết đơn.

Bài tập 2: Viết chương trình cài đặt Queue (hang đợi) trên số nguyên dương theo cấu trúc mảng và cấu trúc danh sách liên kết đơn.

Bài tập 3: Viết chương trình nhập vào một số nguyên không âm bất kỳ, sau đó hiển thị lên màn hình số đảo ngược thứ tự của số nguyên vừa nhập vào (ví dụ: nếu nhập số 12567, hiển thị lên màn hình số 76521) bằng cách:

a) Sử dụng ngăn xếp.

b) Sử dụng hàng đợi.

Câu hỏi Bài tập Danh sách liên kết kép

Bài tập 4: Viết chương trình chuyển đổi một số nguyên N trong hệ thập phân (hệ 10) sang biểu diễn ở các hệ sau, sử dụng ngăn xếp:

a) Hệ nhị phân (hệ 2)

b) Hệ thập lục phân (hệ 16)

c) Hệ bát phân (hệ 8).

Bài tập 5: Hãy viết chương trình mô phỏng cho bài toán "Tháp Hà Nội" bằng cách sử dụng ngăn xếp