CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Data Structures & Algorithms

STACK - QUEUE

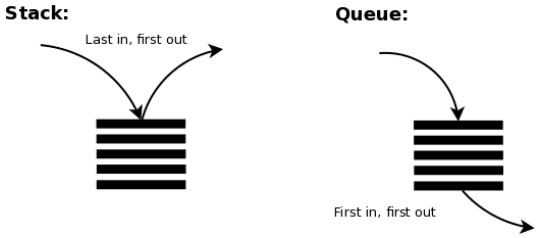
Nội dung

1. Kiểu danh sách
2. Danh sách liên kết đơn.
3. **Stack**
4. **Queue**

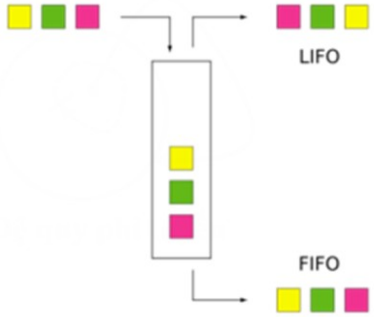
STACK - QUEUE

- **Stack** (ngăn xếp): Là 1 vật chứa các đối tượng làm việc theo cơ chế **LIFO (Last In First Out)**, từ 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**"
- **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**".

STACK - QUEUE



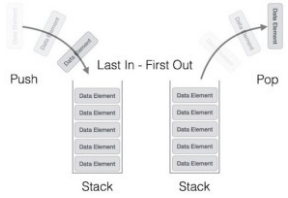
STACK - QUEUE



STACK - QUEUE

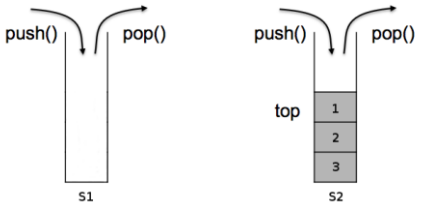
- **Stack**:
  - Trình biên dịch
  - Khử đệ qui đuôi
  - Lưu vết các quá trình quay lui, vét cạn.
- **Queue**:
  - Tổ chức lưu vết các quá trình tìm kiếm theo chiều rộng, và quay lui vét cạn,
  - Tổ chức quản lý và phân phối tiến trình trong các hệ điều hành,
  - Tổ chức bộ đệm bàn phím, …

STACK – Ngăn Xếp -Các thao tác



Push(x): Thêm đối tượng x vào Stack
Pop(): Lấy đối tượng từ Stack
isEmpty(): Kiểm tra Stack có rỗng hay không
Top(): Trả về giá trị của phần tử nằm đầu Stack mà không hủy nó khỏi Stack.

STACK – Ngăn Xếp -Các thao tác

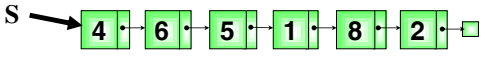


STACK – Cài đặt

- Dùng mảng 1 chiều

Data S[N];
int t;

- Dùng danh sách liên kết đơn



- **Thêm và hủy cùng phía**

STACK – Cài đặt – Mảng một chiều

- *Cấu trúc dữ liệu của Stack*

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

- *Khởi tạo Stack:*

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

STACK – Cài đặt – Mảng một chiều

```
int IsEmpty(Stack s)//Stack có rỗng hay không
{
 if(s.t==-1)
      return 1;
 else
      return 0;
}
int IsFull(Stack s) //Kiểm tra Stack có đầy hay không
{
 if(s.t>=max)
      return 1;
 else
      return 0;
}
```

STACK – Cài đặt – Mảng một chiều

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

STACK – Cài đặt – Mảng một chiều

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

STACK – Cài đặt – Mảng một chiều

```
int main()
{     Stack s;
      int x, tv,i;
      CreateStack(s);
      for(i=2;i<=5;i++)
            Push(s,i);
      tv=Pop(s,x);
      if(tv==1) cout<<"gia tri lay duoc tu Stack  "<< x;
      return 0;
}
```

STACK – Cài đặt – DSLK

```
 int IsEmpty(List s) // Kiểm tra stack rỗng
 {
      if(s.pHead==NULL)//Stack rong
            return 1;
      else
             return 0;
}
```

STACK – Cài đặt – DSLK

```
void Push(List &s, Node *Tam) //  Thêm 1 phần tử vào đầu stack
{
 if(s.pHead==NULL)
 {
      s.pHead=Tam;
      s.pTail=Tam;
 }
 else
 {
      Tam->pNext=s.pHead;
      s.pHead=Tam;
 }
}
```

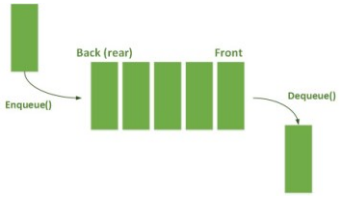
STACK – Cài đặt – DSLK

```
int Pop(List &s,int  &trave) // Lấy 1 phần tử từ stack
{
  Node *p;
  if(IsEmpty(s)!=1)
  {
            p=s.pHead;
            trave=p->Info;
            s.pHead=s.pHead->Next;
            if(s.pHead==NULL)
                  s.pTail=NULL;
            delete p;
            return 1;
  }
  return 0;
}
```

STACK – Cài đặt – DSLK

```
int main()
{     Stack s;
      int tv,i; Node *p;
      CreateStack(s);
      for(i=2;i<=5;i++)
      {     p=CreateNode(i);
            Push(s,p);
      }
      tv=Pop(s,x);
      if(tv==1)
            cout<<"gia tri lay duoc tu Stack "<< x;
      return 0;
}
```

QUEUE – Hàng Đợi - Các thao tác



FIFO
FIRST IN FIRST OUT

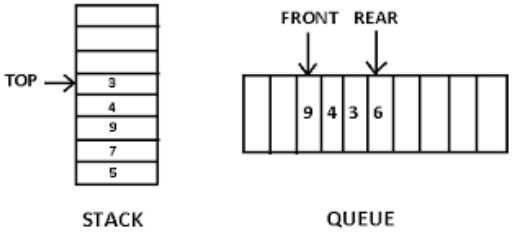
EnQueue(x): Thêm đối tượng x vào cuối hàng đợi.

DeQueue(): Lấy đối tượng ở đầu hàng đợi

isEmpty(): Kiểm tra xem hàng đợi có rỗng hay không?

Front(): Trả về giá trị của phần tử nằm đầu hàng đợi mà không hủy nó.

QUEUE – Hàng Đợi - Các thao tác

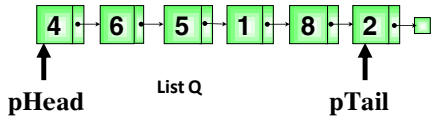


QUEUE – Cài đặt

- Dùng **mảng 1 chiều**


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

- Dùng **danh sách liên kết đơn**



- **Thêm và hủy** khác phía

QUEUE – Cài đặt – Mảng một chiều

- *Cấu trúc dữ liệu:*

```
struct Queue
{
   int a[max];
   int Front; //chỉ số của phần tử đầu trong Queue
   int Rear; //chỉ số của phần tử cuối trong Queue
};
```

- *Khởi tạo Queue rỗng*

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

QUEUE – Cài đặt – Mảng một chiều

```
int DeQueue(Queue &q,int &x) // Lấy một phần tử từ Queue
{
  if(q.Front!=-1)       //queue khong rong
  {
        x=q.a[q.Front];
        q.Front++;
        if(q.Front>q.Rear)//truong hop co mot phan tu
        {
              q.Front=-1;
              q.Rear=-1;
        }
        return 1;
  }
  else //queue trong
  {
        cout<<"Queue rong";
        return 0;
  }
}
```

QUEUE – Cài đặt – Mảng một chiều

```
int EnQueue(Queue &q ,int x) // Thêm một phần tử vào Queue
{
  int i;
  int f,r;
  if(q.Rear-q.Front+1==max)//queue bi day khong the them vao duoc nua
        return 0;
  else
  {
        if(q.Front==-1) // Truong hop queue chua co phan tu nao
        {
              q.Front=0;
              q.Rear=0;
        }
```

QUEUE – Cài đặt – Mảng một chiều

```
    if(q.Rear==max-1)//Queue đầy ảo
    {
        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;
    return 1;
  }
}
```

QUEUE – Cài đặt – Mảng một chiều

```
int main()
{   Queue q;
    int tv,i,x;
    CreateQueue(q);
    for(i=2;i<=5;i++)
        EnQueue(q,i);
    tv=DeQueue(q,x);
    if(tv==1)
        cout<<"gia tri lay duoc tu Queue "<<x;
    return 0;
}
```

QUEUE – Cài đặt – DSLK

- Kiểm tra **Queue có rỗng**?

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

QUEUE – Cài đặt – DSLK

```
void EnQueue(List &Q, Node *Tam) // Thêm một phần tử vào Queue
{
  if(Q.pHead==NULL)
  {
    Q.pHead=Tam;
    Q.pTail=Tam;
  }
  else
  {
    Q.pTail->Next=tam;
    Q.pTail=tam;
  }
}
```

QUEUE – Cài đặt – DSLK

```
int DeQueue(List &Q,int &X) // Lấy 1 phần tử từ Queue
{
  Node *p;
  if(IsEmpty(Q)!=1)
  {
    p=Q.pHead;
    X=p->Info;
    Q.pHead=Q.pHead->Next;
    if(Q.pHead==NULL)
        Q.pTail=NULL;
    delete p;
    return 1;
  }
  return 0;
}
```

QUEUE – Cài đặt – DSLK

```
int main()
{   Queue q;
    int tv,i; Node *p;
    CreateQueue(q);
    for(i=2;i<=5;i++)
    {   p=CreateNode(i);
        EnQueue(q,p);
    }
    tv=DeQueue(q,x);
    if(tv==1)
        cout<<"gia tri lay duoc tu Queue "<< x;
    return 0
}
```

Slide được tham khảo từ

- **Slide được tham khảo từ:**
  - Slide CTDL GT, Khoa Khoa Học Máy Tính, ĐHCNTT
  - Congdongcviet.com
  - Cplusplus.com

31

32