1. Cấu trúc dữ liệu danh sách

Mục tiêu

- Mô tả cấu trúc danh sách
- Mô tả cấu trúc tự tham chiếu
- Giải thích các loại danh sách liên kết
- Danh sách liên kết đơn
- Danh sách Thông tư
- Danh sách liên kết đôi
- Danh sách trong java.util

Liệt kê cấu trúc dữ liệu

- Danh sách là một cấu trúc dữ liệu tuần tự, tức là nó là một chuỗi các mục thuộc một loại cơ sở nhất định, trong đó các mục có thể được thêm, xóa và lấy ra từ bất kỳ vị trí nào trong danh sách.
- Danh sách có thể được triển khai dưới dạng mảng hoặc mảng động để tránh áp đặt kích thước tối đa.
- Một cách triển khai khác là danh sách liên kết, trong đó các mục được lưu trữ trong các nút được liên kết với nhau bằng con trỏ. Hai cách triển khai này có những đặc điểm rất khác nhau.
- Các giá trị có thể có của loại này là chuỗi các mục thuộc loại BaseType (bao gồm cả chuỗi có độ dài bằng 0). Hoạt động của ADT là:

```
getFirst(), getLast(), getNext(p), getPrev(p),
get(p), set( p,x), Insert(p,x),
Remove(p),removeFirst(), RemoveLast(),
RemoveNext(p), RemovePrev(p), find(x),size()
```

Hạn chế của mảng

- Mảng là một cấu trúc dữ liệu rất hữu ích trong nhiều trư ờng hợp. Tuy nhiên, nó có một số hạn chế quan trọng:

  - Họ yêu cầu thông tin kích thư ớc để tạo
  - Chèn một phần tử vào giữa mảng dẫn đến việc di chuyển các yếu tố khác xung quanh
  - Xóa một phần tử ở giữa mảng sẽ dẫn đến việc di chuyển các yếu tố khác xung quanh

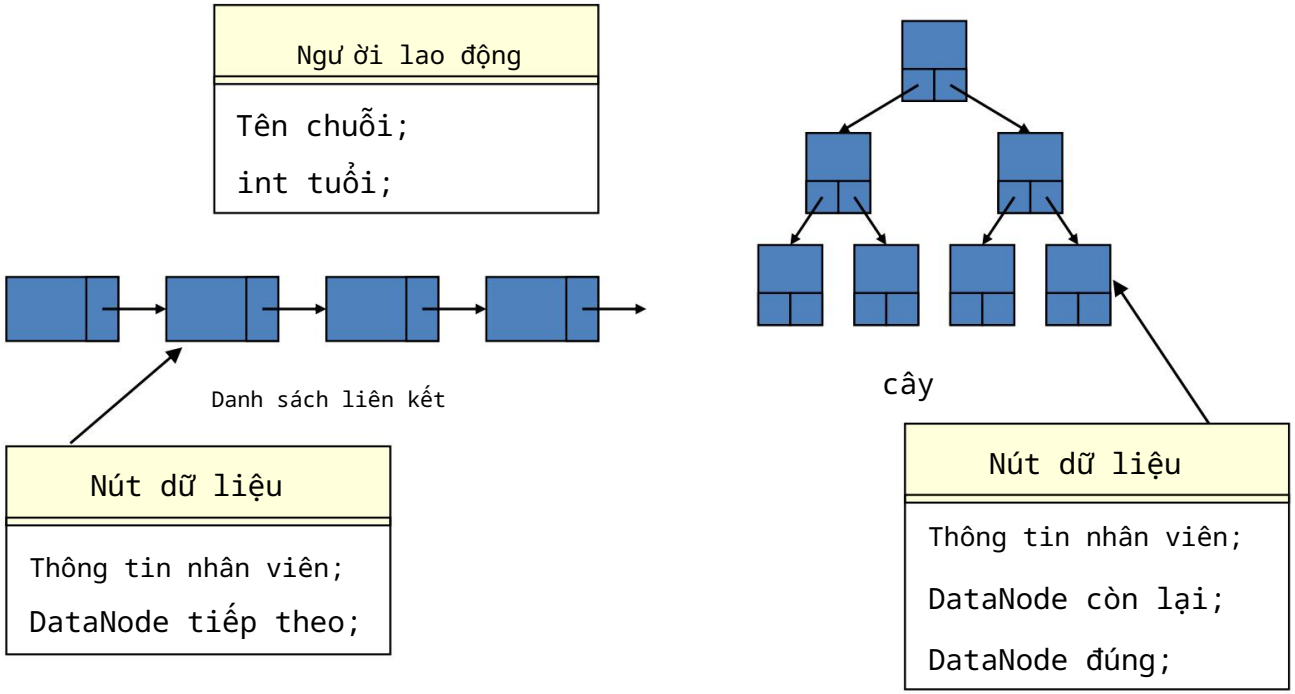
- Các cấu trúc dữ liệu khác hiệu quả hơn trong những tình huống như vậy.

Cấu trúc tự tham chiếu

Nhiều cấu trúc dữ liệu động đư ợc triển khai thông qua việc sử dụng cấu trúc tự tham chiếu.

Cấu trúc tự tham chiếu là một đối tư ợng, một trong các phần tử của nó là tham chiếu đến một đối tư ợng khác cùng loại với nó.

Với sự sắp xếp này, có thể tạo ra các 'chuỗi' dữ liệu có nhiều dạng khác nhau:



Cấu trúc tự tham chiếu

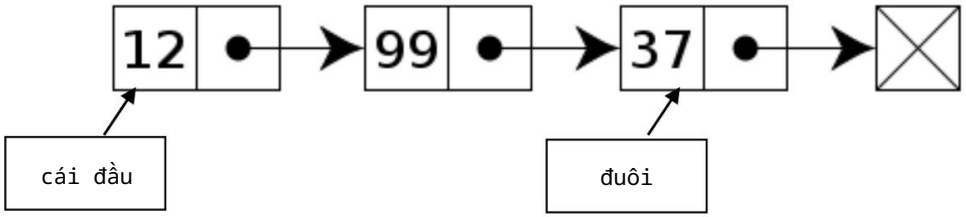
Danh sách liên kết

- Cấu trúc liên kết là tập hợp các nút lưu trữ dữ liệu và liên kết đến các nút khác
- Danh sách liên kết là một cấu trúc dữ liệu tuyến tính bao gồm các nút, mỗi nút chứa một số thông tin và một tham chiếu đến một nút khác trong danh sách
- Các loại danh sách liên kết:
  - Danh sách liên kết đơn
  - Danh sách liên kết đôi

Trong lập trình, tuyến tính có nghĩa là chúng được mô tả bằng một chuỗi (duy nhất) dữ liệu. tức là.
Mỗi mục dữ liệu có nhiều nhất một mục dữ liệu trước và nhiều nhất một mục dữ liệu kế tiếp.
Và, Phi tuyến tính có nghĩa là bất cứ điều gì khác.
Tuyến tính là - Mảng, Danh sách liên kết, Ngăn xếp, Hàng đợi. Phi tuyến tính là - Cây, Đồ thị

Danh sách liên kết đơn

Danh sách liên kết đơn là danh sách có nút bao gồm hai trư ờng dữ liệu: thông tin và tiếp theo. Trư ờng thông tin đư ợc sử dụng để lư u trữ thông tin và điều này rất quan trọng đối với ngư ời dùng. Trư ờng tiếp theo đư ợc sử dụng để liên kết với trư ờng kế tiếp của nó trong chuỗi này Hình ảnh sau đây mô tả một danh sách liên kết số nguyên đơn giản.



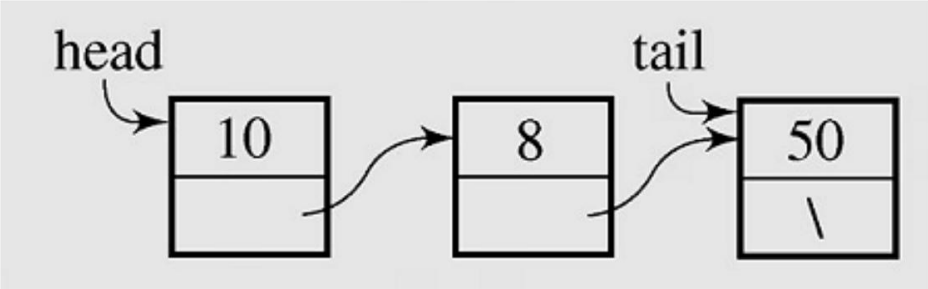
Danh sách liên kết đơn

DREAM OF INNOVATION

Triển khai danh sách liên kết đơn

```
lớp Node
 {thông tin int;
  Nút tiếp theo;
  Nút() {}
  Nút(int x, Nút p)
   {info=x;next=p; } }
```

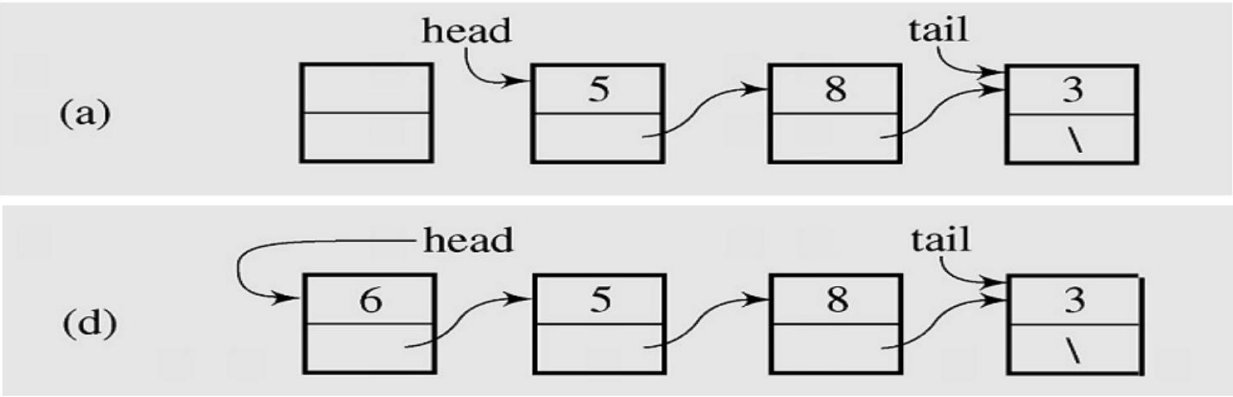
```
lớp Danh sách của tôi
 {Nút đầu, đuôi;
  MyList()
{head=tail=null;}
  boolean isEmpty()
   {return(head==null); }

  void clear()
   {head=tail=null; }
```




```
void add(int x)
   { if(isEmpty())
     head=tail=Nút mới(x,null); khác
     {Nút
      q = Nút mới(x,null); tail.next=q;
       đuôi=q; } } void đi


  qua()
   {Nút p=đầu; trong
    khi(p!=null)
     {System.out.print(" " + p.info);
       p=p.next; }

    System.out.println(); }

  Tìm kiếm nút(int x) {...}
   void dele(int x) {...} }
```

Danh sách liên kết đơn - 1

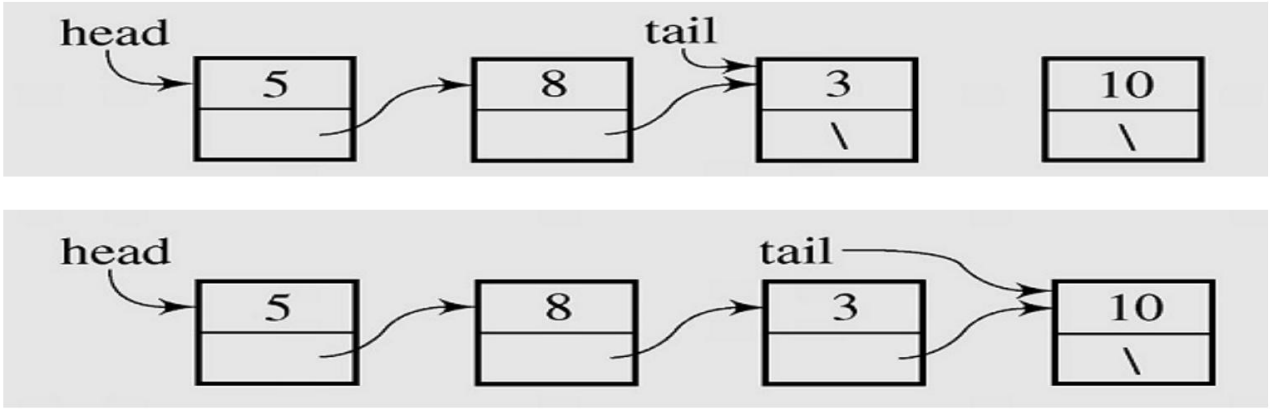
Chèn một nút mới vào đầu danh sách



Chèn một nút mới vào đầu Danh sách liên kết đơn

Danh sách liên kết đơn - 2

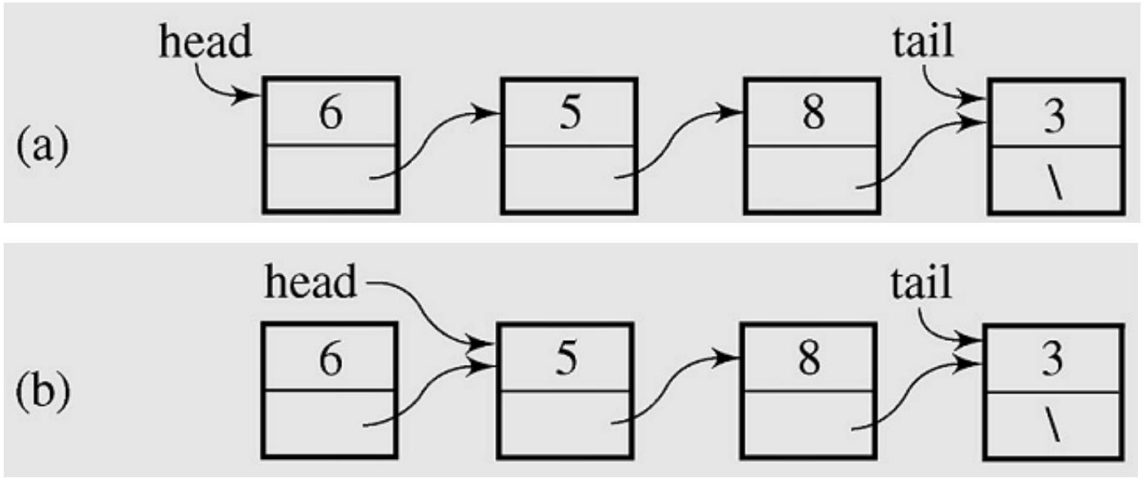
Chèn một nút mới vào cuối danh sách



Chèn một nút mới vào cuối Danh sách liên kết đơn

Danh sách liên kết đơn - 3

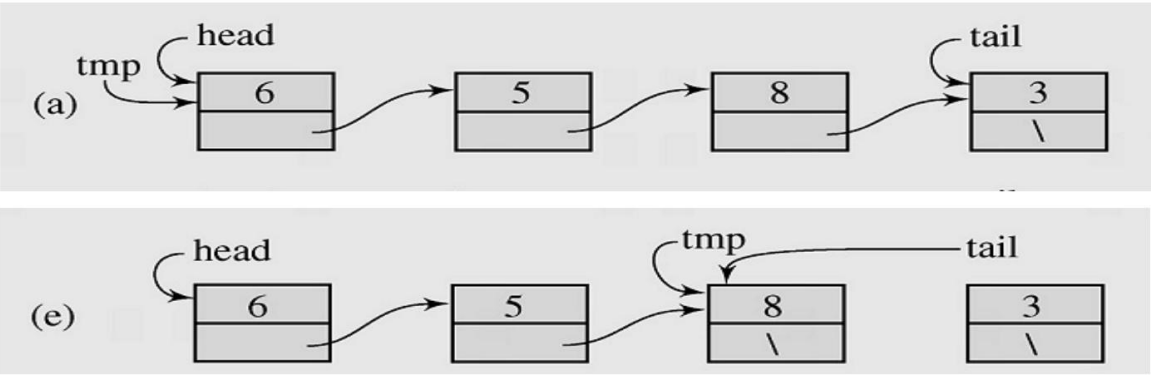
Xóa một nút khỏi đầu danh sách



Xóa một nút khỏi đầu Danh sách liên kết đơn

Danh sách liên kết đơn - 4

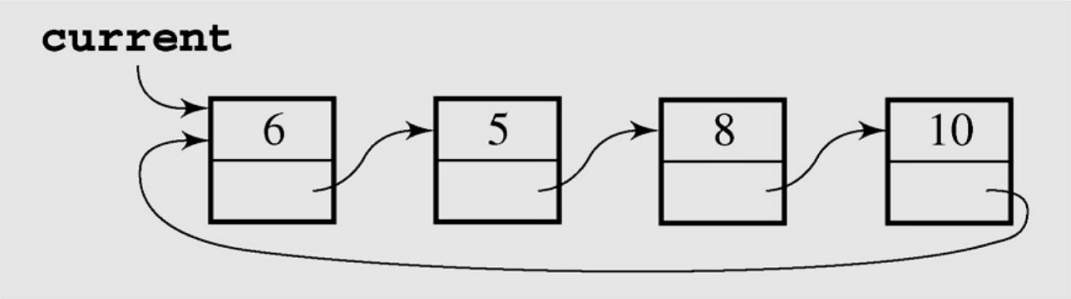
Xóa phần tử ở cuối danh sách



Xóa một nút khỏi cuối Danh sách liên kết đơn

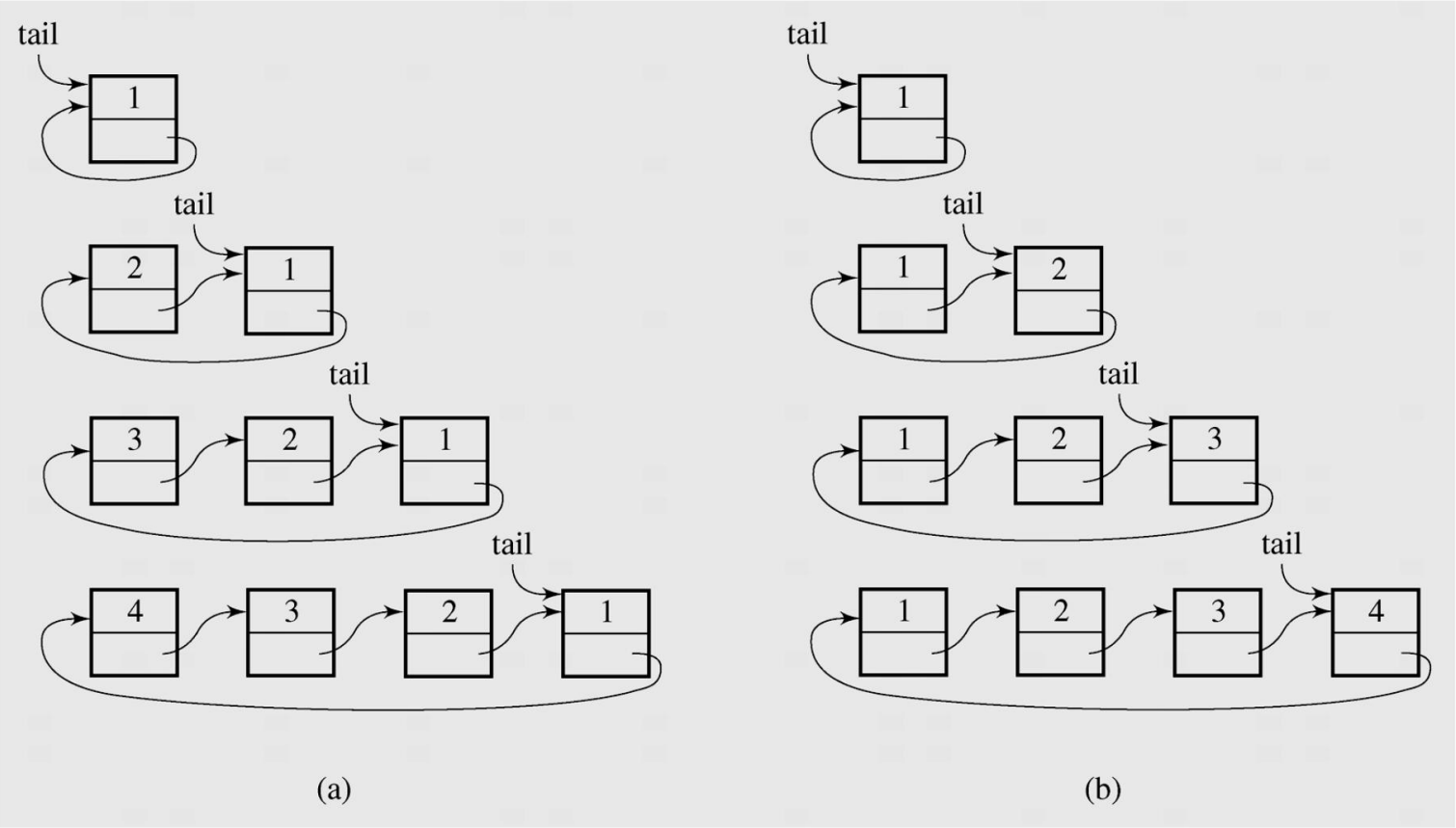
Danh Sách Thông Tư - 1

- Danh sách vòng là khi các nút tạo thành một vòng:
  Danh sách là hữu hạn và mỗi nút có một nút kế tiếp



Danh sách liên kết đơn tròn

Chèn nút



Chèn các nút ở phía trư ớc danh sách liên kết đơn tròn (a) và ở cuối danh sách (b)

Ứng dụng danh sách thông tư

1. Lập lịch quay vòng

Một trong những vai trò quan trọng nhất của hệ điều hành là quản lý nhiều quy trình hiện đang hoạt động trên máy tính, bao gồm cả việc lập lịch trình cho các quy trình đó trên một hoặc nhiều bộ xử lý trung tâm (CPU). Để hỗ trợ khả năng đáp ứng của số lư ợng quy trình đồng thời tùy ý, hầu hết các hệ điều hành đều cho phép các quy trình chia sẻ hiệu quả việc sử dụng CPU, sử dụng một số dạng thuật toán đư ợc gọi là lập lịch vòng tròn . Một tiến trình đư ợc cho một lư ợt ngắn để thực thi, đư ợc gọi là lát cắt thời gian, như ng nó bị gián đoạn khi lát cắt kết thúc, ngay cả khi công việc của nó chư a hoàn thành.

Mỗi tiến trình đang hoạt động đư ợc cung cấp một lát thời gian riêng, lần lư ợt theo thứ tự tuần hoàn.

2. Sử dụng danh sách liên kết vòng để triển khai Lập lịch quay vòng

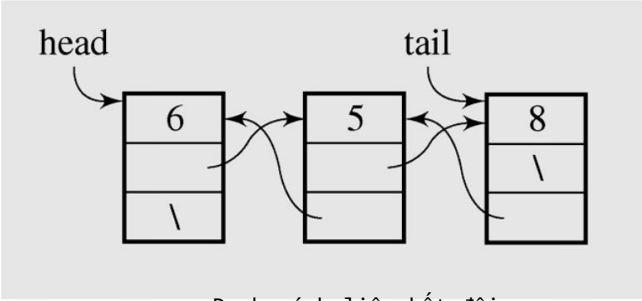
Chúng ta có thể sử dụng danh sách liên kết vòng để triển khai Lập lịch vòng tròn theo phư ơng pháp sau: xoay ( ): Di chuyển phần tử đầu tiên đến cuối danh sách.

Với thao tác mới này, việc lập kế hoạch quay vòng có thể đư ợc triển khai một cách hiệu quả bằng cách thực hiện lặp lại các bư ớc sau trên danh sách liên kết vòng C:

1. Cho một lát thời gian để xử lý C.first( )
2. C.rotate( )

Danh sách liên kết đôi - 1

Trong danh sách liên kết đôi, mỗi nút có hai trường tham chiếu, một cho nút kế tiếp và một cho nút trước đó



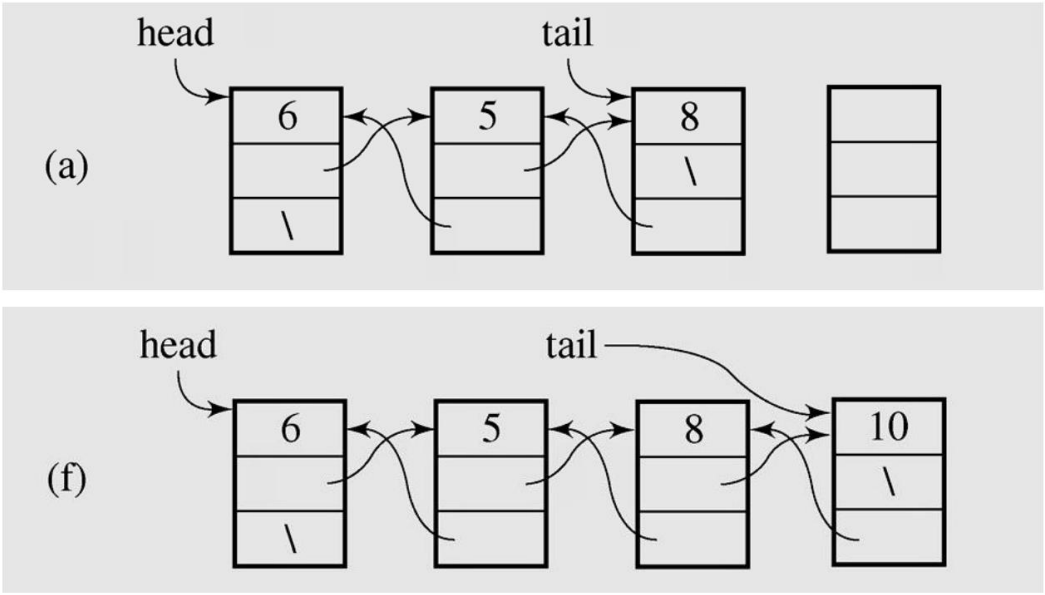
Danh sách liên kết đôi

```
lớp Node
 {thông tin int;
  Nút trước, tiếp theo;
  Nút() {}
  Nút(int x, Nút p, Nút q)
   {info=x;prev=p; tiếp

 theo=q; } }
```

```
lớp MyList
 {Node đầu, đuôi;
  MyList() {head=tail=null;}
  boolean isEmpty()
   {return(head==null); }
  void clear() {head=tail=null;} void
  add(int x)
   {if(isEmpty())
     head=tail=New Node(x,null,null); khác
     {Nút
     q = Nút mới(x,tail,null); tail.next=q;
      đuôi=q; } }

  ...
 }
```

Danh sách liên kết đôi - 2

Thêm một nút mới vào cuối

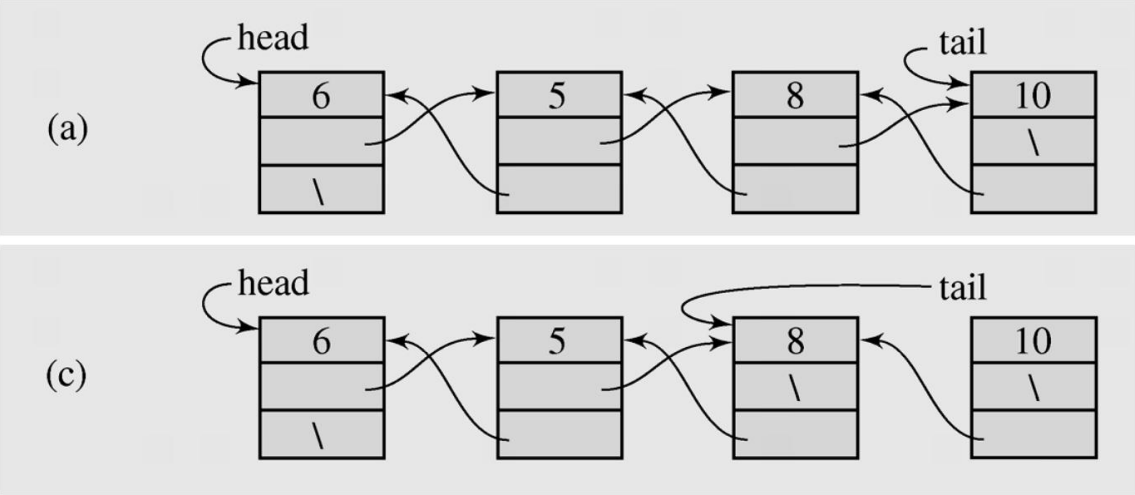


Thêm nút mới vào cuối Danh sách liên kết đôi

DREAM OF INNOVATION

Danh sách liên kết đôi -3

Xóa một nút từ cuối



Xóa một nút ở cuối Danh sách liên kết đôi

DREAM OF INNOVATION

Danh sách trong java.util - Lớp LinkedList

boolean add(E o)	Nối phần tử đã chỉ định vào cuối danh sách này.
void addFirst(E o)	Chèn phần tử đã cho vào đầu danh sách này.
void addLast(E o)	Nối phần tử đã cho vào cuối danh sách này.
khoảng trống rõ ràng()	Loại bỏ tất cả các phần tử khỏi danh sách này.
E get(int index)	Trả về phần tử tại vị trí đã chỉ định trong danh sách này.
E getFirst()	Trả về phần tử đầu tiên trong danh sách này.
E getLast()	Trả về phần tử cuối cùng trong danh sách này.
E xóa(int chỉ mục)	Xóa phần tử tại vị trí đã chỉ định trong danh sách này.
E RemoveFirst()	Xóa và trả về phần tử đầu tiên từ danh sách này.
E loại bỏLast()	Loại bỏ và trả về phần tử cuối cùng trong danh sách này.
kích thước int ()	Trả về số phần tử trong danh sách này.
Sự vật[] toArray()	Trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.

DREAM OF INNOVATION

Danh sách trong

java.util Ví dụ về lớp LinkedList

```
nhập java.util.*;
nút lớp { Tên
  chuỗi; int tuổi;
    Nút() {}

    Nút(Tên chuỗi1, int age1)
      { name=name1; tuổi=tuổi1; }

    void set(String name1, int age1)
      { name=name1; tuổi=tuổi1; }

    Chuỗi công khai toString()
      { Chuỗi s = tên+" "+tuổi; (các)
        trả lại; }

  }
```

```
class Main

  { public static void main(String [] args)

      { LinkedList t = new LinkedList();
        Nút x; int n,i;
        x = Nút mới("A01",25); t.add(x); x =
        Nút mới("A02",23); t.add(x); x = Nút
        mới("A03",21); t.add(x);
        for(i=0;i<t.size();i++)
       System.out.println(t.get(i)); }

  }
```

DREAM OF INNOVATION

Danh sách trong java.util - Lớp ArrayList

boolean add(E o)	Nối phần tử đã chỉ định vào cuối danh sách này.
void add(int chỉ mục, E o)	Chèn phần tử đã cho vào vị trí đã chỉ định.
khoảng trống rõ ràng()	Loại bỏ tất cả các phần tử khỏi danh sách này.
E get(int index)	Trả về phần tử tại vị trí đã chỉ định trong danh sách này.
E xóa(int chỉ mục)	Loại bỏ phần tử tại vị trí đã chỉ định trong này danh sách.
kích thước int ()	Trả về số phần tử trong danh sách này.
void đảm bảoCapacity(int minCapacity)	Tăng dung lượng của ArrayList này Ví dụ, nếu cần, để đảm bảo rằng nó có thể chứa ít nhất số phần tử được chỉ định bởi đối số dung lượng tối thiểu.
làm mất hiệu lực TrimToSize()	Cắt giảm dung lượng của phiên bản ArrayList này thành danh sách Kích cỡ hiện tại.
Sự vật[] toArray()	Trả về một mảng chứa tất cả các phần tử trong danh sách này theo đúng thứ tự.

DREAM OF INNOVATION

Bản tóm tắt

- Danh sách là một cấu trúc dữ liệu tuần tự, tức là nó là một chuỗi các mục thuộc một kiểu cơ sở nhất định.
- Danh sách có thể được triển khai dưới dạng mảng hoặc mảng động để tránh áp đặt kích thước tối đa. • Một cách triển khai thay thế là danh sách liên kết , nơi các mục được lưu trữ trong các nút được liên kết với nhau bằng con trỏ. • Danh sách liên kết đơn là khi một nút có liên kết tới chỉ kế nhiệm (nút tiếp theo).
- Danh sách vòng là khi các nút tạo thành một vòng: Danh sách là hữu hạn và mỗi nút có một nút kế tiếp.
- Danh sách liên kết đôi là khi một nút có liên kết tới nút trước đó và nút tiếp theo.

DREAM OF INNOVATION

Đọc sách ở nhà

Sách giáo khoa: Cấu trúc dữ liệu và thuật toán trong Java