Danh Sách Liên Kết
Đôi & Vòng

Mục tiêu

Bài thực hành này giúp các em nắm vững về cách tổ chức và sử dụng cấu trúc dữ liệu DSLK đôi và vòng để hiện thực và giải quyết các bài toán trong thực tế, nhất là các bài toán cần lưu trữ dữ liệu có kích thước lớn.

Nội dung

Câu 1: Hãy xây dựng một lớp danh sách liên kết đôi DoublyLinkedList<E> (kiểu Generic) và hiện thực các phương thức của lớp:

- int getSize(): cho biết số phần tử đang có trong danh sách
- boolean isEmpty(): cho biết danh sách hiện tại có rỗng hay không ?
- E first(): trả về giá trị của phần tử ở đầu danh sách
- E last(): trả về giá trị của phần tử ở cuối danh sách
- Node<E> search(E e): tìm một Node trong danh sách đang có chứa giá trị e. Kết quả trả về của phương thức này là chính Node đó (nếu tìm được). Nếu Node không tồn tại trong danh sách thì kết quả trả về của phương thức là **NULL**
- void insertFirst(E e): thêm một Node mới có giá trị dữ liệu là e vào đầu danh sách
- void insertLast(E e): thêm một Node mới có giá trị dữ liệu là e vào cuối danh sách
- void insertAfter(E v, E x): thêm một Node mới có giá trị dữ liệu là v vào sau một Node có giá trị x trong danh sách. Nếu Node x không tồn tại thì phần tử mới sẽ được thêm vào **cuối** danh sách
- void insertBefore(E v, E y): thêm một Node mới có giá trị dữ liệu là v vào trước một Node có giá trị y trong danh sách. Nếu Node y không tồn tại thì phần tử mới sẽ được thêm vào **đầu** danh sách
- void removeFirst(): xóa bỏ Node ở đầu danh sách
- void removeLast(): xóa bỏ Node ở cuối danh sách
- void remove(E e): xóa bỏ Node đầu tiên có giá trị là e trong danh sách
- void removeAll(E e): xóa bỏ tất cả các Node có giá trị là e trong danh sách
- void swapNode(Node<E> nodeA, Node<E> nodeB): hoán vị/hoán đổi vị trí của hai Node A và B trong danh sách

- swapValue(E a, E b): hoán vị/ hoán đổi giá trị của hai Node đang có chứa giá trị lần lượt là a và b trong danh sách. Nếu không đồng thời tìm ra được 2 Node như vậy trong danh sách thì phương thức này không cần thực hiện.

Câu 2: Nhập vào một DSLK đôi. Hãy sắp xếp các phần tử trong danh sách này theo thứ tự tăng dần bằng cách giải thuật:

a/ Sắp xếp chèn (Insertion Sort)

b/ Sắp xếp Shellsort

c/ Sắp xếp vun đống (Heap Sort)

Câu 3: Nhập vào một DSLK đôi có ít nhất 10 phần tử theo thứ tự tăng dần. Gọi Y là giá trị của một Node trong danh sách. Hãy cho biết trong danh sách trên:

a/ Có bao nhiêu Node có giá trị là số **chẵn** và lớn hơn Y

b/ Có bao nhiêu Node có giá trị là số **lẻ** và bé hơn Y

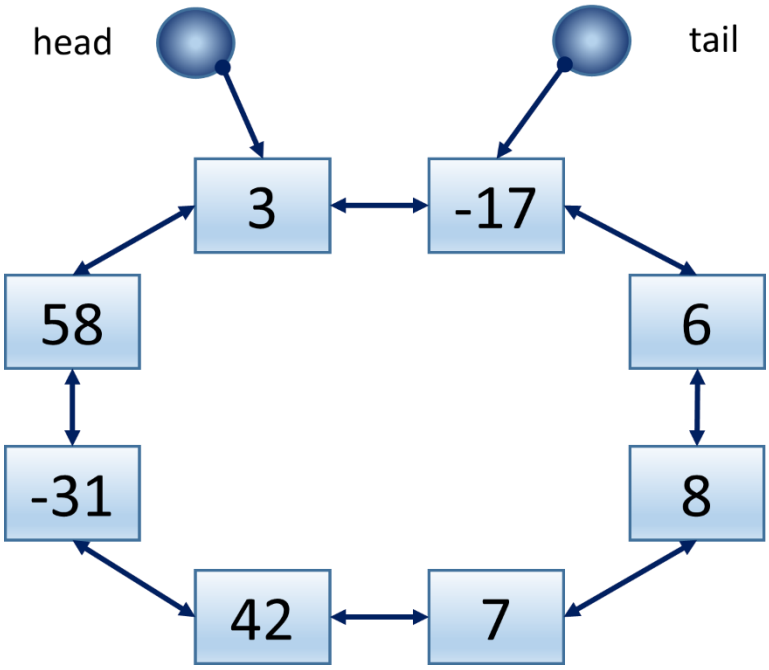
Câu 4: Hãy xây dựng một lớp danh sách liên kết vòng CircularlyLinkedList<E> (kiểu Generic) trên cơ sở danh sách liên kết đôi và hiện thực các phương thức của lớp:

- int getSize(): cho biết số phần tử đang có trong danh sách
- boolean isEmpty(): cho biết danh sách hiện tại có rỗng hay không ?
- Node<E> search(E e): tìm một Node trong danh sách đang có chứa giá trị e. Kết quả trả về của phương thức này là chính Node đó (nếu tìm được). Nếu Node không tồn tại trong danh sách thì kết quả trả về của phương thức là **NULL**
- void insertFirst(E e): thêm một Node mới có giá trị dữ liệu là e vào đầu danh sách
- void insertLast(E e): thêm một Node mới có giá trị dữ liệu là e vào cuối danh sách
- void insertAfter(E v, E x): thêm một Node mới có giá trị dữ liệu là v vào sau một Node có giá trị x trong danh sách. Nếu Node x không tồn tại thì phần tử mới sẽ được thêm vào **cuối** danh sách
- void insertBefore(E v, E y): thêm một Node mới có giá trị dữ liệu là v vào trước một Node có giá trị y trong danh sách. Nếu Node y không tồn tại thì phần tử mới sẽ được thêm vào **đầu** danh sách
- void removeFirst(): xóa bỏ Node ở đầu danh sách
- void removeLast(): xóa bỏ Node ở cuối danh sách
- void remove(E e): xóa bỏ Node đầu tiên có giá trị là e trong danh sách

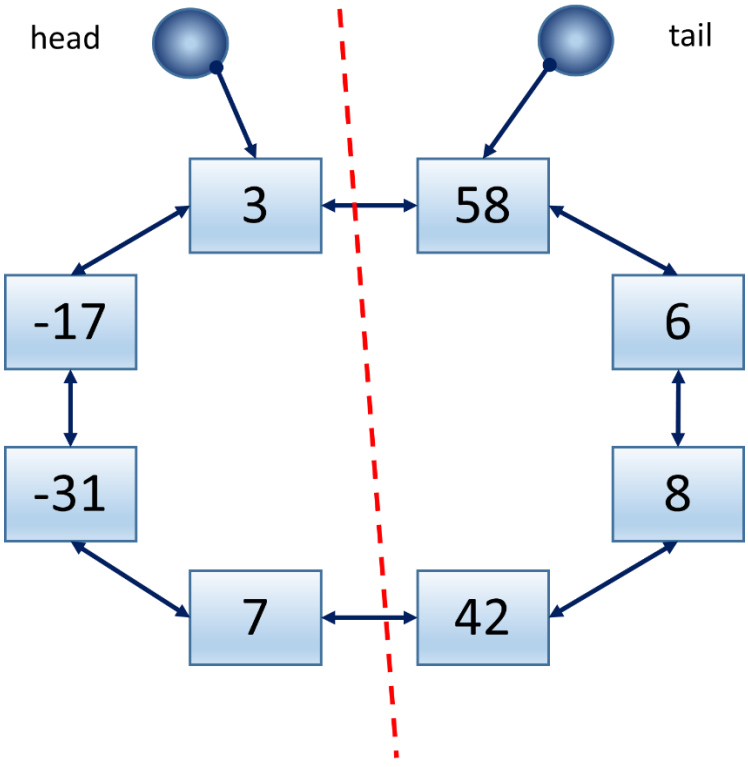
Câu 5: Xây dựng một danh sách liên kết vòng CircularlyLikedList<E> trên cơ sở danh sách liên kết đôi. Nhập vào các phần tử cho danh sách liên kết vòng nói trên (ít nhất 20 phần tử).

Hãy sắp xếp lại danh sách sao cho các phần tử có giá trị chẵn nằm ở một bên và các phần tử có giá trị lẻ nằm ở bên còn lại của danh sách.

Ví dụ: Cho một DSLK vòng với các phần tử có giá trị như sau:



Sau khi sắp xếp các phần tử, danh sách trên sẽ có dạng:



Trên đây là một cách sắp xếp để đưa các phần tử có giá trị **chẵn** nằm ở bên phải của vòng tròn và các phần tử có giá trị **lẻ** nằm ở bên trái của vòng tròn

Câu 6: Cộng hai số lớn

Một trong những công việc có độ phức tạp tính toán cao đối với con người nhưng lại có thể dễ dàng thực hiện được đối với máy tính đó chính là việc tính toán với dữ liệu có kích thước lớn. Giả sử chúng ta có 2 số nguyên dương, mỗi số có vài trăm chữ số, hãy tìm cách tính tổng giá trị của 2 số nguyên này.

Dữ liệu về các số nguyên dương này được cho trong một file có tên là "bignum.input". Hãy tạo ra một file có tên là "bignum.input" và nhập 2 số nguyên vào file này, mỗi số nằm trên 1 dòng liên tục. Sau đó, hãy đọc file để lấy giá trị 2 số nguyên này ra và thực hiện phép tính tổng. Kết quả tìm được hãy xuất ra một file khác có tên là "bignum.output"

Ví dụ 1:

Bignum.input	Bignum.output
11111111111111111111111111111 22222222222222222222222222222	33333333333333333333333333333

Ví dụ 2:

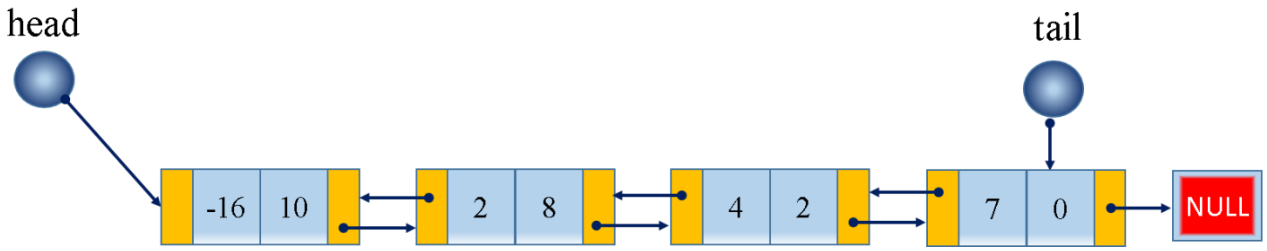
Bignum.input	Bignum.output
54389543857927343943024832 04 38219321937219379127312754 37	…

Hướng dẫn: vì đây là những số nguyên có số lượng chữ số rất lớn nên chúng ta không thể sử dụng các kiểu dữ liệu nguyên thủy (primitive type) có sẵn của Java hay bất cứ ngôn ngữ lập trình nào khác để lưu được. Như vậy để biểu diễn được những con số nguyên này trong máy tính, ta phải dùng một kiểu danh sách liên kết (đôi) có số lượng phần tử gần như không có giới hạn để lưu. Sau đó, ta thực hiện phép tính cộng trên hai danh sách liên kết này bằng cách cộng các chữ số ở hàng đơn vị, rồi hàng chục, rồi hàng trăm,…cho đến khi cộng hết 2 danh sách. Cần lưu ý là hai danh sách này có thể có độ dài khác nhau, và kết quả phép cộng hai số nguyên lớn cũng là một số nguyên lớn, vì vậy cần lưu kết quả này ra một danh sách liên kết nữa. Ngoài ra, khi ta thực hiện phép cộng thì ta bắt đầu cộng từ hàng đơn vị trở lên, nhưng khi xuất ra màn hình thì ta phải xuất từ chữ số ở hàng lớn nhất rồi dần dần mới trở về hàng đơn vị.

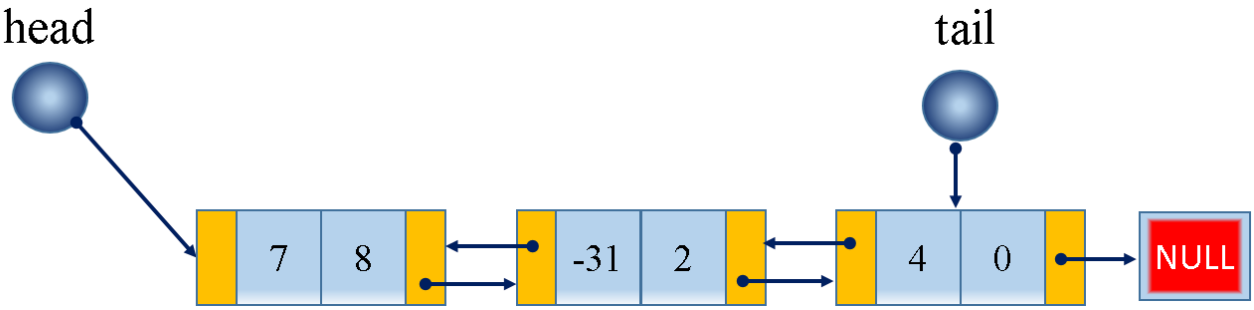
Câu 7: Cộng 2 đa thức

Một đa thức toán học có thể được biểu diễn bằng một danh sách liên kết.

Ví dụ 1: đa thức $A(x) = -16x^{10} + 2x^8 + 4x^2 + 7$ có thể được biểu diễn bằng danh sách liên kết đôi



Ví dụ 2: đa thức $B(x) = 7x^8 + (-31x^2) + 4$ thì có thể được biểu diễn bằng danh sách liên kết đôi



Yêu cầu: cho hai đa thức bất kì, hãy tính tổng của 2 đa thức đó và xuất kết quả ra màn hình

Dữ liệu vào được cho trong một file có tên là "Poly.input", dữ liệu ra được xuất trên màn hình là một đa thức đầy đủ, với phép lũy thừa được biểu diễn bằng kí tự '^'

Ví dụ 3: Tính tổng của 2 đa thức

$A(x) = -16x^{10} + 2x^8 + 4x^2 + 7$

$B(x) = 7x^8 + (-31x^2) + 4$

Poly.input	Console
-16 10 + 2 8 + 4 2 + 7 7 8 + -31 2 + 4	-16x^10 + 9x^8 + (-27x^2) + 11

Ví dụ 4: Tính tổng của 2 đa thức

$A(x) = 5x^4 – 9x^2 + 7$

$B(x) = -5x^4 - 2$

Poly.input	Console
5 4 + -9 2 + 7 -5 4 + -2	(-9x^2) + 5