CSE214 Data Structures

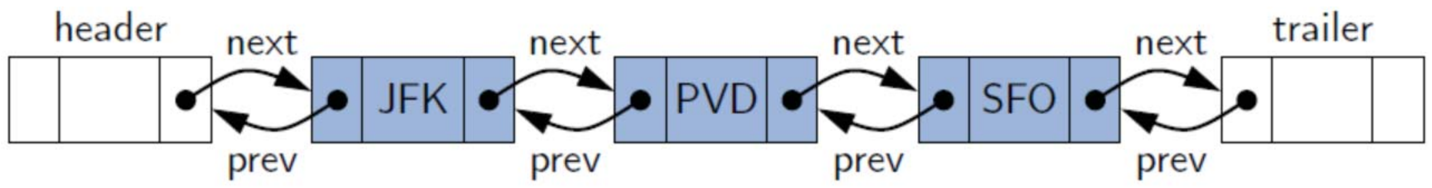
Linked Lists

YoungMin Kwon

Linked Lists

- Sentinels
  - Head and Tail are Nodes with links, but without data
  - They make the algorithm uniform

- Singly Linked List
  - Initialization: make head point to tail
- Doubly Linked List
  - Each node has next and prev links
  - Initialization: make head and tail point to each other



Selection Sort

- Selection sort:
  - For each index i in data[0…n]
  - Find min-index from data[i…n]
  - Swap data[i] and data [min-index]

- Demo
  - https://www.youtube.com/watch?v=Ns4TPTC8whw

```
public class LinkedList {
    //interface Ordered
    public static interface Ordered {
        public boolean ge(Ordered a);   //greater than or equal to
        …
    }

    //abstract class AbsList
    public static abstract class AbsList<E extends Ordered> {
        protected static interface Node<E extends Ordered> {
            E getElement();
            void setElement(E e);
        }

        //sentinels
        Node<E> head, tail;

        public AbsList() {
            head = makeNode(null);
            tail = makeNode(null);
            initList(head, tail);
        }
```

```
//add e to the first position
public void addFirst(E e) {
    //TODO: implement addFirst using addAfter
}

//add e to the last position
public void addLast(E e) {
    //TODO: implement addLast using addAfter and getPrev
}

//find the minimum node
protected Node<E> findMin(Node<E> from) {
    //TODO: implement findMin
    Node<E> min = from;
}

//selection sort
public void selSort() {
    //TODO: implement selSort using findMin
}

//insertion sort (backward: insert from the back to the front)
public void insSort() {
    //TODO: implement insSort
}
```

```
    //print the list
    public void print() { … }

    //swap the elements of the two nodes
    protected void swap(Node<E> a, Node<E> b) {
        E tmp = a.getElement();
        a.setElement(b.getElement());
        b.setElement(tmp);
    }

    //abstract methods
    protected abstract Node<E> makeNode(E e);
    protected abstract void initList(Node<E> head, Node<E> tail);
    protected abstract void addAfter(Node<E> node, Node<E> pos);
    protected abstract Node<E> getNext(Node<E> pos);
    protected abstract Node<E> getPrev(Node<E> pos);
}
```

```java
//singly linked list
public static class SglList<E extends Ordered> extends AbsList<E> {
    protected static class SglNode<E extends Ordered> implements Node<E> {
        E e;
        SglNode<E> next;

        SglNode(E e, SglNode<E> next) { this.e = e; this.next = next; }
        public E getElement()         { return e; }
        public void setElement(E e)   { this.e = e; }
    }

    //TODO: implement all abstract methods of AbsList
}
```

```
//doubly linked list
public static class DblList<E extends Ordered> extends AbsList<E> {
    protected static class DblNode<E extends Ordered> implements Node<E> {
        E e;
        DblNode<E> prev, next;

        DblNode(E e, DblNode<E> prev, DblNode<E> next) {
            this.e = e; this.prev = prev; this.next = next;
        }
        public E getElement()       { return e; }
        public void setElement(E e) { this.e = e; }
    }

    //TODO: implement all abstract methods of AbsList
}
```

```java
public static class Int implements Ordered {
    int n;
    public Int(int n)             { this.n = n; }
    public boolean ge(Ordered a) { return n >= ((Int)a).n; }
    public String toString()     { return "" + n; }
}

public static void test(String msg, AbsList<Int> list, boolean selSort) {
    System.out.println(msg);
    System.out.println("test add...");
    list.addFirst(new Int(1));
    list.addLast( new Int(2));
    …
    list.addLast( new Int(8));
    list.print();

    System.out.println("test sort...");
    if(selSort) list.selSort();
    else        list.insSort();
    list.print();

    System.out.println("test done...");
}
```

```
    public static void main(String[] args) {
        /* Expected output
            SglList
            test add...
            7 5 3 1 2 4 6 8
            test sort...
            1 2 3 4 5 6 7 8
            test done...
            DblList
            test add...
            7 5 3 1 2 4 6 8
            test sort...
            1 2 3 4 5 6 7 8
            test done...
        */
        test("SglList", new SglList<Int>(), true);
        test("DblList", new DblList<Int>(), false);
    }
}
```