Raymond McLeod, Evan Warner

CS 340

Prof. Barry

2/21/22

Linked Lists

B) A linked list is a leaner collection of data elements which is made of a collection of nodes each pointing to the next node to have a sequence. The second most used data storage type.

C) The name basically describes itself, it’s just a list of items each connecting in some way to the next item on the list. You read from top to bottom but not the other way around. But that can change depending on the type of list it is. There are many pros and cons of using lists instead of arrays and we are going to go through them. A linked list is able to delete and insert much easier and quicker than any array, as an array can either be really fast at adding things but be slow at indexing or vise versa. The reason for this is because an array is basically memory and by deleting something that space in the array will be empty causing fragmentation and slowing the whole process. With a list as long as you delete the node as well as the data inside the node, this problem doesn’t occur. This can also come in handy with inserting any data into the list. But enough about that lets talk about how they work.

A linked list is a sequence of links which hold data, and each link connects to another link. These links are called nodes. And there are three separate parts

* Node – this holds both the data and the Next call
* Next – this is the link to the next node on the list
* LinkedList – this is the head of the list and is called first

Head

Data

Data

Data

LinkedList

Node

Node

Node

Node

Node

Next

This is an example of a linked list.

As you can see you start with a header and then each node holds both data and the index for the next node in the list. Each node is connected in a linear fashion. There are different types of lists that you can use an I will list them but not go into great detail about them

* Double linked list – each node contains both the link to the next node as well as the previous
* Multiply linked list – each node contains more than one next link meaning that one node is connected to 2 or more nodes
* Circular linked list – at the end of list it links back to the beginning of the list.

D) these are the data structures with a linked list:

* Delete first
* Insert first
* Delete last
* Insert last

E) Example Code:

// linkList.java

// demonstrates linked list

// to run this program: C>java LinkListApp

////////////////////////////////////////////////////////////////

class Link

{

public int iData; // data item (key)

public double dData; // data item

public Link next; // next link in list

// -------------------------------------------------------------

public Link(int id, double dd) // constructor

{

iData = id; // initialize data

dData = dd; // (‘next’ is automatically

} // set to null)

// -------------------------------------------------------------

public void displayLink() // display ourself

{

System.out.print(“{“ + iData + “, “ + dData + “} “); }

} // end class Link

////////////////////////////////////////////////////////////////

class LinkList

{

private Link first; // ref to first link on list

// -------------------------------------------------------------

public LinkList() // constructor

{

first = null; // no items on list yet

}

// -------------------------------------------------------------

public boolean isEmpty() // true if list is empty

{

return (first==null);

}

// -------------------------------------------------------------

// insert at start of list

public void insertFirst(int id, double dd)

{ // make new link

Link newLink = new Link(id, dd);

newLink.next = first; // newLink --> old first

first = newLink; // first --> newLink

}

// -------------------------------------------------------------

public Link deleteFirst() // delete first item

{

// (assumes list not empty)

Link temp = first; // save reference to link

first = first.next; // delete it: first-->old next

return temp; // return deleted link

}

// -------------------------------------------------------------

public void displayList()

{

System.out.print(“List (first-->last): “);

Link current = first; // start at beginning of list

while(current != null) // until end of list,

{

current.displayLink(); // print data

current = current.next; // move to next link

}

System.out.println(“”);

}

// -------------------------------------------------------------

}

// end class LinkList

////////////////////////////////////////////////////////////////

class LinkListApp

{

public static void main(String[] args)

{

LinkList theList = new LinkList(); // make new list

theList.insertFirst(22, 2.99); // insert four items

theList.insertFirst(44, 4.99);

theList.insertFirst(66, 6.99);

theList.insertFirst(88, 8.99);

theList.displayList(); // display list

while( !theList.isEmpty() ) // until it’s empty,

{

Link aLink = theList.deleteFirst(); // delete link

System.out.print(“Deleted “); // display it

aLink.displayLink();

System.out.println(“”);

}

theList.displayList(); // display list

}

// end main()

}

// end class LinkListApp

////////////////////////////////////////////////////////////////