

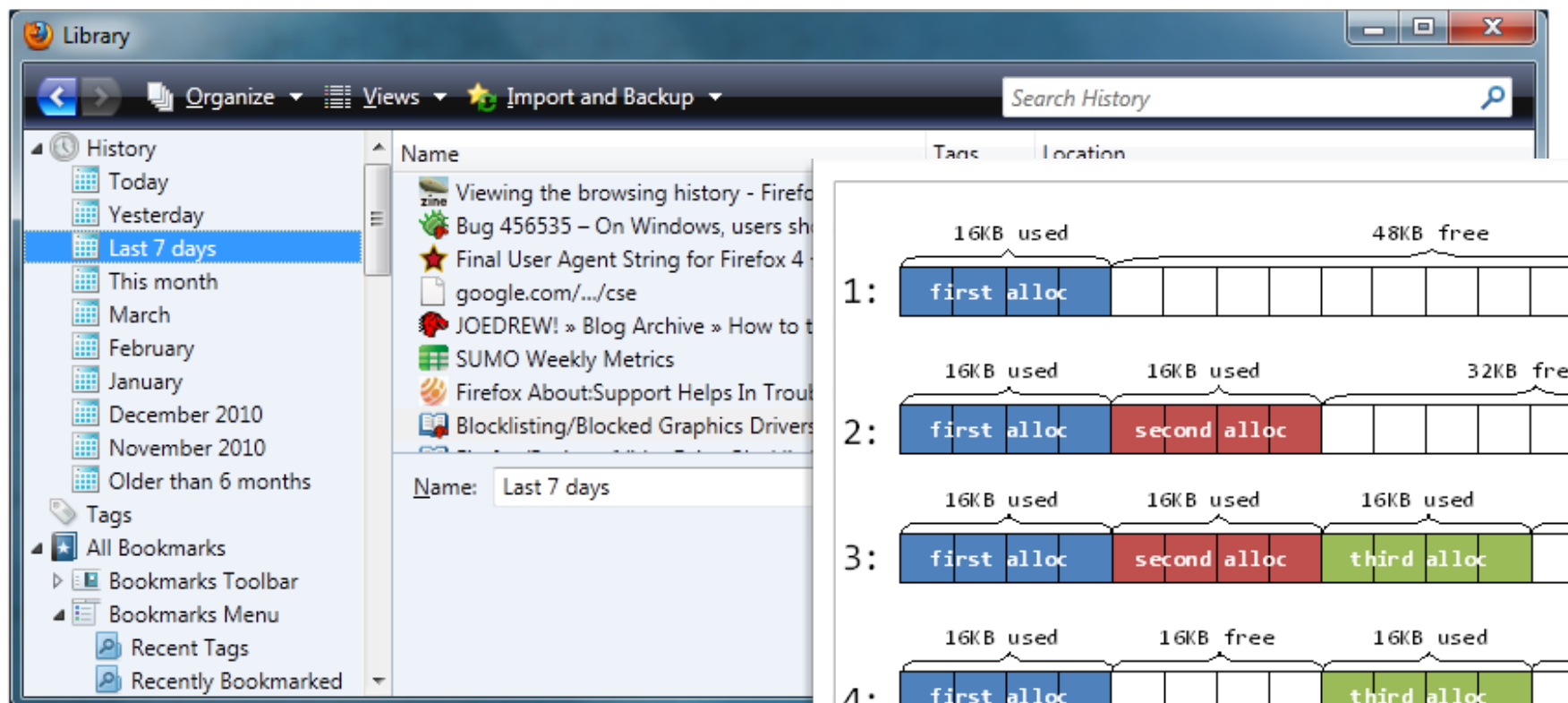


Linked Lists

David Greenstein
Monta Vista High School

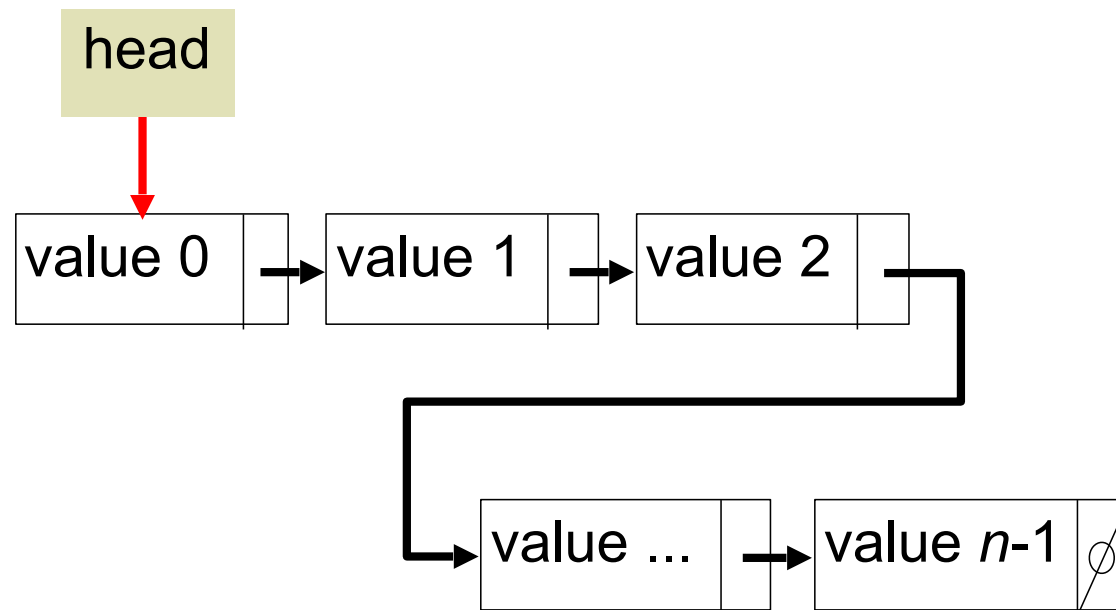
Lists in Computers

- Computers use lists to keep track of many things.
 - browser history
 - memory allocation



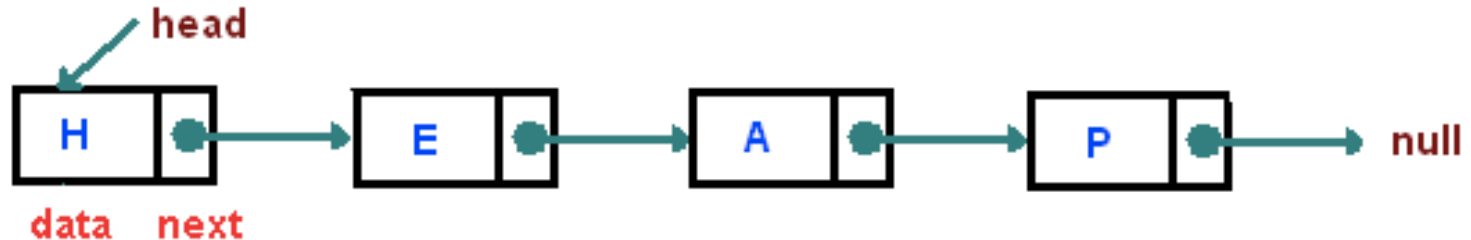
Linked List

- A linear data structure where each element is a separate object and refers to the next element, and sometimes to the preceding element.

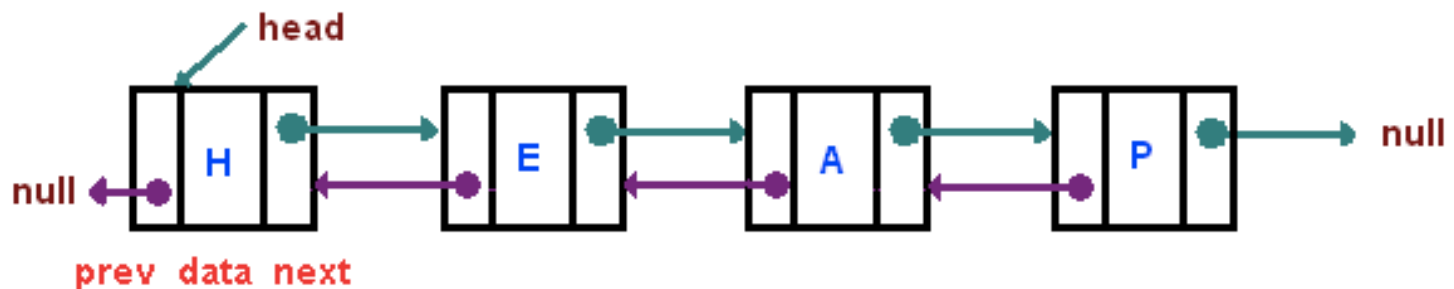


Types of Linked Lists

- **Singly linked list** - A linear data structure where each element is a separate object and refers to the next element.



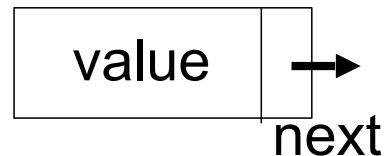
- **Doubly linked list** - A list that has two references, one to the next node and another to previous node.



List Node

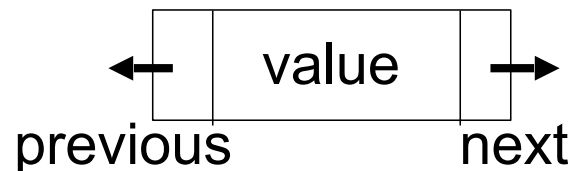
- A “node” is a data element that contains at least two fields:
 - value (the data object)
 - a reference to the next node

Singly-linked Node



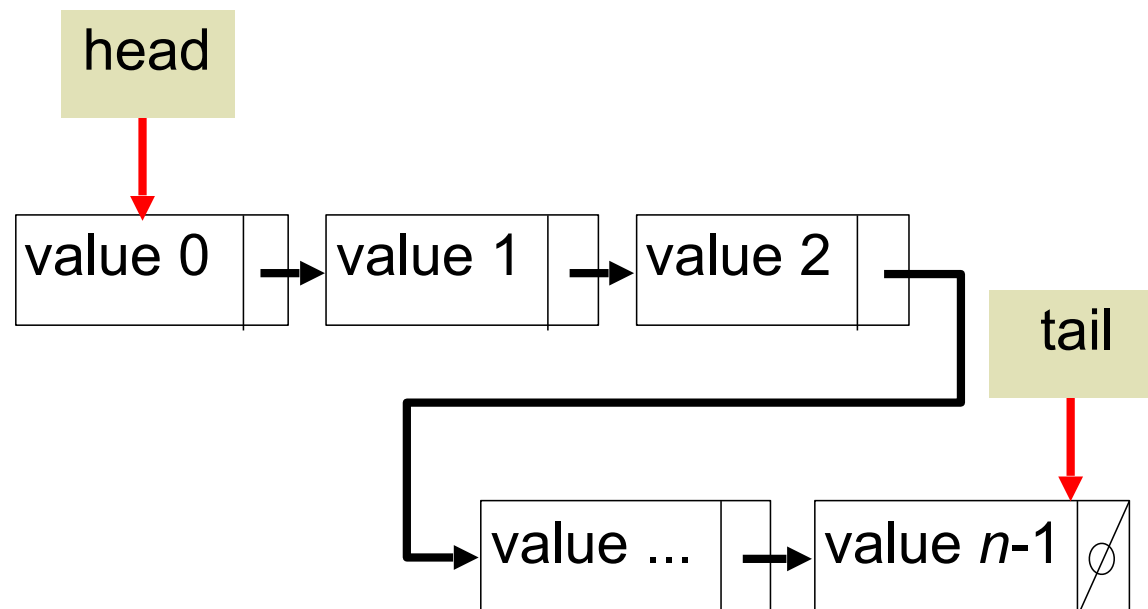
- (optional) a reference to the previous node (doubly-linked)

Doubly-linked Node



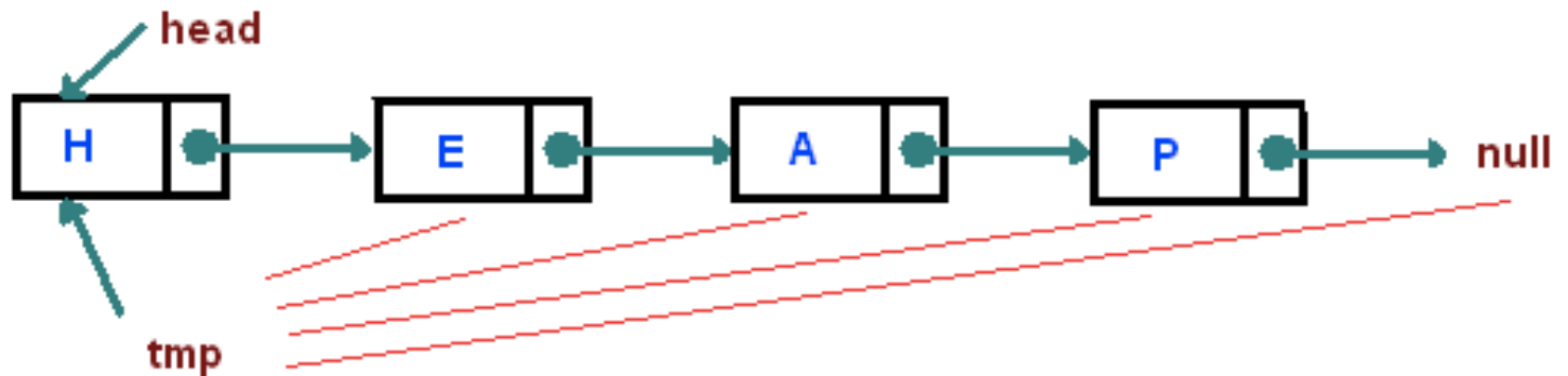
“Special” List Nodes

- The first node is named the **head**, **first**, or **front**.
- The last node is named the **tail**, **last**, or **back**.
- In the last node, the next reference is **null** (the end of the list).
- If the list is empty, both **head** and **tail** point to **null**.



Singly Linked List Operations

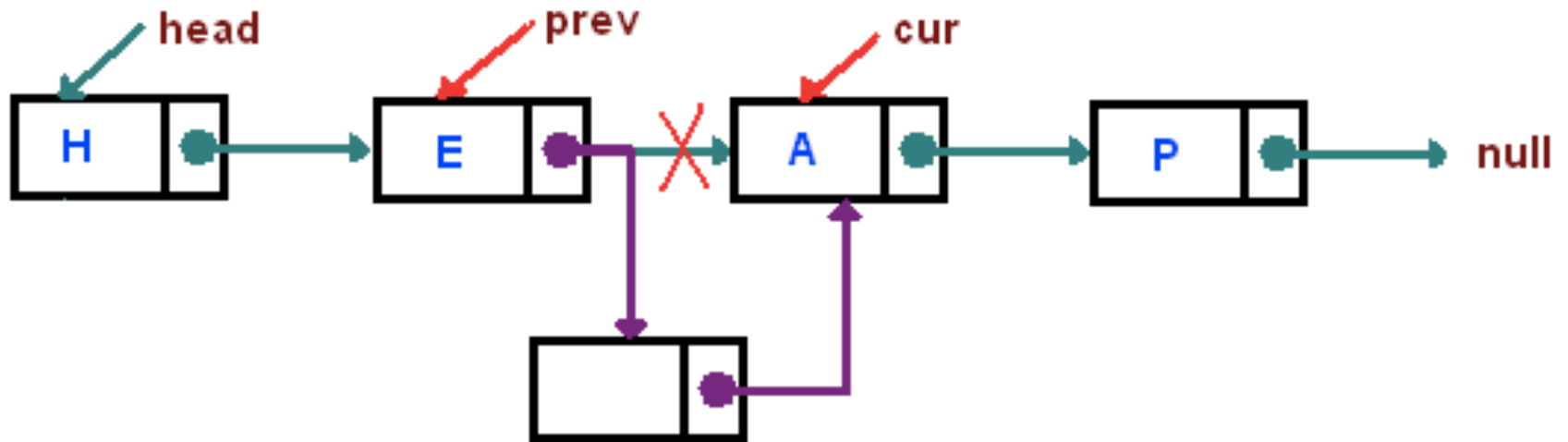
- **traversing** - moving from the head down the list



```
// traverse to end of list
ListNode<String> tmp = head;
while (tmp.getNext() != null)
    tmp = tmp.getNext();
```

Singly Linked List Operations

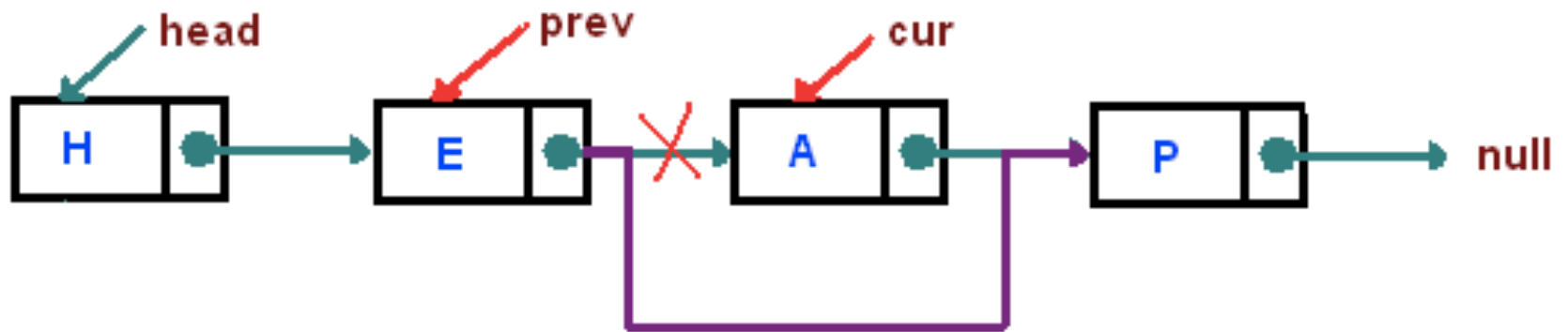
- **inserting** - insert a node inside the list



```
prev.setNext(newNode);  
newNode.setNext(cur);
```


Singly Linked List Operations

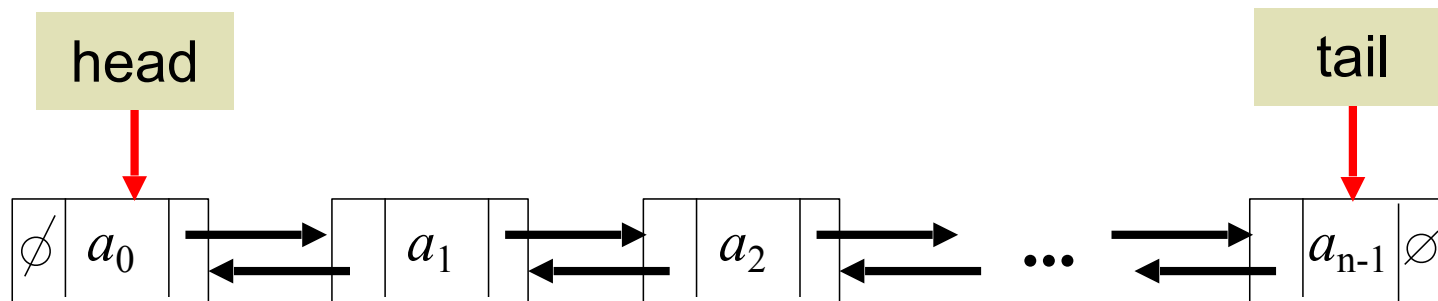
- **remove** - removing a node from the list



```
prev.setNext(cur.getNext());
```

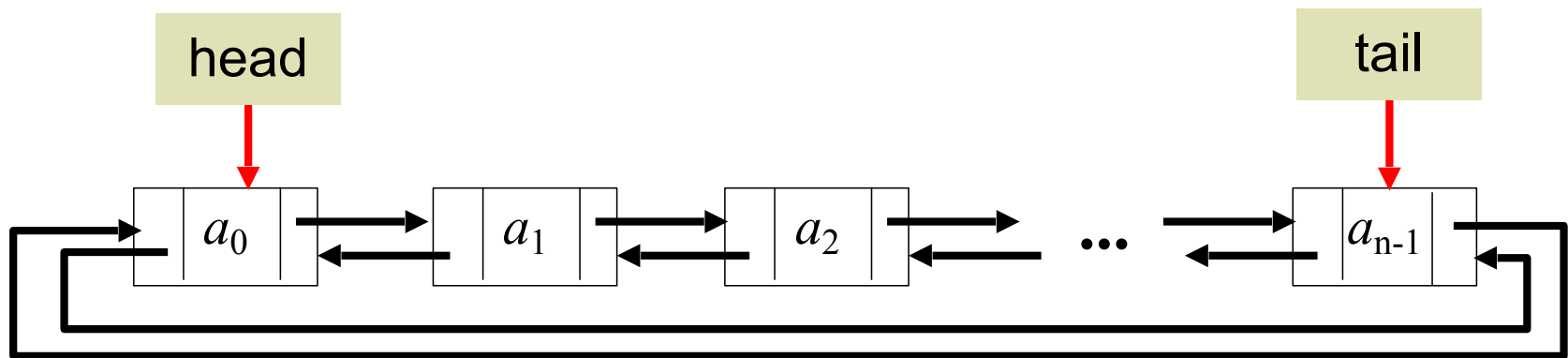
Doubly Linked List

- Each node has references to the **next** and **previous** nodes.
- In the last node, **next** is **null**; in the first node, **previous** is **null**.
- Can be traversed backwards.



Circular List

- **next** in the last node points to the first node
- **previous** in the first node points to the last node
- to tell if you get to the end when traversing, test if the next node equals **head** (or current is **tail**)





Questions?