computer science
illuminated

Linked Lists

Nell Dale & John Lewis

(adaptation by Michael Goldwasser)

Motivation

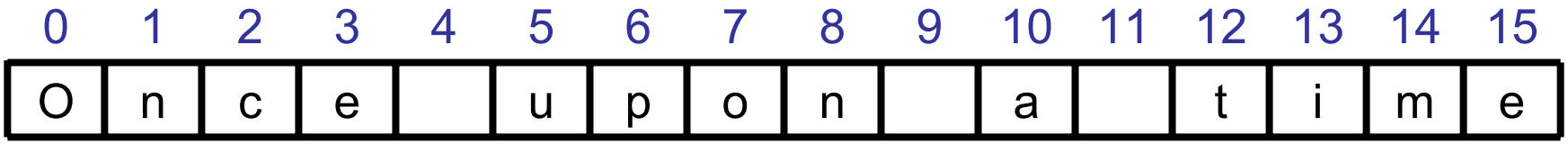
Goal:

design a (simple) word processor.

Must choose data structure to maintain an ordered sequence of characters.

(#chars: 1000? 10000? 100000?)

Use an Array?

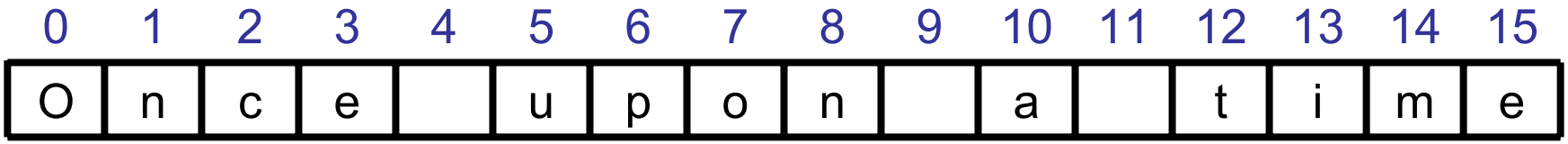


Advantage: Easy to process data in order.

```
i = 0
while (i < Length) do
      print A[i]
      i = i  + 1
```

- Next character is always in very next memory cell

Use an Array?



Disadvantages:

- Since array must use consecutive memory cells, you must declare size of array at outset

- Inserting or Deleting a character can be very costly. (in worst case, proportional to document length)

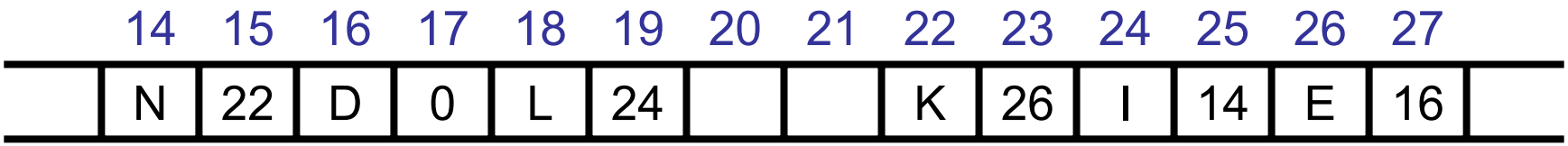
Linked List

- Instead of assuming next character is the very next memory cell ("implicit reference")
- Give **Explicit Reference** (a.k.a. pointer) to location of next character in sequence.

We see explicit references in many areas of life:

- Airline connections (Flight #'s)
- Post Office forwarding
- Links on a Web page

An Example



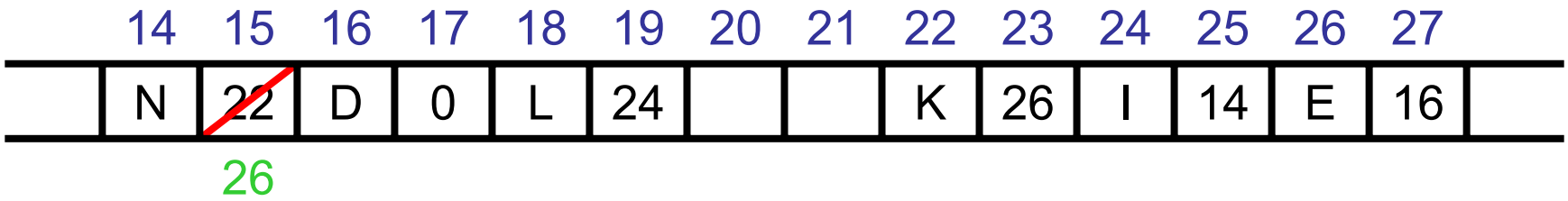
- Must know where to start: Head 18
- Must recognize the end: ("0" is null reference)

So what is character sequence in this example?

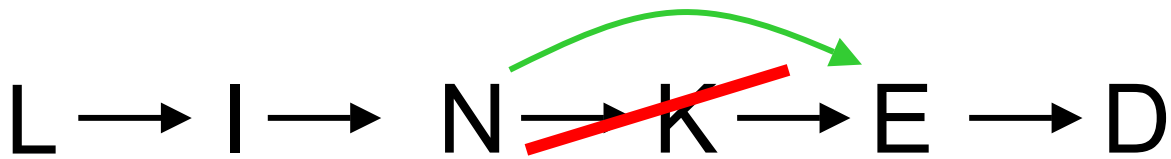
L → I → N → K → E → D

Deleting from a Linked List

Head 18

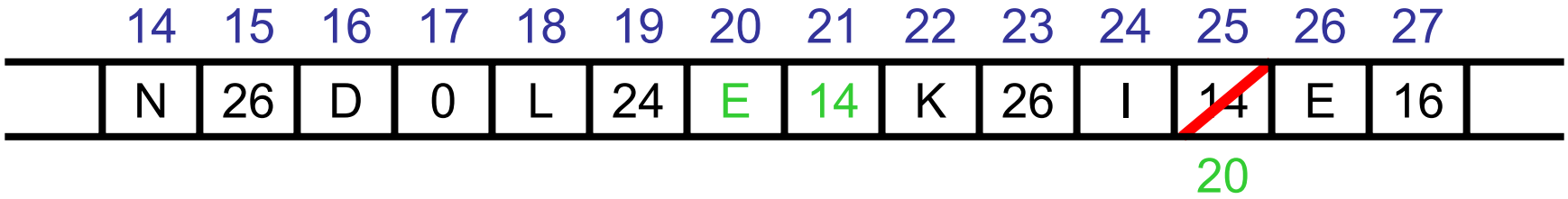


How can I delete “K” from the sequence?

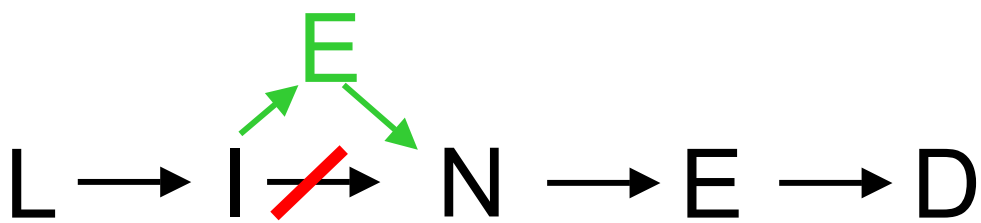


Inserting into a Linked List

Head 18



How can I insert “E” after the “I”?



Linked Implementation

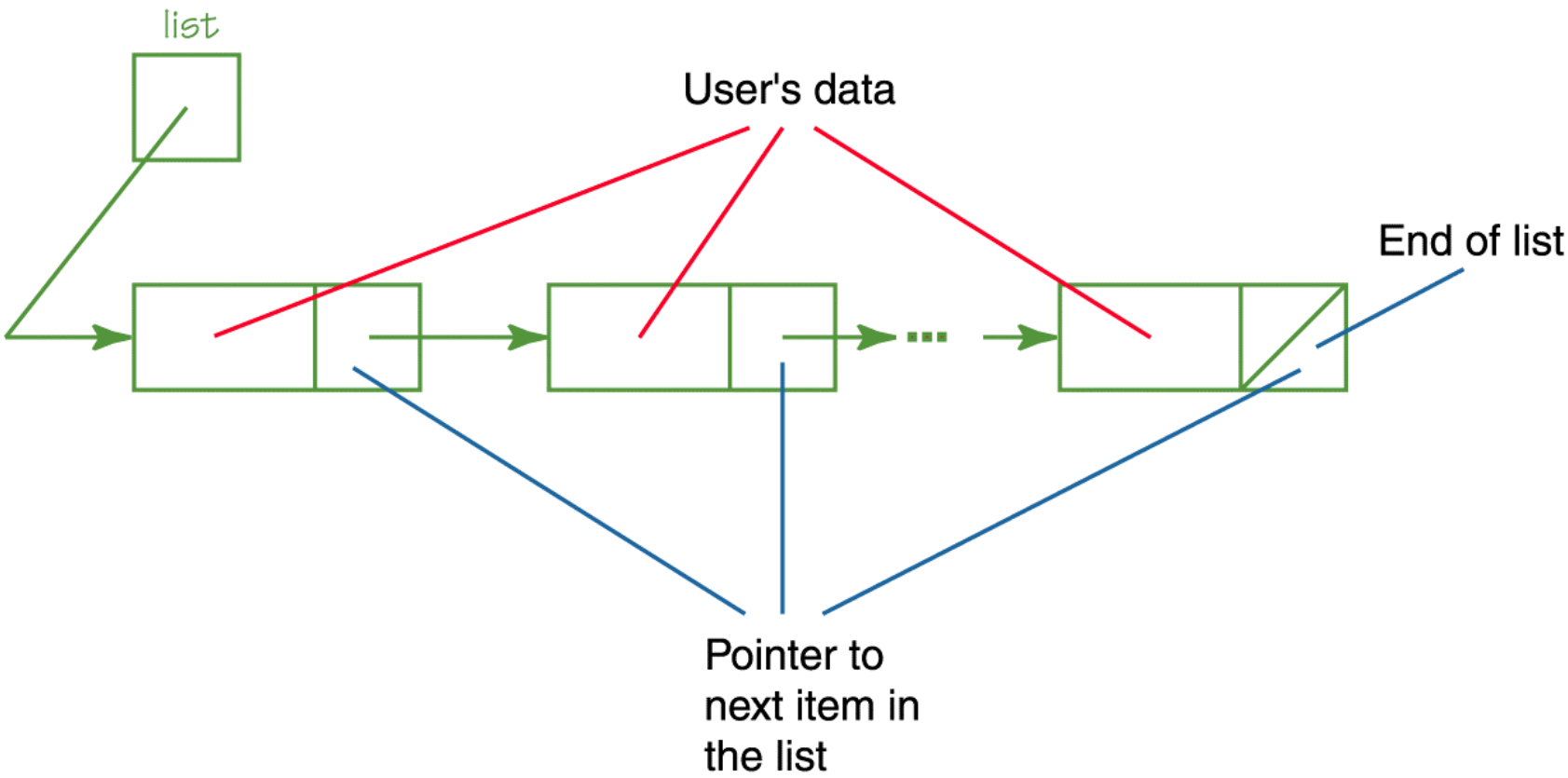


Figure 9.4 Anatomy of a linked list

Linked Implementation

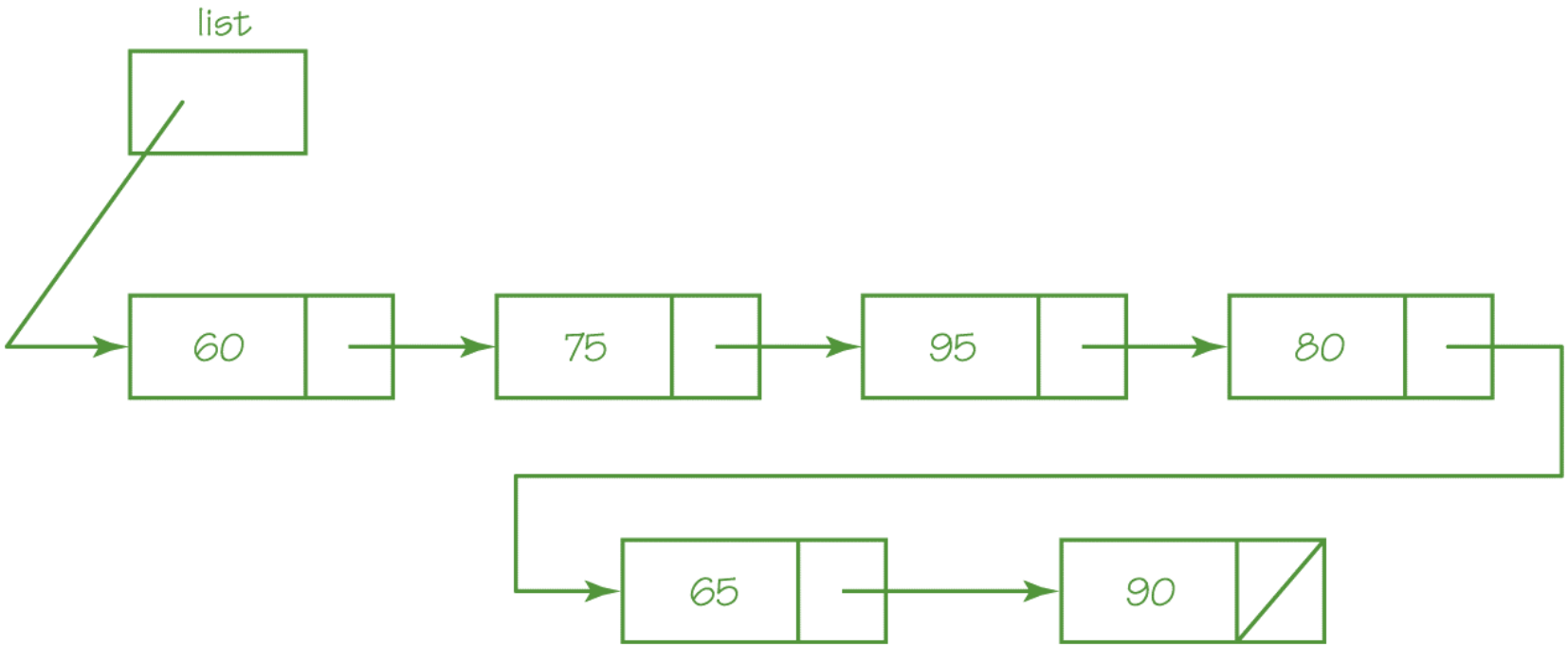


Figure 9.5 An unsorted linked list

Linked Implementation

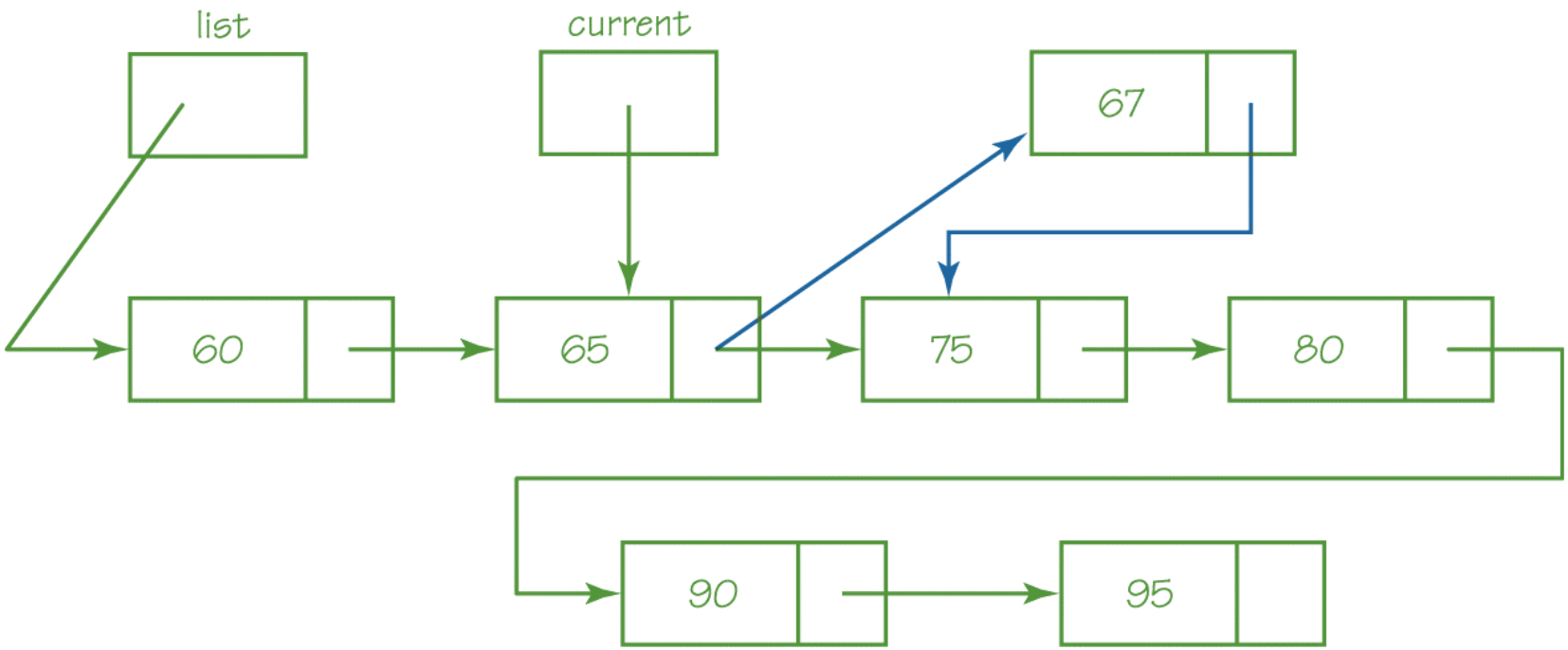


Figure 9.7 **Store a node with *info* of 67 after *current***

Linked Implementation

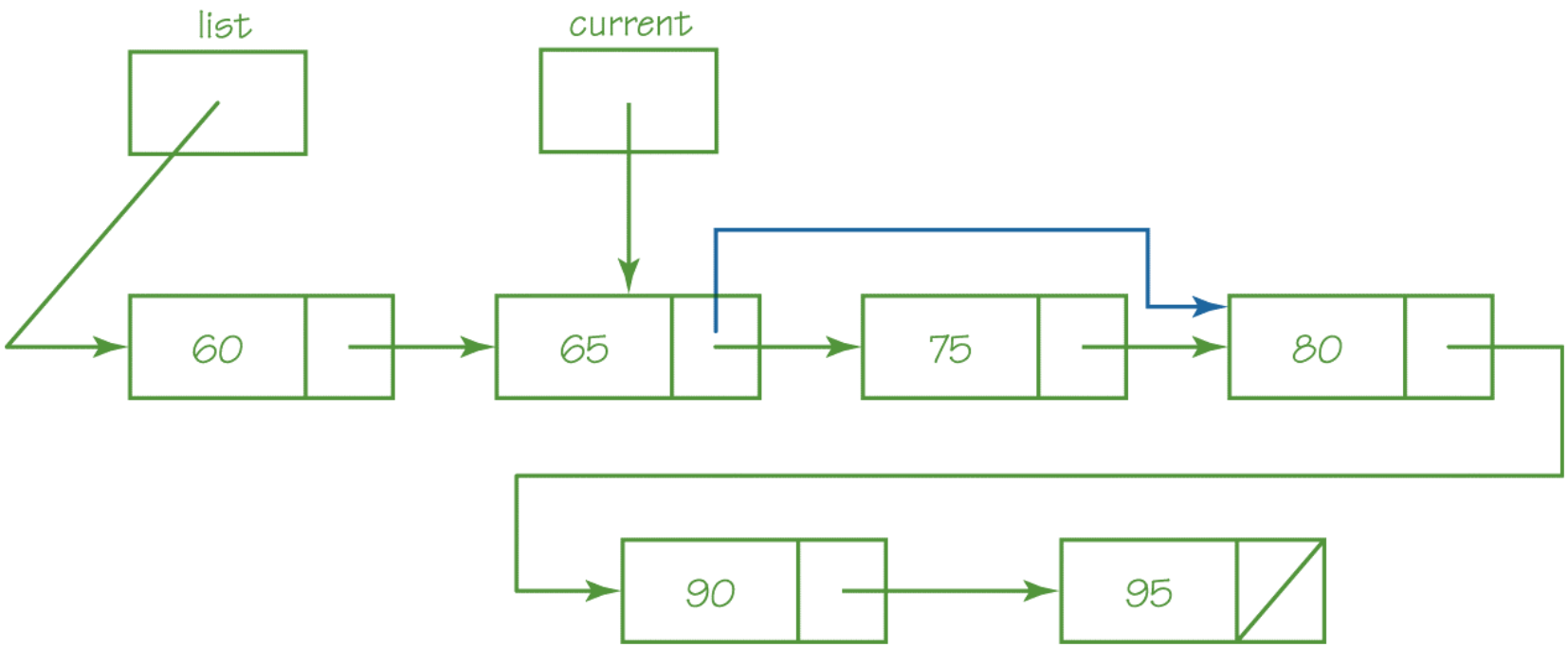


Figure 9.8 **Remove node** *next(current)*

Advantages

- Insertion/Deletion efficiency:
  independent of sequence length
  (not true with Arrays)
- Can Adapt memory usage to sequence length
  (not true with Arrays)
- Can “cut-and-paste” large subsequences

Disadvantages

- Extra Memory usage for Explicit Pointers
- Not Random Access
- Disaster if single link gets corrupted