COMP108
Data Structures and Algorithms

Bubble Sort Algorithm (Part II)

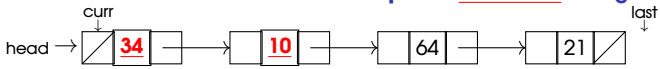
Professor Prudence Wong

pwong@liverpool.ac.uk

2022-23

Bubble sort with linked list - Example

underlined: being considered *italic*: sorted

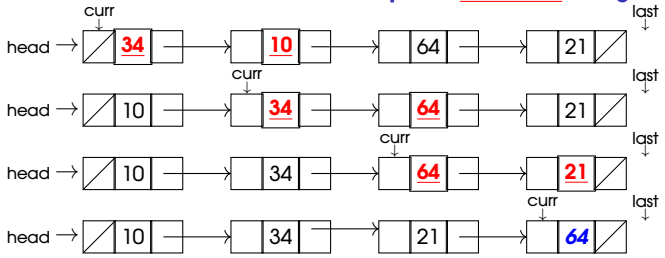


curr

curr.next

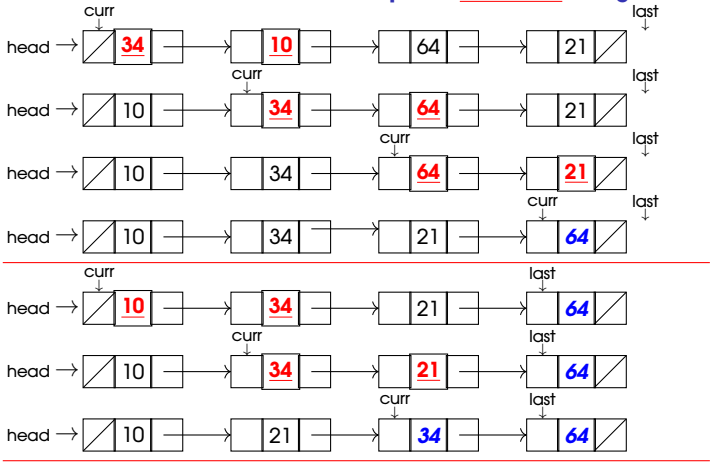
Bubble sort with linked list - Example

underlined: being considered *italic*: sorted



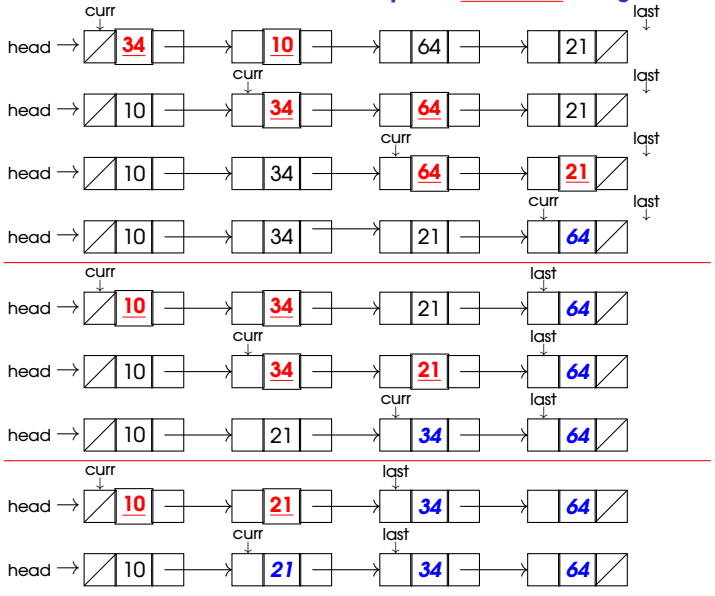
Bubble sort with linked list - Example

underlined: being considered *italic*: sorted



Bubble sort with linked list - Example

underlined: being considered *italic*: sorted



Bubble sort with linked list

- First of all, if head is NIL, then the list is empty & nothing to sort.

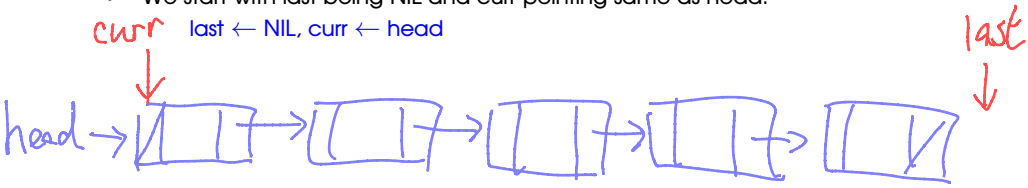
  if head == NIL then Empty list and STOP!

Bubble sort with linked list

- First of all, if head is NIL, then the list is empty & nothing to sort.

  if head == NIL then Empty list and STOP!

- Consider the first round.
  - We start with last being NIL and curr pointing same as head.



Bubble sort with linked list

- First of all, if head is NIL, then the list is empty & nothing to sort.

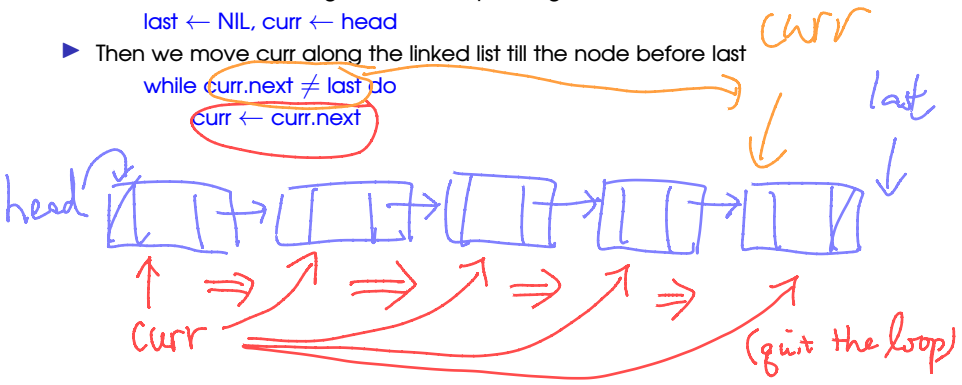
  if head == NIL then Empty list and STOP!

- Consider the first round.
  - We start with last being NIL and curr pointing same as head.

    last ← NIL, curr ← head
  - Then we move curr along the linked list till the node before last

    while curr.next ≠ last do

    curr ← curr.next



Bubble sort with linked list

- First of all, if head is NIL, then the list is empty & nothing to sort.

  if head == NIL then Empty list and STOP!

- Consider the first round.
  - We start with last being NIL and curr pointing same as head.

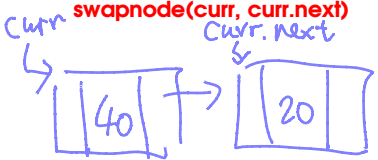
    last ← NIL, curr ← head
  - Then we move curr along the linked list till the node before last

    while curr.next ≠ last do

    curr ← curr.next
  - In the loop, we swap two neighbouring nodes if they are in wrong order

    if curr.data > curr.next.data then

    swapnode(curr, curr.next)



swap curr.data &
curr.next.data

Bubble sort with linked list

- First of all, if head is NIL, then the list is empty & nothing to sort.

  if head == NIL then Empty list and STOP!

- Consider the first round.
  - We start with last being NIL and curr pointing same as head.

    last ← NIL, curr ← head
  - Then we move curr along the linked list till the node before last

    while curr.next ≠ last do

    curr ← curr.next
  - In the loop, we swap two neighbouring nodes if they are in wrong order

    if curr.data > curr.next.data then

    swapnode(curr, curr.next)
- After each round, last should point one node to the left & curr back to head

  last ← curr & curr ← head

Bubble sort with linked list

- First of all, if head is NIL, then the list is empty & nothing to sort.

  if head == NIL then Empty list and STOP!

- Consider the first round.
  - We start with last being NIL and curr pointing same as head.

    last $\leftarrow$ NIL, curr $\leftarrow$ head
  - Then we move curr along the linked list till the node before last

    while curr.next $\neq$ last do

    curr $\leftarrow$ curr.next
  - In the loop, we swap two neighbouring nodes if they are in wrong order

    if curr.data > curr.next.data then

    swapnode(curr, curr.next)

- After each round, last should point one node to the left & curr back to head

  last $\leftarrow$ curr & curr $\leftarrow$ head

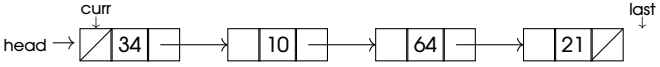
- Then repeat until one node left, i.e., outer loop should be

  while curr.next $\neq$ last do

Bubble sort with linked list - putting things together

if head == NIL then

Empty list and STOP!

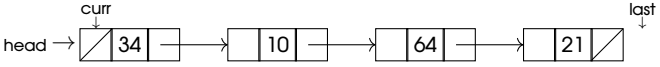


Bubble sort with linked list - putting things together

if head == NIL then

Empty list and STOP!

last ← NIL, curr ← head



Bubble sort with linked list - putting things together

if head == NIL then

Empty list and STOP!

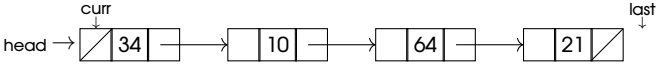
last ← NIL, curr ← head

while curr.next ≠ last do

begin

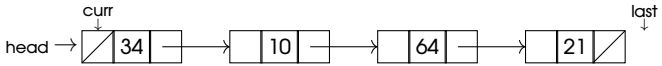
curr ← curr.next

end



Bubble sort with linked list - putting things together

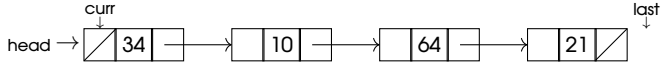
```
if head == NIL then
    Empty list and STOP!
last ← NIL, curr ← head

while curr.next ≠ last do
begin
    if curr.data > curr.next.data then
        swapnode(curr, curr.next)
    curr ← curr.next
end
```



Bubble sort with linked list - putting things together

```
if head == NIL then
    Empty list and STOP!
last ← NIL, curr ← head

while curr.next ≠ last do
begin
    if curr.data > curr.next.data then
        swapnode(curr, curr.next)
    curr ← curr.next
end
```

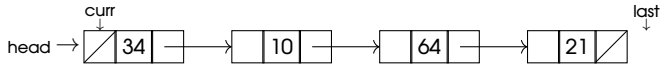
```
swapnode(a, b)
    tmp ← a.data
    a.data ← b.data
    b.data ← tmp
```



Bubble sort with linked list - putting things together

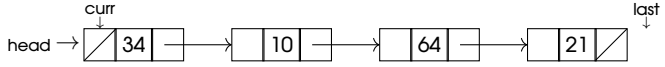
```
if head == NIL then
    Empty list and STOP!
last ← NIL, curr ← head

while curr.next ≠ last do
begin
    if curr.data > curr.next.data then
        swapnode(curr, curr.next)
    curr ← curr.next
end
last ← curr
curr ← head
```

```
swapnode(a, b)
    tmp ← a.data
    a.data ← b.data
    b.data ← tmp
```



Bubble sort with linked list - putting things together

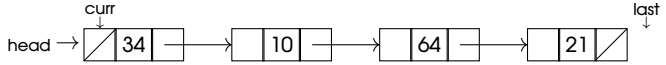
```
if head == NIL then
    Empty list and STOP!
last ← NIL, curr ← head
while curr.next ≠ last do
begin
    while curr.next ≠ last do
    begin
        if curr.data > curr.next.data then
            swapnode(curr, curr.next)
        curr ← curr.next
    end
    last ← curr
    curr ← head
end
```



Bubble sort with linked list - putting things together

```
if head == NIL then
    Empty list and STOP!
last ← NIL, curr ← head
while curr.next ≠ last do
begin
    while curr.next ≠ last do
    begin
        if curr.data > curr.next.data then
            swapnode(curr, curr.next)
        curr ← curr.next
    end
    last ← curr
    curr ← head
end
```

Time complexity?



Summary: Bubble Sort Algorithm with Linked List

Next: Selection Sort and Insertion Sort Algorithms

For note taking