Lect - 17

Searching A Linked List

- when the list is Unsorted
- when the list is sorted.

List is Unsorted :- This algorithm finds the location LOC of the node where item first appears in LIST. or it sets LOC = NULL if search is unsuccessful

1. Set ptr = Start
2. Repeat step 3 while ptr ≠ NULL
3. If item = INFO[ptr] then
   Set LOC = ptr & Exit || item found & we return the location of the node.

   Else
   Set Ptr = LINK[ptr] || update the pointer to next node

   End of if
   End of loop.
4. Set LOC = NULL || Search is unsuccessful.
5. Exit.

List is Sorted :- In this algorithm we find the location LOC of the node where item first appears in LIST or sets LOC = NULL

1. Set Ptr = Start
2. Repeat Step 3 while ptr ≠ NULL
3. if item < INFO[ptr] then
   Set ptr = LINK[ptr]
   Else if item = INFO[ptr] then
   Set LOC = ptr & Exit // search is successful.
   Else
   Set LOC = NULL & Exit // item now exceeds INFO[Ptr]
   End of if
   End of loop at 2
4. Set LOC = NULL
5. Exit.

Sorting a given list :-

This algorithm sort the given linked list. [i.e info part]

1. ptr = start // initialize a pointer ptr
2. Repeat step 3 to 7 while (ptr → link != NULL)
   Begin of outer loop.
3. cptr = ptr → link
4. Repeat 5 to 6 while cptr ≠ NULL
   Begin of inner loop
5. if INFO[ptr] > info[cptr] then
   {
   swap
   }
6. cptr = LINK[cptr] // update the pointer cptr
   End of Inner loop
7. ptr = LINK[ptr] // update the pointer ptr
   End of outer loop.
8. Stop

This algorithm working is same as that of Bubble sort.

Reverse of Link List

① node *temp, *current, *prev = NULL

② Set current = head;

③ while (current != NULL)

{

temp = current → Next;

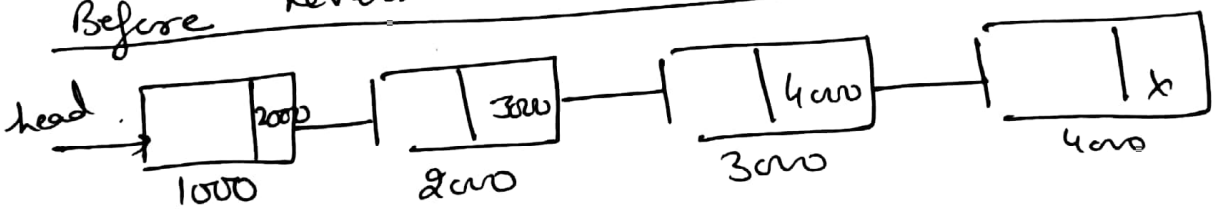
Current → next = prev; // for the first time prev is NULL so current → Next will hold NULL

prev ⟹ current;

Current = temp;

} head = prev;

Before Reverse the list look like



After Reverse