Linked List

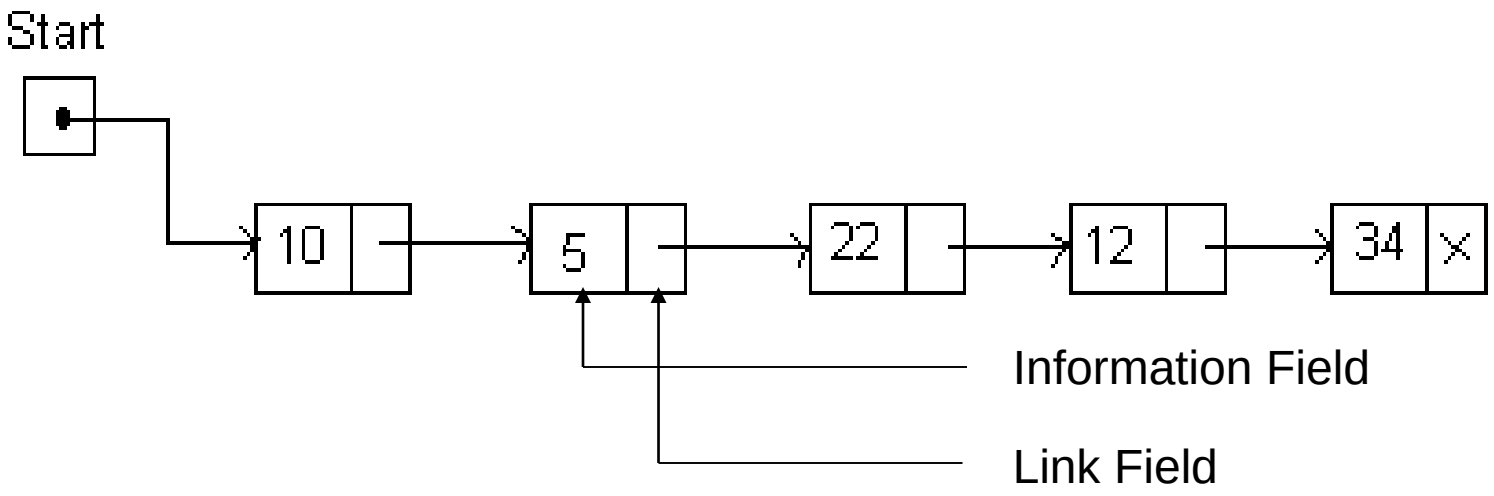
(One-way Linked List)

Linked List

- A linear collection of data elements (linear data structure).
- Each element is represented by a node.
- Each node is divided into two parts.
  1. Information part contains the information of element.
  2. Link part contains the address of the next node in the link.
- Linked list has a list pointer variable, **Head** or **Start**, containing the address of the 1st node in the list.
- If Start contains NULL value, it means the list is empty.
- The address field of last node in the list contains NULL representing invalid address.

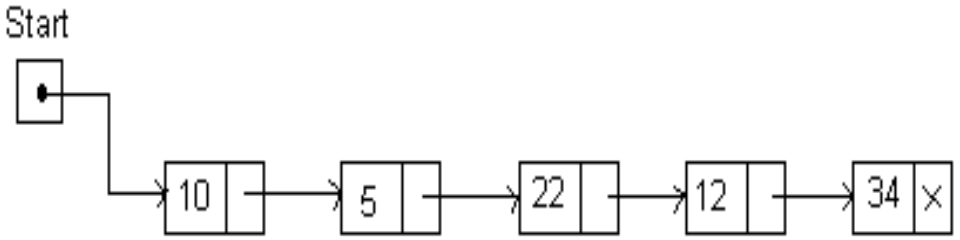
- Example:

The following figure shows a linked list having 5 nodes.



Memory Representation of Linked List

- A linked list can be maintained in memory using two linear arrays.
    1. Info
    2. Link
- Info[K] represents the information part of a node in the list.
- Link[K] represents the nextpointer field of a node in the list.
- The beginning of the list is denoted by the variable S



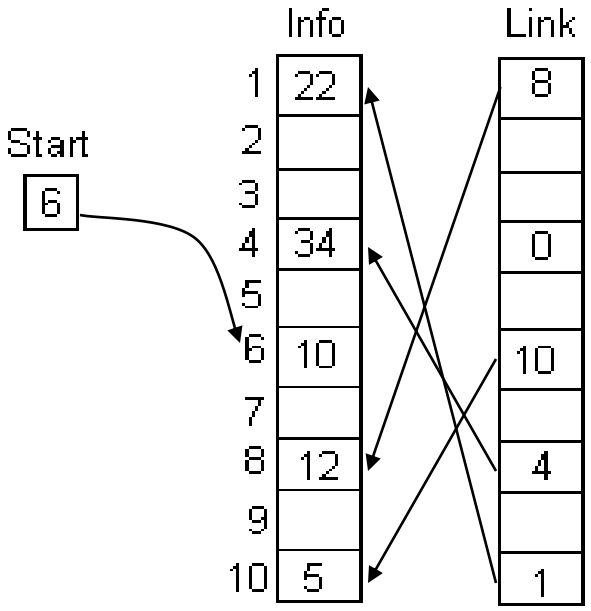


Figure: Linked list & its memory representation

Linked Lists versus arrays

Linked lists have several advantages over arrays.

• Elements can be inserted into linked lists indefinitely, while an array will eventually either fill up or need to be resized, an expensive operation that may not even be possible if memory is fragmented.

• An array from which many elements are removed may become wastefully empty or need to be made smaller.

• Further memory savings can be achieved, in certain cases, by sharing the same "tail" of elements among two or more lists — that is, the lists end in the same sequence of elements. In this way, one can add new elements to the front of the list while keeping a reference to both the new and the old versions — a simple example of a persistent data structure.

Linked lists have several disadvantages over arrays.

- Arrays allow random access, while linked lists allow only sequential access to the elements.
- Singly-linked lists can only be traversed in one direction. This makes linked lists unsuitable for applications where it is useful to look up an element by its index quickly, such as heapsort.
- Another disadvantage of linked lists is the extra storage needed for references.

Creating A Linked List

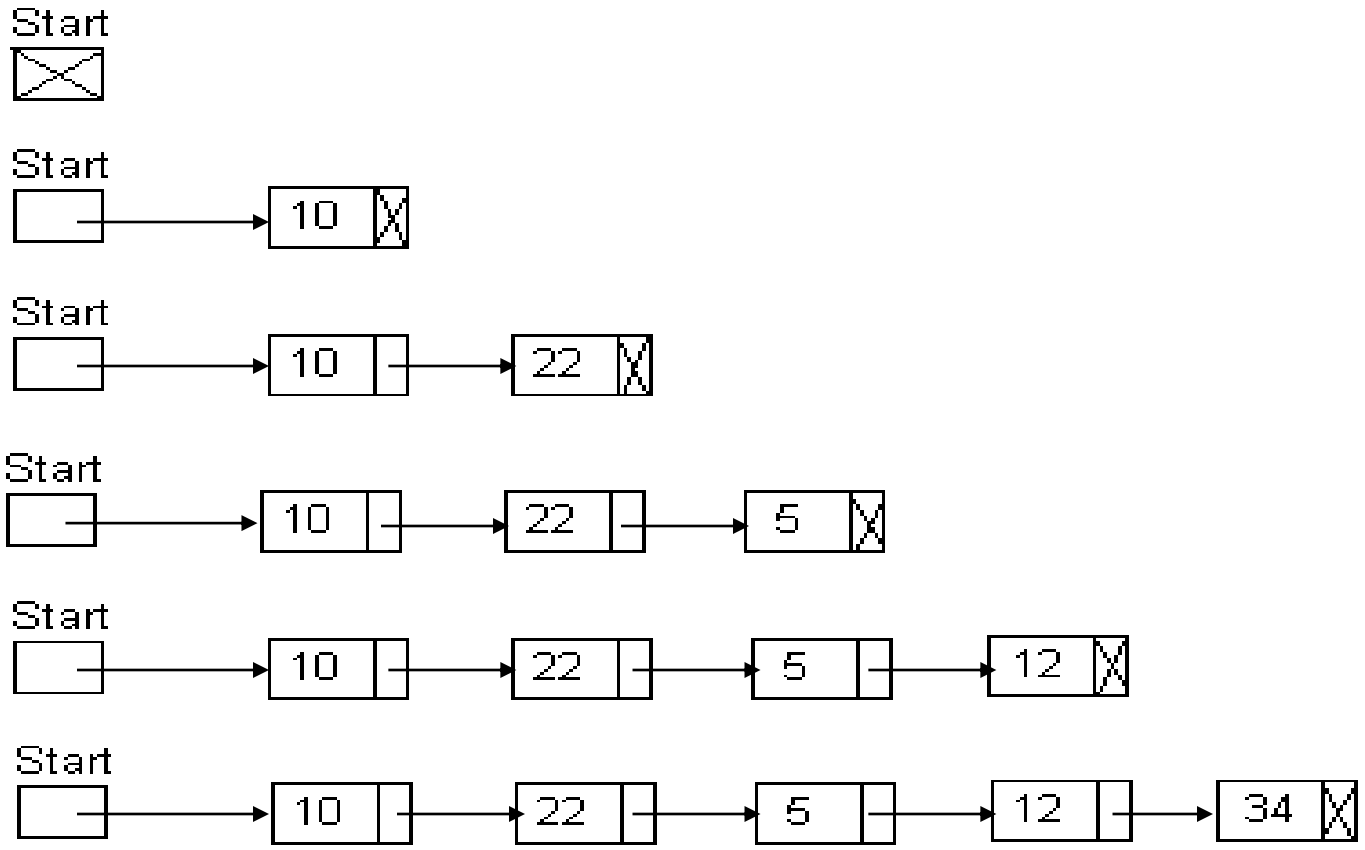
Here we will describe how can a linked list be implemented.

Create_List (Info, Link, Item, Start)

1. Set Info[New] := Item and Link[New] := NULL /…New represents a new node
2. If Start = NULL /…List is empty.
3. then Set Start := New and goto step 9.
4. Set Save:=Start and PTR := Link[Start] /…List is not empty
5. While PTR ≠ NULL repeat steps 6 and 7
6. Set Save :=PTR
7. Set PTR := Link[PTR].
8. Set Link[Save]:=New.
9. go to step 1 /…For another new item

Example: Creating a Linked List

Items: 10, 22, 5, 12, 34



Traversing A Linked List

Traverse_List (List, Info, Link, Start)

1. Set PTR := Start
2. While PTR ≠ NULL repeat steps 3 and 4
3. Apply process to Info[PTR]
4. PTR := Link[PTR]
5. Return.

Counting The No. of Elements in a Linked List

Count_Node_List (List, Info, Link, Start)

1. Set PTR := Start and Num := 0
2. While PTR ≠ NULL repeat steps 3 and 4
3. Num := Num + 1
4. PTR := Link[PTR]
5. Return.

Searching a Linked List

1. List is unsorted
2. List is sorted

Searching an Item from an Unsorted List

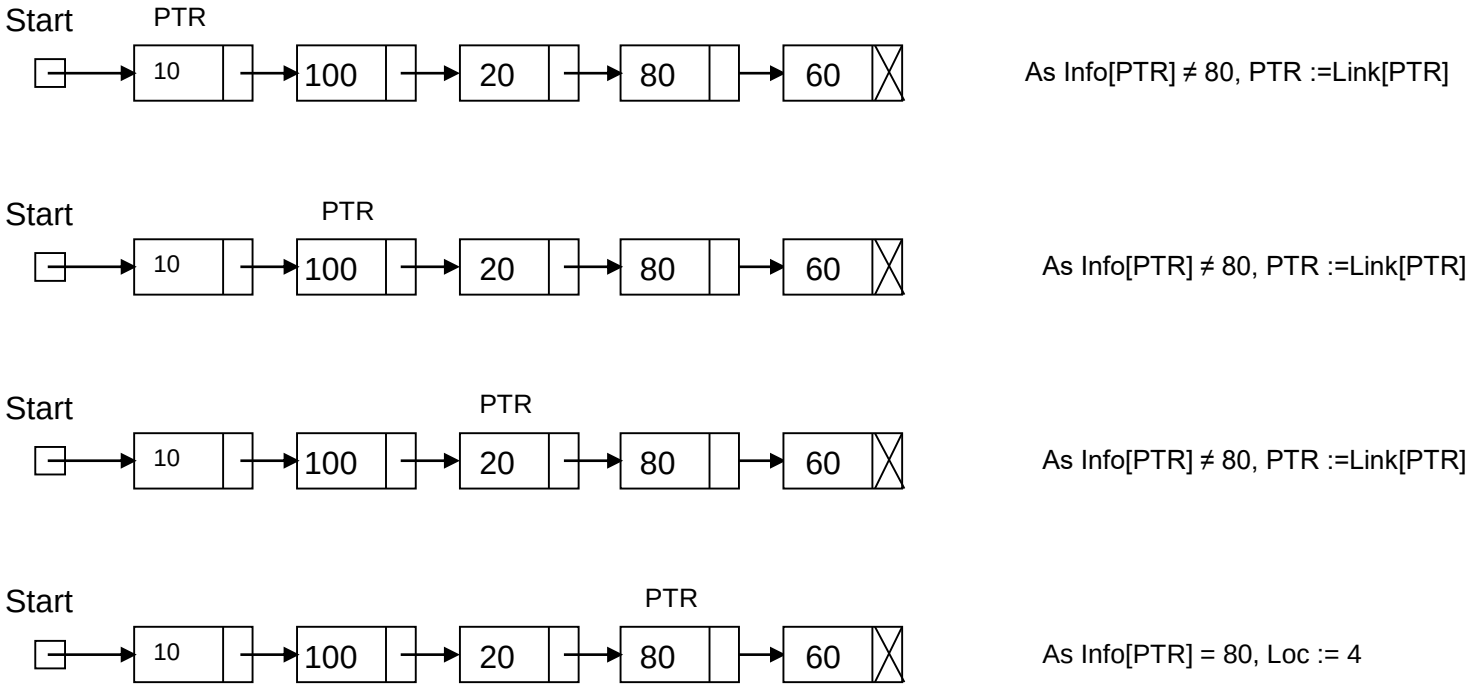
Search_UnsortedList (Info, Link, Start, Item, Loc)

1. Set PTR := Start and Loc := NULL
2. While PTR ≠ NULL do steps 3 to 5
3. If Item = Info[PTR] then
4. Loc := PTR , print: item found and return. /…Successful Search
5. PTR := Link[PTR]
6. If Loc = NULL then Print: 'Item not in the list.' /…Unsuccessful Search
7. Return

Example:

Suppose item 10 is in memory location 1.
item 100 is in memory location 2.
item 20 is in memory location 3.
item 80 is in memory location 4.
item 60 is in memory location 5.

Item to be searched is 80.



Searching an Item from a Sorted List

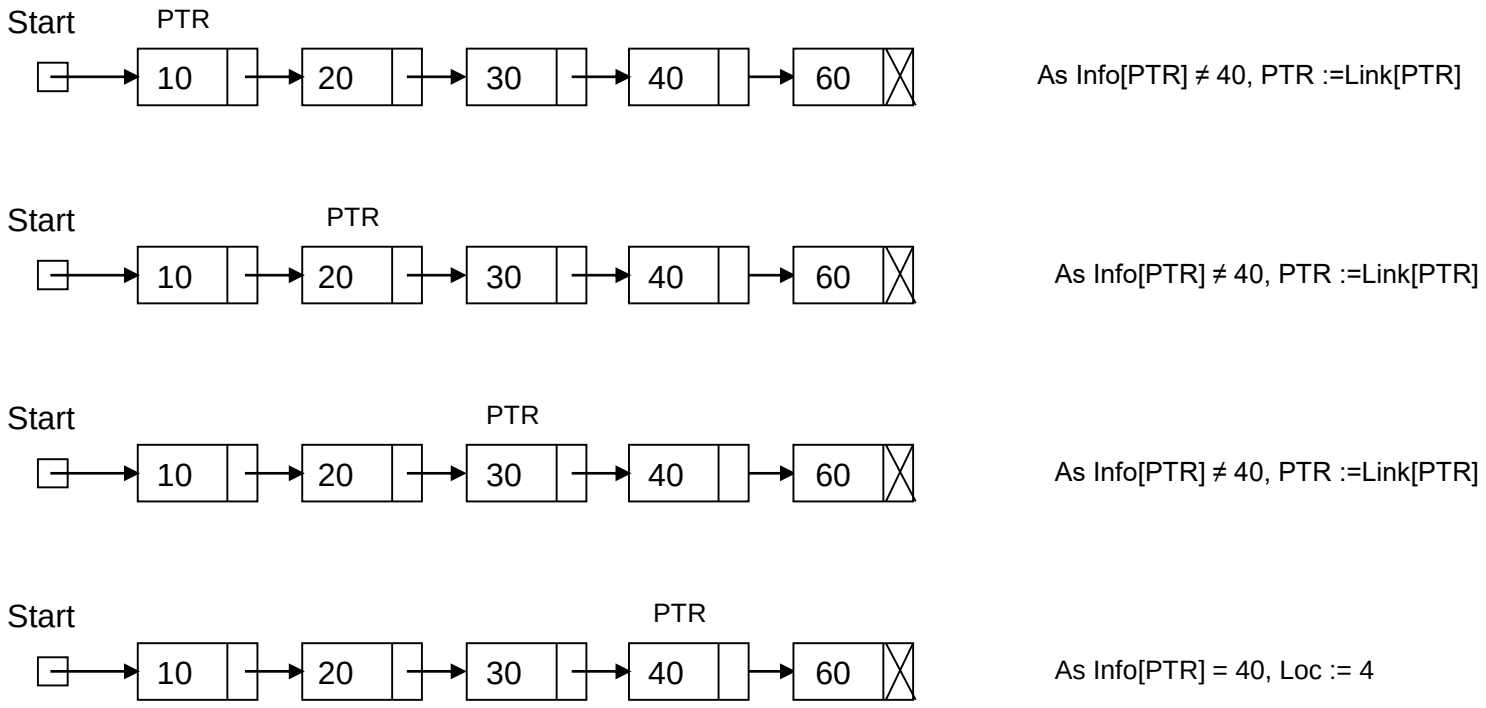
Sch_SortedList (Info, Link, Start, Item, Loc) / List is sorted in ascending order

1. Set PTR := Start and Loc := NULL
2. While PTR ≠ NULL do steps 3 to 7
3. if Item = Info[PTR]
4. then Loc := PTR, print: Item found and return. /…Successful Search
5. else if Item > Info[PTR]
6. then PTR := Link[PTR]
7. else Exit.
8. If Loc = NULL then Print: 'Item not in the list.' /…Unsuccessful Search
9. Return

Example:

Suppose item 10 is in memory location 1.
item 20 is in memory location 2.
item 30 is in memory location 3.
item 40 is in memory location 4.
item 60 is in memory location 5.

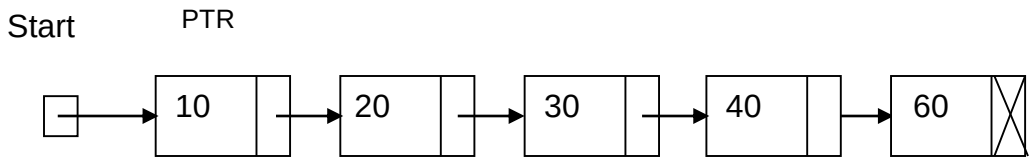
Item to be searched is 40.



Example:

Suppose item 10 is in memory location 1.
item 20 is in memory location 2.
item 30 is in memory location 3.
item 40 is in memory location 4.
item 60 is in memory location 5.

Item to be searched is 6.



As item 6 < Info[PTR] and the given list is already sorted in ascending order, so item can not be in the list.

So, Loc = NULL

END