Linked Lists:

In array’s adding now items in between is difficult.

Linked lists

List of pointers: Basically a list is created that contains objects with a linear order. But that order is maintained through pointers

2 Types

Singly linked list: Where an object points to the next object.

Doubly linked list: Object points to the next and the last object.

Linked list where an object has 2 members

Integer key

Next pointer

(9, ptr)-> (10, ptr)

Contains address of the next item.

End is reached when the last pointer is NULL, could also be a circular queue where last pointer points to the address of the first item,

**Struct node{**

**Int x;**

**Node \*next;**

**}**

Node 1 Node 2 Node 3

(9, ptr) -> (10, ptr)

node1.next = node 2

node2.next = node 3

node3.next = NULL

Actions in linked lists:

Insert

Delete

Search

List-insert (L,x) //add a node to the head of the list

(9, ptr) -> (10, ptr)

X

//L is the linked list

//X is the node to the instersect

//L.head is the head of the list

**If X.next == L.head**

**L.head is the head of list**

Not necessarily continuous memory.

Complexity of inserting a node

Add to beginning: O(1)

0(1)

Independent of the size of the list it probably isn’t one but it is a constant number independent of n

But when adding items to the end of the list

O(n), unless end of list is stored separately

List search (L,k) //L is list, in code you’d use the head of the list, k is the value to search for

**x = x.head**

**While x.next != NIL and**

**x.key != k**

**x=x.next**

**return x**

**last = x**

//This function returns a pointer to the node where k is found or its going to return NILL if it is not found.

Runtime O(n) worst case

Best case:O(1)