DOUBLY LINKED LISTS

Definition

- A doubly linked list or a two-way linked list is a more complex type of linked list which contains a pointer to the next as well as the previous node in the sequence.
- Therefore, it consists of three parts—data, a pointer to the next node, and a pointer to the previous node as shown in Figure

hown in Figure	
START	
\rightarrow X 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5	X
The structure of a doubly linked list can be given as, struct node	
IZ () ~ () /) / / / / / / / / / / / / / / / /	
truct node *prev;	
nt data;	
truct node *next;	
; ALAULAM, KANYAKUMARI	

- The PREV field of the first node and the NEXT field of the last node will contain NULL.
- The PREV field is used to store the address of the preceding node, which enables us to traverse the list in the backward direction.
- Doubly linked list calls for more space per node and more expensive basic operations.
- However, a doubly linked list provides the ease to manipulate the elements of the list as it maintains pointers to nodes in both the directions (forward and backward).
- The main advantage of using a doubly linked list is that it makes searching twice as efficient.

Memory representation of a doubly linked list

1	DATA	PREV	NEXT
□ →1	Н	-1	3
2	4		
3	E	1	6
4			
5			
6	L	3	7
7	L	6	9
8			
9	0	7	-1

- Variable START is used to store the address of the first node.
- In this example, START = 1, so the first data is stored at address 1, which is H.
- Since this is the first node, it has no previous node and hence stores NULL or -1 in the PREV field.
- We will traverse the list until we reach a position where the NEXT entry contains –1 or NULL. This denotes the end of the linked list.
- When we traverse the DATA and NEXT in this manner, we will finally see that the linked list in the above example stores characters that when put together form the word HELLO.

Inserting a New Node in a Doubly Linked List

Case 1: The new node is inserted at the beginning.

Case 2: The new node is inserted at the end.

Case 3: The new node is inserted after a given node.

Case 4: The new node is inserted before a given node.

Case 1: The new node is inserted at the beginning.

We want to add a new node with data 9 as the first node of the list. Then the following changes will be done in the linked list.

Х	1	\rightarrow	7		3		4	\rightarrow	2)	<		
STAR		memory	for the	new node	and in	itialize	its	DATA part	to 9 a	nd PREV	field to	NULL
Х	9											
	the list		before	the STAR	T node.	Now the	new	node beco	mes the	first	node of	
X	9	\rightarrow	1	↓	7	₹	3		4	₹	2 X]
STAF	T											

Algorithm to insert a new node at the beginning

	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	11 Y 12	
	Step	1:	IF AVAIL = NULL
1.0			Write OVERFLOW
. 4	0		Go to Step 9
74			[END OF IF]
	Step	2:	SET NEW_NODE = AVAIL
-4	Step	3:	SET AVAIL = AVAIL -> NEXT
0	Step	4:	SET NEW_NODE -> DATA = VAL
()	Step	5:	SET NEW_NODE -> PREV = NULL
	Step	6:	SET NEW_NODE -> NEXT = START
\mathbf{z}	Step	7:	SET START -> PREV = NEW_NODE
÷.	Step	8:	SET START = NEW_NODE
	Step	9:	EXIT
		1.1	

- In Step 1, we first check whether memory is available for the new node.
- If the free memory has exhausted, then an OVERFLOW message is printed.
- Otherwise, if free memory cell is available, then we allocate space for the new node.
- Set its DATA part with the given VAL and the NEXT part is initialized with the address of the first node of the list, which is stored in START.
- Now, since the new node is added as the first node of the list, it will now be known as the START node, that

is, the START pointer variable will now hold the address of NEW_NODE.

Case 2: The new node is inserted at the end.

We want to add a new node with data 9 as the last node of the list. Then the following changes will be done in the

linked list.

X 1	_←	7	3		4	_ ←	2 X		
START	memory for	the new	node and	initializ	e its DA	TA nart	to 9 and	d its	
	d to NULL.		noue and	101112	C 103 DF		co 5 and	115	
9	ĸ								
Take a po	ointer varia	able PTR	and make	it point	to the f	irst nod	e of the	e list.	
X 1		7	→ 3		4	\rightarrow	2 X]	
START, PT			10 10 ST		199 - 199 -			a.e.	
	so that it	points t	to the las	t node of	the lis	t. Add t	he new r	node af	ter t
	ted by PTR								
X 1		7 🖌	→ 3	\rightarrow	4		2	\rightarrow	9
START							PTR		- 51
		(V)				$\sim \gamma$			
thm to insert :	new node	at the en	d						
thm to insert a	a new node	at the end	d		_		6		
thm to insert a	14						5		
thm to insert a	14	1: IF A	d AVAIL = N Write OV				B II		
thm to insert a	14	1: IF A	AVAIL = N	ERFLOW				2	
thm to insert a	14	1: IF A	AVAIL = N Write OV	ERFLOW			K) FE	HOH	
thm to insert a	Step	1: IF A	AVAIL = N Write OV Go to St	ERFLOW ep 11			KO TEO	NUN	
thm to insert a	Step Step	1: IF A [END 2: SET	WAIL = N Write OV Go to St OF IF]	ERFLOW ep 11 = AVAIL			K)	- CHNO	
thm to insert a	Step Step Step	1: IF A [END 2: SET 3: SET	AVAIL = N Write OV Go to St OF IF] NEW_NODE	ERFLOW ep 11 = AVAIL AVAIL ->	NEXT		K)	CHNO	
thm to insert a	Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = 1	ERFLOW ep 11 = AVAIL AVAIL -> -> DATA	NEXT = VAL		NO TEOLOGI	CHNO	
thm to insert a	Step Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET 5: SET	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = NEW_NODE	ERFLOW ep 11 = AVAIL AVAIL -> -> DATA -> NEXT	NEXT = VAL		NO TEOLEO	CHNOLO	
thm to insert a	Step Step Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET 5: SET 6: SET	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = NEW_NODE NEW_NODE PTR = ST	ERFLOW ep 11 = AVAIL AVAIL -> -> DATA -> NEXT ART	NEXT = VAL = NULL	XT != NL	2003	CHNOLO	
thm to insert a	Step Step Step Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET 5: SET 6: SET 7: Repe	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = NEW_NODE NEW_NODE PTR = ST eat Step 8	ERFLOW ep 11 = AVAIL AVAIL -> -> DATA -> NEXT ART 3 while P	NEXT = VAL = NULL PTR -> NE	XT != NL	2003	ACHNO/ A	
thm to insert a	Step Step Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET 5: SET 6: SET 7: Repe 8:	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = N NEW_NODE NEW_NODE PTR = ST eat Step 8 SET PTR	ERFLOW ep 11 AVAIL -> -> DATA -> NEXT ART While P = PTR ->	NEXT = VAL = NULL PTR -> NE	XT != NL	2003	CHNOL	
thm to insert a	Step Step Step Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET 5: SET 6: SET 7: Repe 8: [END	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = NEW_NODE PTR = ST eat Step 8 SET PTR OF LOOP]	ERFLOW ep 11 = AVAIL AVAIL -> -> DATA -> NEXT ART 3 while P = PTR ->	NEXT = VAL = NULL PTR -> NE NEXT	XT != NL	2003	CHNOLO	
thm to insert a	Step Step Step Step Step Step Step Step	1: IF A [END 2: SET 3: SET 4: SET 5: SET 6: SET 7: Repe 8: [END 9: SET	AVAIL = N Write OV Go to St OF IF] NEW_NODE AVAIL = N NEW_NODE NEW_NODE PTR = ST eat Step 8 SET PTR	ERFLOW ep 11 = AVAIL AVAIL -> -> DATA -> NEXT ART 3 while P = PTR -> XT = NEW	NEXT = VAL = NULL PTR -> NE NEXT	XT != NL	2003	ACHNOLO	

- In Step 6, we take a pointer variable PTR and initialize it with START.
- In the while loop, we traverse through the linked list to reach the last node.
- Once we reach the last node, in Step 9, we change the NEXT pointer of the last node to store the address of the new node.
- Remember that the NEXT field of the new node contains NULL which signifies the end of the linked list.
- The PREV field of the NEW_NODE will be set so that it points to the node pointed by PTR (now the second last node of the list).

Case 3: The new node is inserted after a given node.

We want to add a new node with value 9 after the node containing 3.

	$X 1 \rightarrow 7 \rightarrow 3 \rightarrow 4 \rightarrow 2 X$
	START
	Allocate memory for the new node and initialize its DATA part to 9.
	9
	Take a pointer variable PTR and make it point to the first node of the list.
	$X 1 \rightarrow 7 \rightarrow 3 \rightarrow 4 \rightarrow 2 X$
	START, PTR
	Move PTR further until the data part of PTR = value after which the
	node has to be inserted.
	$X 1 \rightarrow 7 \rightarrow 3 \rightarrow 4 \rightarrow 2 X$
	START PTR
	Insert the new node between PTR and the node succeeding it.
	START PTR
	9
	$X 1 \rightarrow 7 \rightarrow 3 \rightarrow 9 \rightarrow 4 \rightarrow 2 X$
	START
Algorithm	to insert a new node after a given node
	Step 1: IF AVAIL = NULL
	Write OVERFLOW
	Go to Step 12
	[END OF IF]
	<pre>Step 2: SET NEW_NODE = AVAIL</pre>
	Step 3: SET AVAIL = AVAIL -> NEXT
	<pre>Step 4: SET NEW_NODE -> DATA = VAL</pre>
	Step 5: SET PTR = START
	Step 6: Repeat Step 7 while PTR -> DATA != NUM
	Step 7: SET PTR = PTR -> NEXT
	[END OF LOOP]
	Step 8: SET NEW_NODE -> NEXT = PTR -> NEXT
	<pre>Step 9: SET NEW_NODE -> PREV = PTR Step 10: SET PTR -> NEXT = NEW NODE</pre>
	Step 10: SET PTR -> NEXT -> PREV = NEW NODE
	Step 12: EXIT

- In Step 5, we take a pointer PTR and initialize it with START.
- That is, PTR now points to the first node of the linked list. In the while loop, we traverse through the linked list to reach the node that has its value equal to NUM.
- We need to reach this node because the new node will be inserted after this node.
- Once we reach this node, we change the NEXT and PREV fields in such a way that the new node is inserted after the desired node.

Case 4: The new node is inserted before a given node.

We want to add a new node with value 9 before the node containing 3.

	x	1	*	>	7	+	>	3	4	≻	4	*		2 X		
	STAR	Т			-1		<u>L</u>			5	<u>k k</u>		L	2 3		
	Allo	cate	mem	ory f	or the	e new	node	and i	nitia	lize	its D	ATA par	rt to	9.		
		9														
	Take	a p	oint	er va	riable	e PTR	and m	nake i	t poi	nt to	the .	first I	node o	of the	list.	
	х	1	*	>	7	•	>	3	*	>	4	*		2 X		
	STAR						10	20 HZ								
							1.				erted	ose da	ta is	equal		
	1	. 3	arue					10.3			erteu	—>				
	X	1	-		7	-	<u> </u>	3	-		4	→		2 X		
	STAR Add		new	node	in be	tween	the r	node p	ointe	d by	PTR a	nd the	node	prece	ding it	
	x	1	7	→	7			3		*	4	→		2 X		
	STAR	т				J.	1A	PTR	_							
	JIAN				ŝ											
	20		_	~				k	-1	-						
	х	1	+	-	7		<u> </u>	9	+	*	3	< ^	33	4	←	2 X
	STAR	Т						0	- 74							
				2		- 11		٩. 🖌	 	ŗ.	(A)				28	
Algorithm	to in	sert	a ne	w noc	le bef	ore a	given	node	- 2 6	Ķ.	×44					
C				2	// .		ĺ	9	- Ji	£	\mathbb{R}^{1}	c H_{c}		10		
				χq	Ston	1: I		T1 -	NULL	2	9 11	- //		-0		
					step	1. 1			VERF					- 7	r	
				- (N					Step					- E.	i	
						[EN	ND OF									
						2: SI		1000								
					Contraction of the	3: SI										
					SR 01 108 11			_			= VAL	-				
					P. 0.85	5: SI					DTD	DATA	1 - N	LIM		
					Step				R = P			> DATA	:- N	UN		
			1		step		ND OF				NEAT				1	
			- 7	H	Step	T 20			R. C. Lawrence	NEXT	= PTF	2				
			- 44		Step	9: SI	ET NE	W_NOE	DE -> 1	PREV	= PTF	-> PR	EV			
		(Step	10: 9	SET P	TR ->	PREV	= NE	W_NOD	E				
					Step	11: 9	SET P	TR -	> PREV	V -> N	IEXT =	NEW_I	NODE			
					Step	12: 1	EXIT									

- In Step 1, we first check whether memory is available for the new node.
- In Step 5, we take a pointer variable PTR and initialize it with START.
- That is, PTR now points to the first node of the linked list.
- In the while loop, we traverse through the linked list to reach the node that has its value equal to NUM.

- We need to reach this node because the new node will be inserted before this node.
- Once we reach this node, we change the NEXT and PREV fields in such a way that the new node is inserted before the desired node.

Deleting a Node from a Doubly Linked List

In this section, we will see how a node is deleted from an already existing doubly linked list

Case 1: The first node is deleted.

Case 2: The last node is deleted.

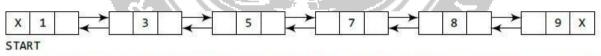
Case 3: The node after a given node is deleted.

Case 4: The node before a given node is deleted.

Case 1: The first node is deleted.

When we want to delete a node from the beginning of the list, then the following changes will be done in the linked

list



Free the memory occupied by the first node of the list and make the second node of the list as the START node.



START

Algorithm to delete the first node ESERVE OPTIMIZE OUTSPREI

Step 1: IF START = NULL Write UNDERFLOW Go to Step 6 [END OF IF] Step 2: SET PTR = START Step 3: SET START = START -> NEXT Step 4: SET START -> PREV = NULL Step 5: FREE PTR Step 6: EXIT

- In Step 1 of the algorithm, we check if the linked list exists or not.
- If START = NULL, then it signifies that there are no nodes in the list and the control is transferred to the last statement of the algorithm.
- However, if there are nodes in the linked list, then we use a temporary pointer variable PTR that is set to point to the first node of the list. For this, we initialize PTR with START that stores the address of the first node of the list.
- In Step 3, START is made to point to the next node in sequence and finally the memory occupied by PTR (initially the first node of the list) is freed and returned to the free pool.

Case 2: The last node is deleted.

We want to delete the last node from the linked list, then the following changes will be done in the linked list.

	445 CONTRACT 1000					- N					- C
	X 1		3		5	\checkmark	7		8		9 X
	START		5. GA		25 HER.			1002 54	AL- 101		
	Take a po	inter var	iable	PTR that	t point	s to the	first	node of	the lis	t.	
	X 1	\rightarrow	3	\rightarrow	5	\rightarrow	7	\rightarrow	8		9 X
	START, PTR	_	Angel - 125						10-00		58
	Move PTR	so that i	t now	points	to the	last node	of th	e list.			
	X 1	\rightarrow	3	\rightarrow	5	\rightarrow	7	\rightarrow	8	\rightarrow	9 X
	START										PTR
	Free the its prece			by the	node po	inted by	PTR an	d store	NULL in	NEXT fi	eld of
	X 1	\rightarrow	3	\rightarrow	5	*	7	\rightarrow	8	x	
	START										
Algorithr	START m to delete t	he last no	de			_			<	2	
Algorithr				IF STA	ART =	NULL			<	-	,
Algorithr						NULL	N		2		
Algorithr			ep 1:	Wr Go	rite U		N		2		,
Algorithr		Ste	2p 1:	Wr Go [END Of	rite U b to S F IF]	NDERFLOW	N		2		
Algorithr		Ste	ep 1: ep 2:	Wr Go [END OF SET P1	rite U to S IF] TR = S	NDERFLOW tep 7 TART					•
Algorithr		Ste Ste	2p 1: 2p 2: 2p 3:	Wr Go END OF SET P1 Repeat	rite U to S IF] IR = S t Step	NDERFLOW tep 7 TART 4 while	e PTR		!= NU		•
Algorithr		Ste Ste	ep 1: ep 2:	Wr Go END OF SET P1 Repeat	rite U to S IF] IR = S t Step	NDERFLOW tep 7 TART	e PTR		!= NU	u	
Algorithr		Ste Ste Ste	2p 1: 2p 2: 2p 3: 2p 4:	Wr Go SET PT Repeat SE END OF	rite U to S IF] IR = S t Step ET PTR LOOP	NDERFLOU tep 7 TART 4 while = PTR -	e PTR > NEXT	[!= NU		
Algorithr		Ste Ste Ste	2p 1: 2p 2: 2p 3: 2p 4:	Wr Go SET PT Repeat SE END OF	rite U to S IF] IR = S t Step ET PTR LOOP	NDERFLOU tep 7 TART 4 while = PTR -	e PTR > NEXT	[!= NU	u	
Algorithr		Ste Ste Ste	2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2: 2	Wr Go SET PT Repeat SE END OF	rite U o to S F IF] TR = S t Step ET PTR F LOOP TR -> P	NDERFLOU tep 7 TART 4 while = PTR -	e PTR > NEXT	[! = NU	u	

- In Step 2, we take a pointer variable PTR and initialize it with START.
- That is, PTR now points to the first node of the linked list. The while loop traverses through the list to reach the last node.
- Once we reach the last node, we can also access the second last node by taking its address from the PREV field of the last node.
- To delete the last node, we simply have to set the next field of second last node to NULL, so that it now becomes the (new) last node of the linked list. The memory of the previous last node is freed and returned

to the free pool.

Case 3: The node after a given node is deleted.

We want to delete the node that succeeds the node which contains data value 4. Then the following changes will be done in the linked list.

$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
START
Take a pointer variable PTR and make it point to the first node of the list.
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
START, PTR
Move PTR further so that its data part is equal to the value after which the node hat to be inserted.
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$
START PTR
Delete the node succeeding PTR.
$X 1 \rightarrow 3 \rightarrow 4 \downarrow 7 \downarrow 8 \rightarrow 9 \chi$
START PTR
$X 1 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 9 X$
START

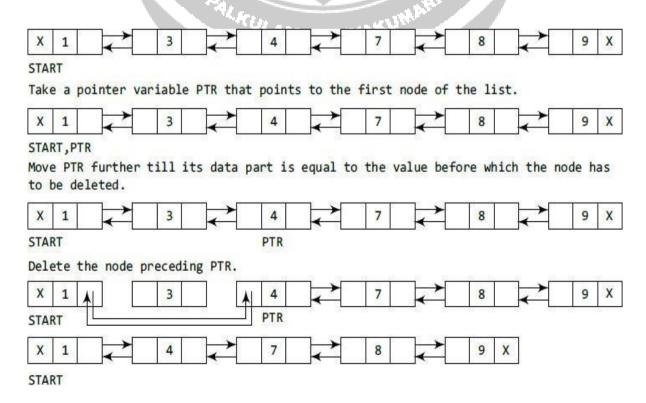
Algorithm to delete a node after a given node

```
Step 1: IF START = NULL
Write UNDERFLOW
Go to Step 9
[END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Step 4 while PTR -> DATA != NUM
Step 4: SET PTR = PTR -> NEXT
[END OF LOOP]
Step 5: SET TEMP = PTR -> NEXT
Step 6: SET PTR -> NEXT = TEMP -> NEXT
Step 7: SET TEMP -> NEXT = TEMP -> NEXT
Step 8: FREE TEMP
Step 9: EXIT
```

In Step 2, we take a pointer variable PTR and initialize it with START. That is, PTR now points to the first node of the doubly linked list. The while loop traverses through the linked list to reach the given node. Once we reach the node containing VAL, the node succeeding it can be easily accessed by using the address stored in its NEXT field. The NEXT field of the given node is set to contain the contents in the NEXT field of the succeeding node. Finally, the memory of the node succeeding the given node is freed and returned to the free pool.

Case 4: The node before a given node is deleted.

Suppose we want to delete the node preceding the node with value 4



Algorithm to delete a node before a given node

```
Step 1: IF START = NULL
Write UNDERFLOW
Go to Step 9
[END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Step 4 while PTR->DATA != NUM
Step 4: SET PTR = PTR->NEXT
[END OF LOOP]
Step 5: SET TEMP = PTR->PREV
Step 6: SET TEMP = PTR->PREV
Step 7: SET PTR->PREV = TEMP->PREV
Step 8: FREE TEMP
Step 9: EXIT
```

- In Step 2, we take a pointer variable PTR and initialize it with START.
- That is, PTR now points to the first node of the linked list.
- The while loop traverses through the linked list to reach the desired node.
- Once we reach the node containing VAL, the PREV field of PTR is set to contain the address of the node preceding the node which comes before PTR.

ULAM, KANYA

OBSERVE OPTIMIZE OUTSPREP

• The memory of the node preceding PTR is freed and returned to the free pool.