**Data Structures**

Implementation of a directed Graph using

Adjacency List

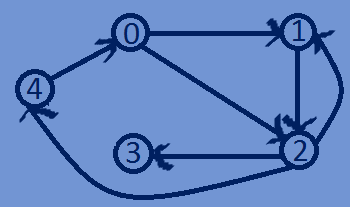
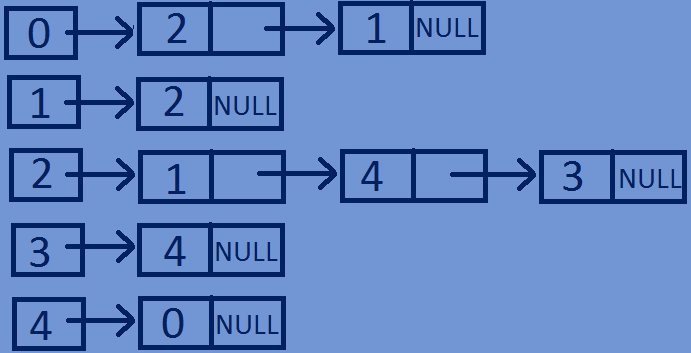
**Adjacency List representation of a Graph:**

* Adjacency list is a collection of unordered lists that are used to represent a graph.
* Here we are going to use Array of linked lists for storing the adjacent nodes of

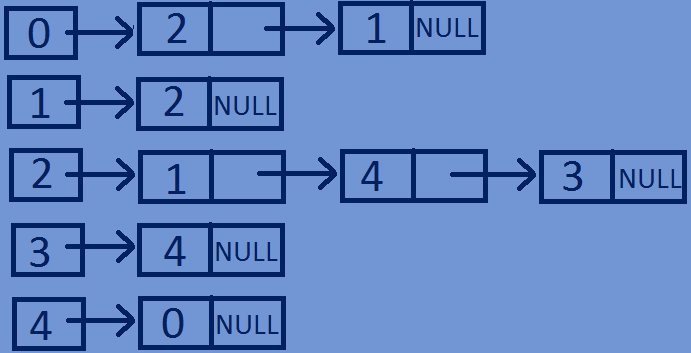
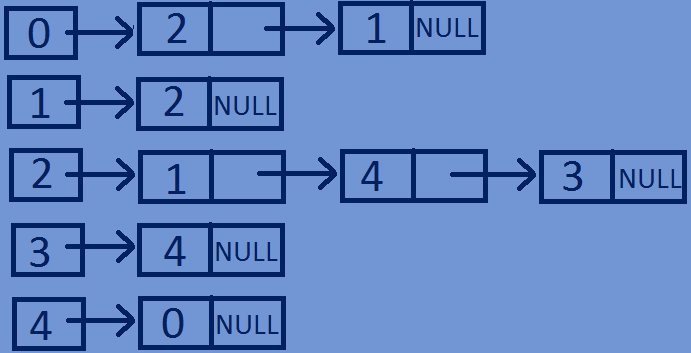
Each node.

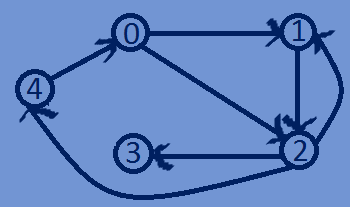
* The size of the array will be equal to the total no of vertices in the undirected graph.
* Here the index of the array represents the vertex of the graph.
* Consider our array is array[],now array[i] represents the linked list containing all the

Vertices adjacent to ith vertex.



**Implementation of directed Graph using Adjacency List:**

#include<stdio.h>  
#include<stdlib.h>  
//Structure for representing a NODE in the Adjacency List  
**typedef struct** Node{  
 **int** dest;  
 **int** weight;  
 **struct** Node \*next;  
}Node;   
//structure for representing an adjacency liat  
**typedef struct** List{  
 Node \*Head;  
}List;  
// A structure to represent a graph - here graph is an array of Adjacency lists  
// size of the array will be equal to the number of vertices in graph  
**typedef struct** Graph{  
 **int** totVertices;  
 List \*array;  
}Graph;  
  
//function To create a new node in the adjacency list  
Node \*createNewNode(**int** dest,**int** weight){  
 Node \*newnode=(Node\*)malloc(**sizeof**(Node));  
 newnode->dest=dest;  
 newnode->weight=weight;  
 newnode->next=NULL;  
 **return** newnode;  
}  
  
//Function To creates a graph of n vertices  
Graph \*createGraph(**int** n){  
 Graph \*graph=(Graph\*)malloc(**sizeof**(Graph));  
 graph->totVertices=n;  
 graph->array=(List\*)malloc(n\***sizeof**(List));  
 //Initialise each adjacency list as empty by making head as NULL  
 **for**(**int** i=0;i<n;i++){  
 graph->array[i].Head=NULL;  
 }  
 **return** graph;  
}  
//function for Adding an edge to a directed graph  
**void** addedge(Graph \*graph,**int** src,**int** dest,**int** weight){  
 Node \*newnode=createNewNode(dest,weight);  
 newnode->next=graph->array[src].Head;  
 graph->array[src].Head=newnode;  
}  
//Function for printing Adjacency list corresponding to each vertex  
**void** printGraph(Graph \*graph){  
 **for**(**int** i=0;i<graph->totVertices;i++){  
 Node \*Headnode=graph->array[i].Head;  
 printf("connected vertices of vertex %d are:head",i);  
 **while**(Headnode){  
 printf("->%d",Headnode->dest);  
 Headnode=Headnode->next;  
 }  
 printf("\n");  
 }  
}  
//main function  
**int** main(){  
 **int** n=5;  
 Graph \*graph=createGraph(n);  
 addedge(graph,0,1,2);  
 addedge(graph,0,2,1);  
 addedge(graph,1,2,3);  
 addedge(graph,2,3,1);   
 addedge(graph,2,4,7);  
 addedge(graph,2,1,1);  
 addedge(graph,3,4,5);  
 addedge(graph,4,0,4);  
 printGraph(graph);  
}

**Output:**

connected vertices of vertex 0 are:head->2->1

connected vertices of vertex 1 are:head->2

connected vertices of vertex 2 are:head->1->4->3

connected vertices of vertex 3 are:head->4

connected vertices of vertex 4 are:head->0