**Data Structures**

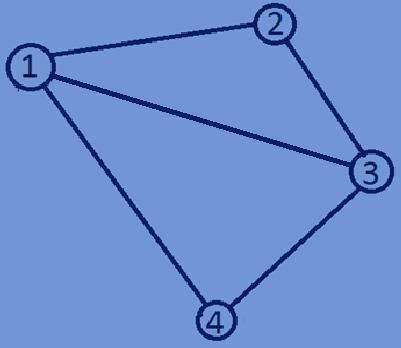
Implementation of an Undirected graph

Adjacency Matrix

**Representation of an Undirected graph using adjacency matrix:**

* In the adjacency matrix representation of a graph ,we represent a graph using

Matrix of size V×V where V is the total no of vertices in a graph.So here we represent a graph using a two dimensional array of size V×V.Let the 2d array be array[][] and array[i][j]=1 indicates that there is a connection(edge) from vertex i to vertex j.For exampleA graph with four vertices can be represented using a 4×4 matrix and if there is An edge between any two vertices then we store value 1 in ith row and jth column.Adjacency matrix of an undirected graph is always symmetric.

 1 2 3 4

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |

1

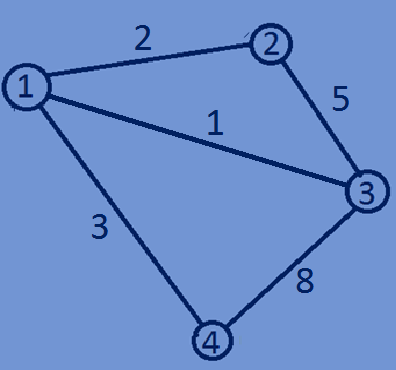
2

3

4

**Representation of a weighted Undirected graph using adjacency matrix:**

* We can also represent weighted graphs using adjacency matrix.
* If there is a weighted edge of weight ‘w’ from vertex ‘i’ to vertex ‘j’,then array[i][j]=’w’



1 2 3 4

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2 | 1 | 3 |
| 2 | 0 | 5 | 0 |
| 1 | 5 | 0 | 8 |
| 3 | 0 | 8 | 0 |

1

2

3

4

**Allocating memory dynamically for a 2d array using single pointer:**

* If there are ‘n’ vertices allocate n\*n memory blocks because here we have to

Construct a adjacency matrix of size n×n.

* In the following example, we are going to represent matrix of size 4×4 using

A integer pointer.

#include<stdio.h>  
#include<stdlib.h>  
**typedef int** array;  
**int** main(){  
 **int** n=5;  
 array \*matrix=(array\*)malloc((n\*n)\***sizeof**(array));  
 **for**(**int** i=0;i<n;i++){  
 **for**(**int** j=0;j<n;j++){  
 \*(matrix+i\*n+j)=i+j;  
 }  
 }  
 **for**(**int** i=0;i<n;i++){  
 **for**(**int** j=0;j<n;j++){  
 printf("%d ",\*(matrix+i\*n+j));  
 }  
 printf("\n");  
 }  
 **return** 0;  
}

**Output:**

0 1 2 3 4

1 2 3 4 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
| 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 6 |
| 3 | 4 | 5 | 6 | 7 |
| 4 | 5 | 6 | 7 | 8 |

2 3 4 5 6 0

3 4 5 6 7 1

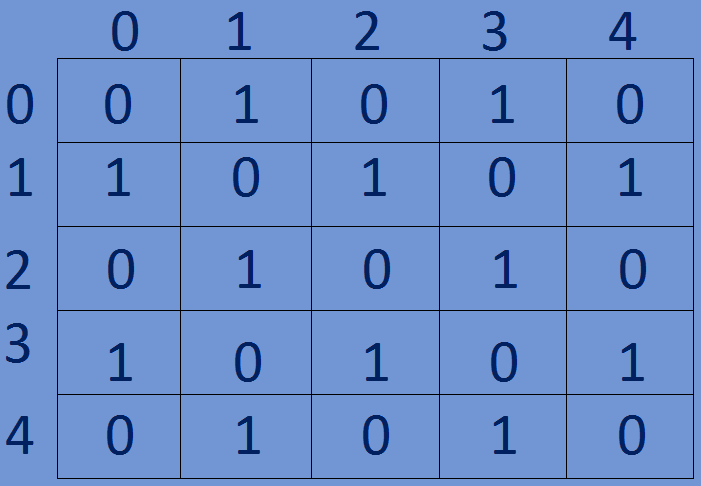
4 5 6 7 8 2

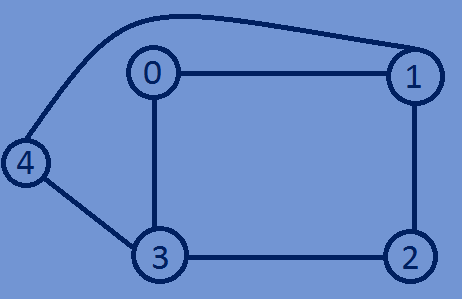
3

4

**Function for implementing undirected graph using Adjacency Matrix:**

#include<stdio.h>  
#include<stdlib.h>  
**typedef int** graph;  
//Constructing a undirectedgraph  
graph \*buildUndirectedGraph (**int** n) {  
 **int** i,j;  
 graph \*array = (graph \*) malloc(n \* n \* **sizeof**(graph));  
 **for** (i = 0; i < n; i++) {  
 **for** (j = 0; j < n; j++) {  
 **if** (i == j) {  
 \*(array + i \* n + j) = 0;  
 } **else if** (i != j) {  
 \*(array + i \* n + j) = (i + j) % 2;  
 \*(array + i + j \* n) = (i + j) % 2;  
 }  
 }  
 }  
 **return** array;  
}  
**int** main(){  
 **int** n=5,i,j;  
 graph \*array;  
 array=buildUndirectedGraph(n);  
}

****

****