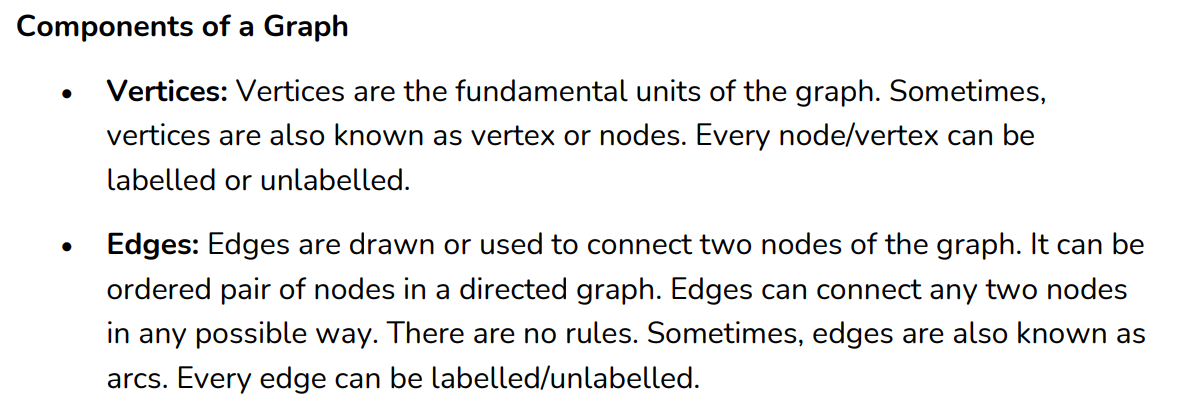
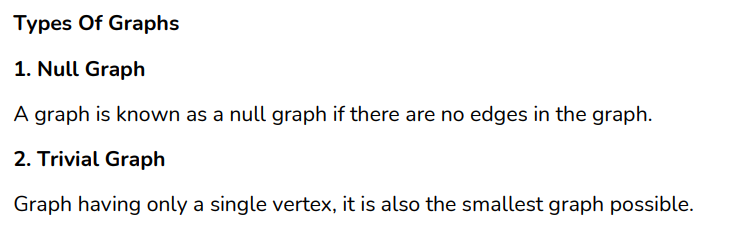
**Practical 11**

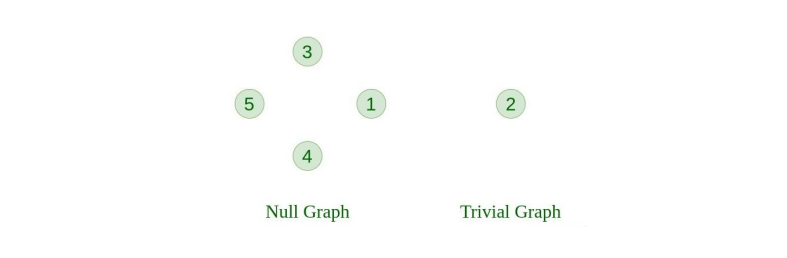
**Aim :** Create Graph storage structure using adjacency list.

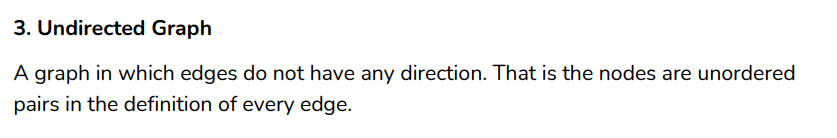
**Theory :**

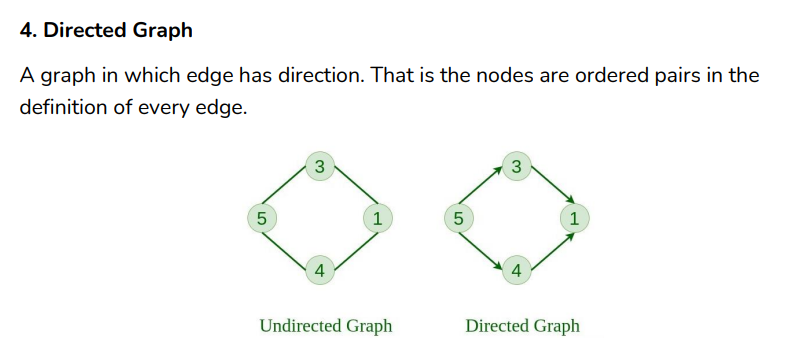
A **Graph i**s a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices (V) and a set of edges (E). The graph is denoted by G (E, V).

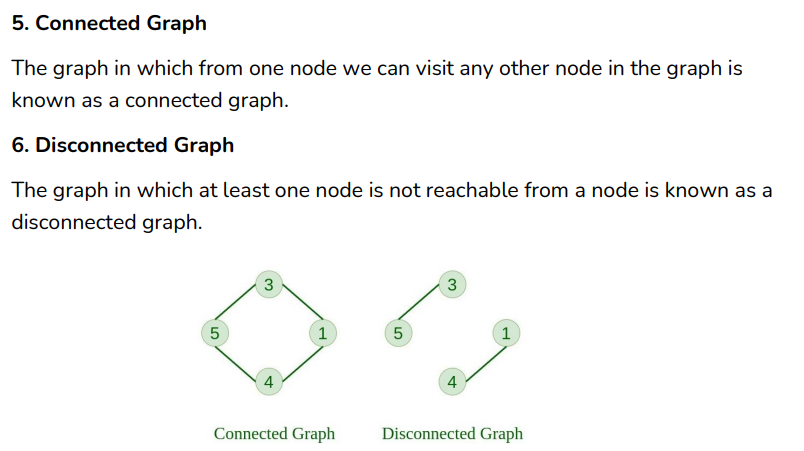


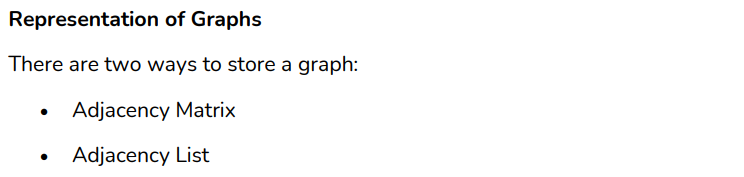


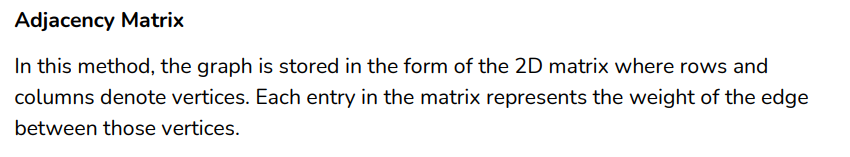


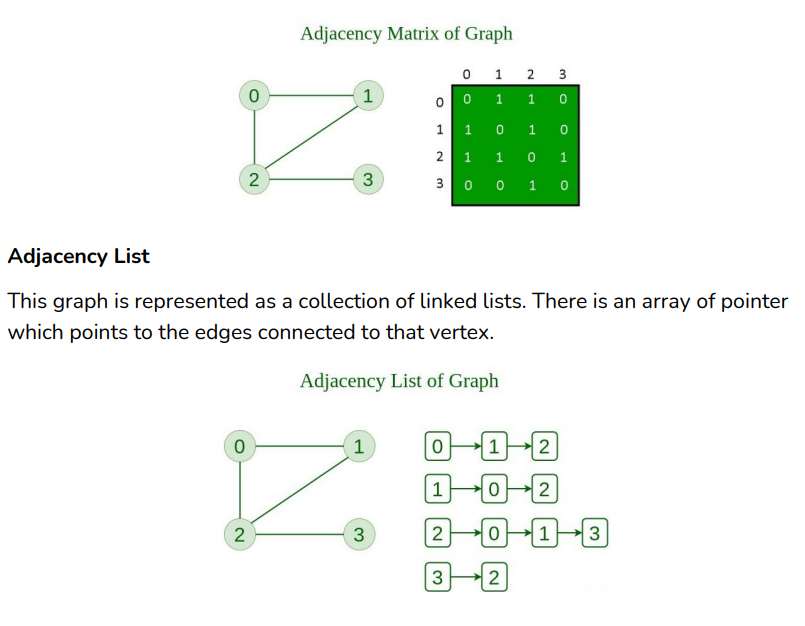












**Code :**

#include<iostream>

#include<list>

#include<iterator>

using namespace std;

void displayAdjList(list<int> adj\_list[], int v)

{

for(int i = 0; i<v; i++)

{

cout << "Vertex "<<i << "--->";

list<int> :: iterator it;

for(it = adj\_list[i].begin(); it != adj\_list[i].end(); ++it) {

cout << \*it << " ";

}

cout << endl;}

}

void add\_edge(list<int> adj\_list[], int u, int v)

{

adj\_list[u].push\_back(v);

adj\_list[v].push\_back(u);

}

int main()

{

int v = 6;

list<int> adj\_list[v];

add\_edge(adj\_list, 0, 4);

add\_edge(adj\_list, 0, 1);

add\_edge(adj\_list, 1, 5);

add\_edge(adj\_list, 1, 4);

add\_edge(adj\_list, 1, 3);

add\_edge(adj\_list, 2, 2);

add\_edge(adj\_list, 2, 5);

add\_edge(adj\_list, 5, 1);

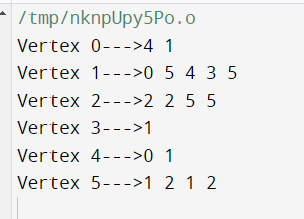
add\_edge(adj\_list, 5, 2);

displayAdjList(adj\_list, v);

return 0;

}

**Output :**

****

**Conclusion : learned Graph structure using adjacency list.**