Data Structures

Finding Degree of each vertex of a Graph

Adjacency Matrix

Degree of a vertex of undirected graph:

Degree of a Vertex: Total no of edges connected to a vertex is called as Degree of That vertex.

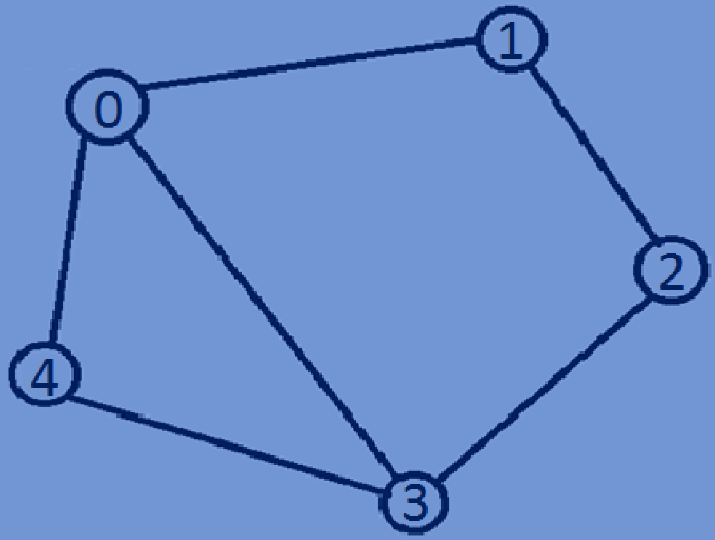
Degree(0)=3

Degree(1)=2

Degree(2)=2

Degree(3)=3

Degree(4)=2



$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 \\ 3 & 1 & 0 & 1 & 0 & 1 \\ 4 & 1 & 0 & 0 & 1 & 0 \end{array}$$

- In case of an undirected graph,the degree of a vertex is equal to the total no of Integers that are greater than zero in the vertex row of an adjacent matrix.

Function for finding the degree of each vertex in an Undirected graph:

```c
void findDegreeUndirectedgraph(graph *array,int n){
    int count=0,i,j;
    for (i = 0; i <  n; i++) {
        count=0;
        for (j = 0; j < n; j++) {
            if(*(array + i*n + j)>0)
                count++;
        }
        printf("degree of vertex %d is ->%d\n",i,count);
    }
}
```

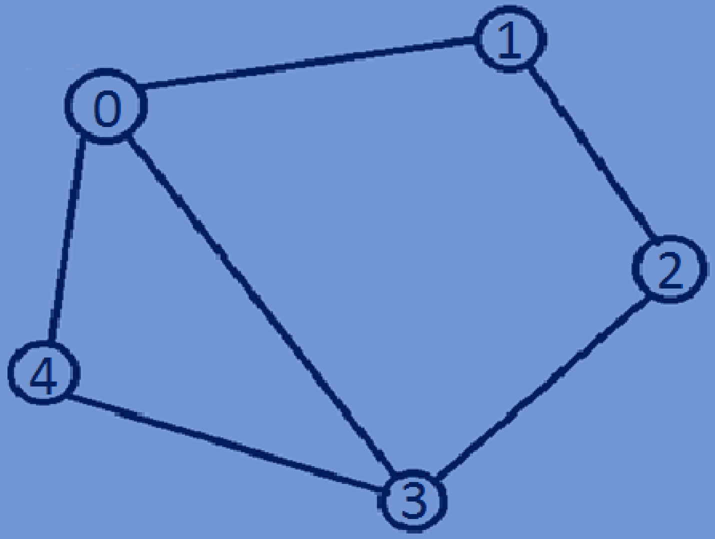
Degree(0)=3

Degree(1)=2

Degree(2)=2

Degree(3)=3

Degree(4)=2



	0	1	2	3	4
0	0	1	0	1	1
1	1	0	1	0	0
2	0	1	0	1	0
3	1	0	1	0	1
4	0	0	0	1	0

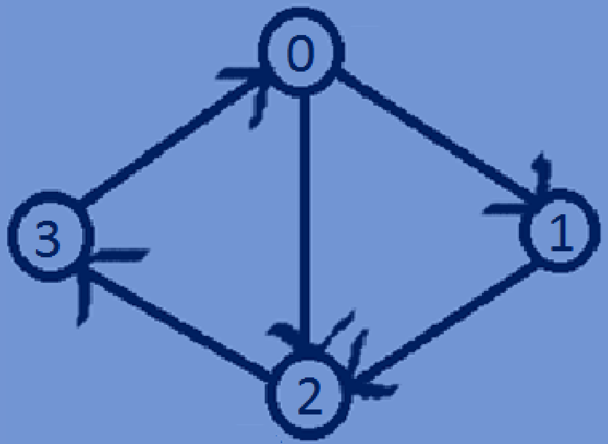
Finding indegree and outdegree of a directed graph:

- For a directed graph each vertex has indegree and outdegree.

Indegree: The total no of incoming edges of a vertex is called as indegree of the vertex.

Outdegree: The total no of Outgoing edges of a vertex is called as outdegree of the vertex.

Indegree(0)=1,Outdegree(0)=2
Indegree(1)=1,Outdegree(1)=1
Indegree(2)=2,Outdegree(2)=1
Indegree(3)=1,Outdegree(3)=1



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 & 1 \\ 3 & 1 & 0 & 0 & 0 \end{array}$$

- In case of directed graph indegree of a vertex is equal to the total no of integers Greater than zero in the vertex column and outdegree of a vertex is equal to the total No of integers greater than zero in the vertex row of Adjacency matrix.

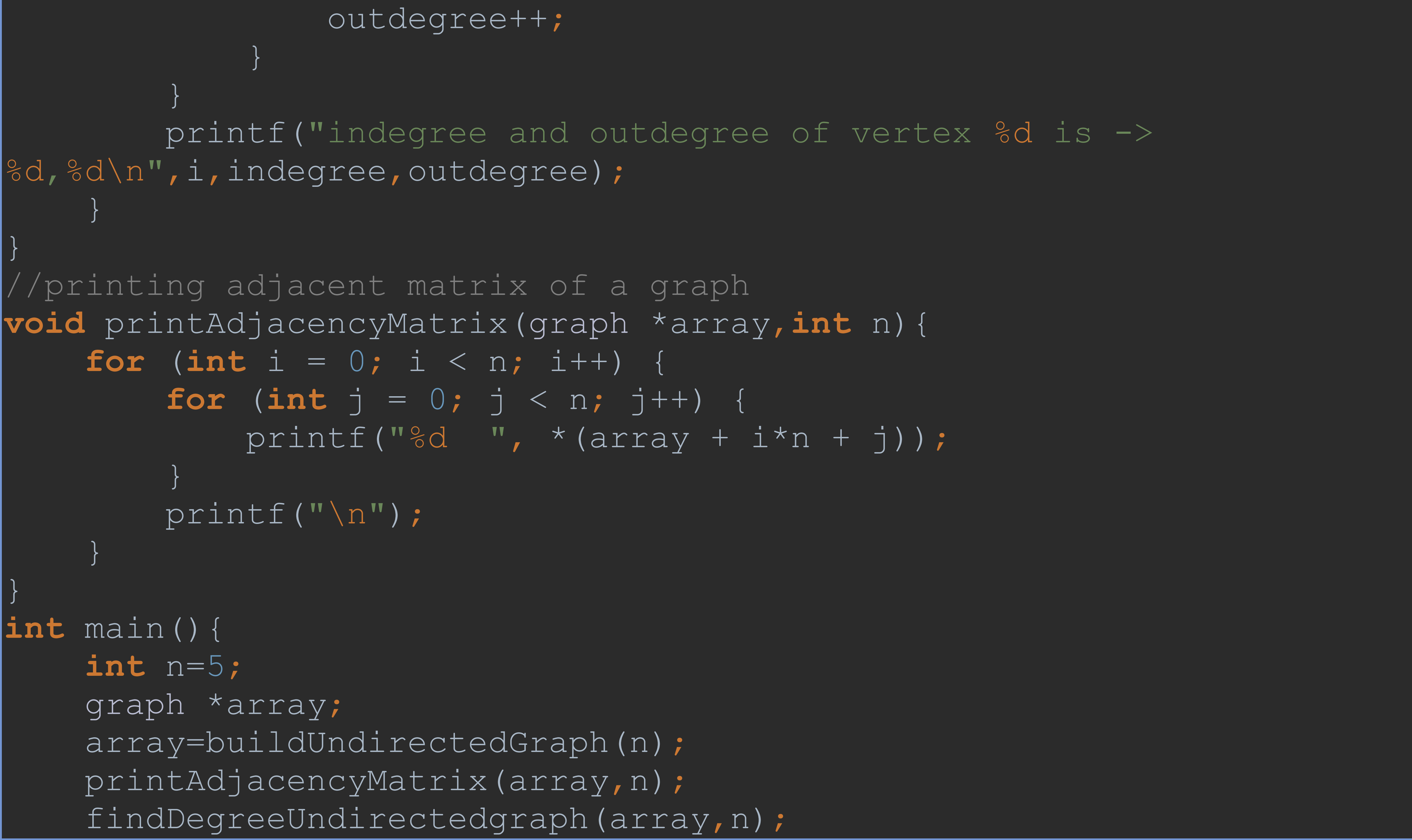
Function for finding indegree and outdegree of a directed graph using Adjacency matrix:

```c
void findDegreeDirectedgraph(graph *array,int n){
    int indegree=0,outdegree=0;
    for(int i=0;i < n;i++){
        indegree=0;outdegree=0;
        for(int j=0;j < n;j++){
            if(*(array + i +j*n)>0){
                indegree++;
            }
            if(*(array + i*n +j)>0){
                outdegree++;
            }
        }
        printf("indegree and outdegree of vertex %d is ->
%d,%d\n",i,indegree,outdegree);
    }
}
```

Whole program:

```c
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
typedef int graph;
//constructing a weighted undirectedgraph
graph *buildUndirectedGraph (int n) {
    int i,j;
    graph *array = (graph *) malloc(n * n * sizeof(graph));
    srand((unsigned)time(NULL));
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i == j) {
                *(array + i * n + j) = 0;
            } else if (i != j) {
                int temp=rand()%2;
                *(array + i * n + j) =temp;
                *(array + j * n + i) =temp;
            }
            }
        }
    return array;
}
//constructing a weighted directed graph
graph *buildDirectedGraph(int n){
    int i,j;
    graph *array = (graph *) malloc(n * n * sizeof(graph));
    srand((unsigned)time(NULL));
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (i == j) {
                *(array + i * n + j) = 0;
            } else if (i != j) {
                int temp=rand()%2;
                *(array + i * n + j) =temp;
            }
        }
    }
    return array;
}
//Finding indegree and outdegree of a directed graph
```

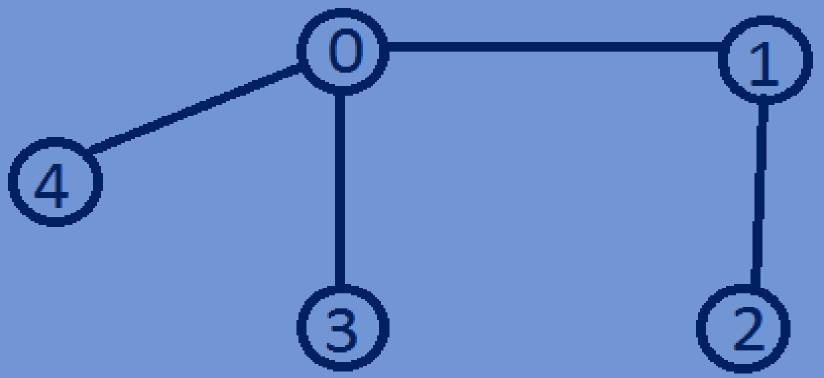
```c
void findDegreeUndirectedgraph(graph *array,int n){
    int count=0,i,j;
    for (i = 0; i <  n; i++) {
        count=0;
        for (j = 0; j < n; j++) {
            if(*(array + i*n + j)>0)
                count++;
        }
        printf("degree of vertex %d is ->%d\n",i,count);
    }
}
//Function for finding indegree and outdegree of a directed graph
using Adjacent matrix
void findDegreeDirectedgraph(graph *array,int n){
    int indegree=0,outdegree=0;
    for(int i=0;i < n;i++){
        indegree=0;outdegree=0;
        for(int j=0;j < n;j++){
            if(*(array + i +j*n)>0){
                indegree++;
            }
            if(*(array + i*n +j)>0){
```



```
                outdegree++;
            }
        }
        printf("indegree and outdegree of vertex %d is ->
%d,%d\n",i,indegree,outdegree);
    }
}
//printing adjacent matrix of a graph
void printAdjacencyMatrix(graph *array,int n){
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d  ", *(array + i*n + j));
        }
        printf("\n");
    }
}
int main(){
    int n=5;
    graph *array;
    array=buildUndirectedGraph(n);
    printAdjacencyMatrix(array,n);
    findDegreeUndirectedgraph(array,n);
    array=buildDirectedGraph(n);printf("\n\n");
    printAdjacencyMatrix(array,n);
    findDegreeDirectedgraph(array,n);
    return 0;
}
```

Output:

```
0 1 0 1 1
1 0 1 0 0
0 1 0 0 0
1 0 0 0 0
1 0 0 0 0
degree of vertex 0 is ->3
degree of vertex 1 is ->2
degree of vertex 2 is ->1
degree of vertex 3 is ->1
degree of vertex 4 is ->1
```



```
0  0  1  0  0
0  0  0  1  0
0  1  0  0  0
0  0  0  0  1
0  0  0  0  0
indegree and outdegree of vertex 0 is -> 0,1
indegree and outdegree of vertex 1 is -> 1,1
indegree and outdegree of vertex 2 is -> 1,1
indegree and outdegree of vertex 3 is -> 1,1
indegree and outdegree of vertex 4 is -> 1,0
```