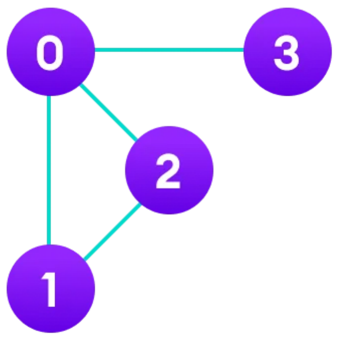
Adjacency Matrix

In this tutorial, you will learn what an adjacency matrix is. Also, you will find working examples of adjacency matrix in C, C++, Java and Python.

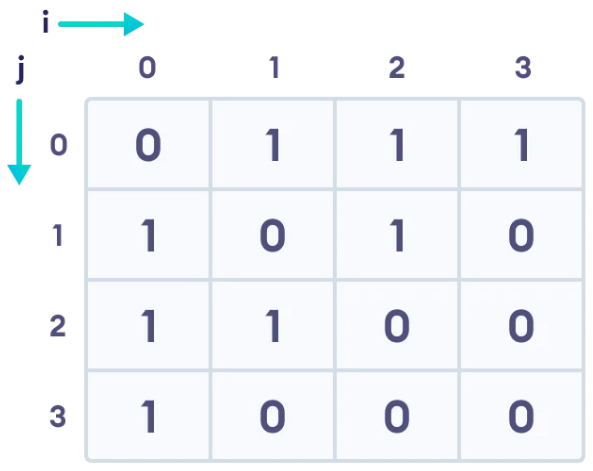
An adjacency matrix is a way of representing a graph as a matrix of booleans (0's and 1's). A finite graph can be represented in the form of a square matrix on a computer, where the boolean value of the matrix indicates if there is a direct path between two vertices.

For example, we have a graph below.



An undirected graph

We can represent this graph in matrix form like below.



$$\begin{array}{c|cccc} j \downarrow \; i \rightarrow & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \end{array}$$

Matrix representation of the graph

Each cell in the above table/matrix is represented as A_{ij}, where `i` and `j` are vertices. The value of A_{ij} is either 1 or 0 depending on whether there is an edge from vertex `i` to vertex `j`.

If there is a path from `i` to `j`, then the value of A_{ij} is 1 otherwise its 0. For instance, there is a path from vertex 1 to vertex 2, so A_{12} is 1 and there is no path from vertex 1 to 3, so A_{13} is 0.

In case of undirected graphs, the matrix is symmetric about the diagonal because of every edge `(i,j)`, there is also an edge `(j,i)`.

Pros of Adjacency Matrix

- The basic operations like adding an edge, removing an edge, and checking whether there is an edge from vertex i to vertex j are extremely time efficient, constant time operations.
- If the graph is dense and the number of edges is large, an adjacency matrix should be the first choice. Even if the graph and the adjacency matrix is sparse, we can represent it using data structures for sparse matrices.
- The biggest advantage, however, comes from the use of matrices. The recent advances in hardware enable us to perform even expensive matrix operations on the GPU.

- By performing operations on the adjacent matrix, we can get important insights into the nature of the graph and the relationship between its vertices.

Cons of Adjacency Matrix

- The `VxV` space requirement of the adjacency matrix makes it a memory hog. Graphs out in the wild usually don't have too many connections and this is the major reason why adjacency lists are the better choice for most tasks.
- While basic operations are easy, operations like `inEdges` and `outEdges` are expensive when using the adjacency matrix representation.

Adjacency Matrix Code in Python, Java, and C/C++

If you know how to create two-dimensional arrays, you also know how to create an adjacency matrix.

Python Java C C++

```
// Adjacency Matrix representation in C

#include <stdio.h>
#define V 4

// Initialize the matrix to zero
void init(int arr[][V]) {
  int i, j;
  for (i = 0; i < V; i++)
    for (j = 0; j < V; j++)
      arr[i][j] = 0;
}

// Add edges
void addEdge(int arr[][V], int i, int j) {
  arr[i][j] = 1;
  arr[j][i] = 1;
}

// Print the matrix
void printAdjMatrix(int arr[][V]) {
  int i, j;

  for (i = 0; i < V; i++) {
    printf("%d: ", i);
    for (j = 0; j < V; j++) {
      printf("%d ", arr[i][j]);
    }
```

Adjacency Matrix Applications

- Creating routing table in networks
- Navigation tasks