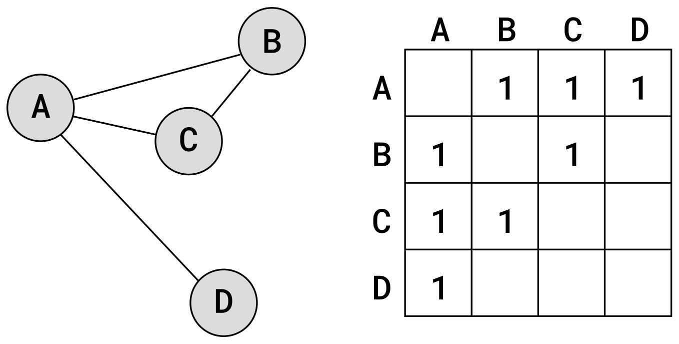
DSA Graphs Implementation

A Basic Graph Implementation

Before we can run algorithms on a Graph, we must first implement it somehow.

To implement a Graph we will use an Adjacency Matrix, like the one below.



	A	B	C	D
A		1	1	1
B	1		1	
C	1	1		
D	1			

An undirected Graph and its adjacency matrix

To store data for each vertex, in this case the letters A, B, C, and D, the data is put in a separate array that matches the indexes in the adjacency matrix, like this:

For an undirected and not weighted Graph, like in the image above, an edge between vertices `i` and `j` is stored with value `1`. It is stored as `1` on both places `(j,i)` and `(i,j)` because the edge goes in both directions. As you can see, the matrix becomes diagonally symmetric for such undirected Graphs.

Let's look at something more specific. In the adjacency matrix above, vertex A is on index `0`, and vertex D is on index `3`, so we get the edge between A and D stored as value `1` in position `(0,3)` and `(3,0)`, because the edge goes in both directions.

Below is a basic implementation of the undirected Graph from the image above.

Example

Python:

```python
vertexData = ['A', 'B', 'C', 'D']

adjacency_matrix = [
    [0, 1, 1, 1],  # Edges for A
    [1, 0, 1, 0],  # Edges for B
    [1, 1, 0, 0],  # Edges for C
    [1, 0, 0, 0]   # Edges for D
]

def print_adjacency_matrix(matrix):
    print("\nAdjacency Matrix:")
    for row in matrix:
        print(row)

print('vertexData:',vertexData)
print_adjacency_matrix(adjacency_matrix)
```

Try it Yourself »

Example

Python:

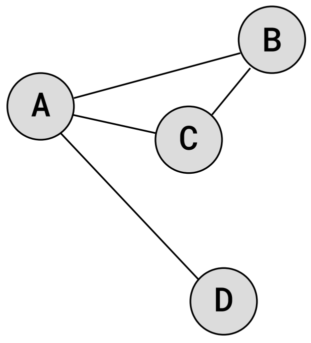
```python
def print_connections(matrix, vertices):
    print("\nConnections for each vertex:")
    for i in range(len(vertices)):
        print(f"{vertices[i]}: ", end="")
        for j in range(len(vertices)):
            if matrix[i][j]:  # if there is a connection
                print(vertices[j], end=" ")
        print()  # new line
```

Try it Yourself »

Graph Implementation Using Classes

A more proper way to store a Graph is to add an abstraction layer using classes so that a Graph's vertices, edges, and relevant methods, like algorithms that we will implement later, are contained in one place.

Programming languages with built-in object-oriented functionality like Python and Java, make implementation of Graphs using classes much easier than languages like C, without this built-in functionality.



	A	B	C	D
A		1	1	1
B	1		1	
C	1	1		
D	1			

An undirected Graph and its adjacency matrix

Python:

```
class Graph:
    def __init__(self, size):
        self.adj_matrix = [[0] * size for _ in range(size)]
        self.size = size
        self.vertex_data = [''] * size

    def add_edge(self, u, v):
        if 0 <= u < self.size and 0 <= v < self.size:
```

```

    def add_vertex_data(self, vertex, data):
        if 0 <= vertex < self.size:
            self.vertex_data[vertex] = data

    def print_graph(self):
        print("Adjacency Matrix:")
        for row in self.adj_matrix:
            print(' '.join(map(str, row)))
        print("\nVertex Data:")
        for vertex, data in enumerate(self.vertex_data):
            print(f"Vertex {vertex}: {data}")

g = Graph(4)
g.add_vertex_data(0, 'A')
g.add_vertex_data(1, 'B')
g.add_vertex_data(2, 'C')
g.add_vertex_data(3, 'D')
```

```

g.print_graph()
```

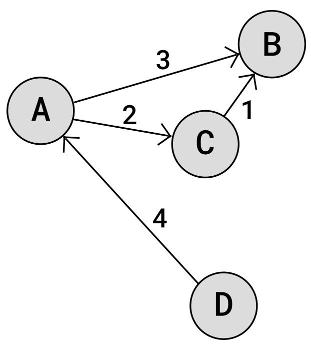
In the code above, the matrix symmetry we get for undirected Graphs is provided for on line 9 and 10, and this saves us some code when initializing the edges in the Graph on lines 29-32.

Implementation of Directed and Weighted Graphs

To implement a Graph that is directed and weighted, we just need to do a few changes to previous implementation of the undirected Graph.

To create directed Graphs, we just need to remove line 10 in the previous example code, so that the matrix is not automatically symmetric anymore.

The second change we need to do is to add a `weight` argument to the `add_edge()` method, so that instead of just having value `1` to indicate that there is an edge between two vertices, we use the actual weight value to define the edge.



	A	B	C	D
A		3	2	
B				
C		1		
D	4			

A directed and weighted Graph, and its adjacency matrix.

Below is the implementation of the directed and weighted Graph above.

Example

Python:

```
class Graph:
    def __init__(self, size):
        self.adj_matrix = [[None] * size for _ in range(size)]
```

```
8          if 0 <= u < self.size and 0 <= v < self.size:
9              self.adj_matrix[u][v] = weight

11
12     def add_vertex_data(self, vertex, data):
13         if 0 <= vertex < self.size:
14             self.vertex_data[vertex] = data
15
16     def print_graph(self):
17         print("Adjacency Matrix:")
18         for row in self.adj_matrix:
19             print(' '.join(map(lambda x: str(x) if x is not None el
20         print("\nVertex Data:")
21         for vertex, data in enumerate(self.vertex_data):
22             print(f"Vertex {vertex}: {data}")
23
24 g = Graph(4)
25 g.add_vertex_data(0, 'A')
26 g.add_vertex_data(1, 'B')
27 g.add_vertex_data(2, 'C')
28 g.add_vertex_data(3, 'D')
29 g.add_edge(0, 1, 3)  # A -> B with weight 3
30 g.add_edge(0, 2, 2)  # A -> C with weight 2
31 g.add_edge(3, 0, 4)  # D -> A with weight 4
32 g.add_edge(2, 1, 1)  # C -> B with weight 1
33
34 g.print_graph()
```

Try it Yourself »

Line 3: All edges are set to `None` initially.

Line 7: The weight can now be added to an edge with the additional `weight` argument.

Line 10: By removing line 10, the Graph can now be set up as being directed.

On the next page we will see how Graphs can be traversed, and on the next pages after that we will look at different algorithms that can run on the Graph data structure.

Test Yourself With Exercises

Exercise:

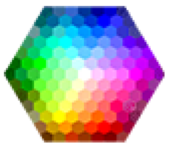
How are the edges in a graph implemented?

```
The edges, and edge weights,
in a graph are normally
implemented in an           matrix.
```

Submit Answer »

Start the Exercise

COLOR PICKER



Tutorials References Exercises Sign In

SS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C

PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
AngularJS Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate

FORUM ABOUT ACADEMY