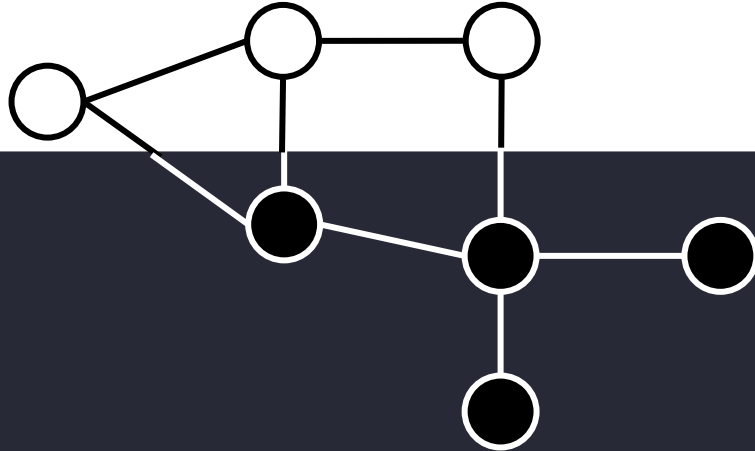


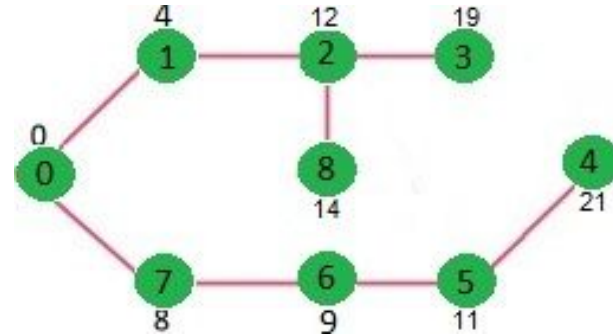
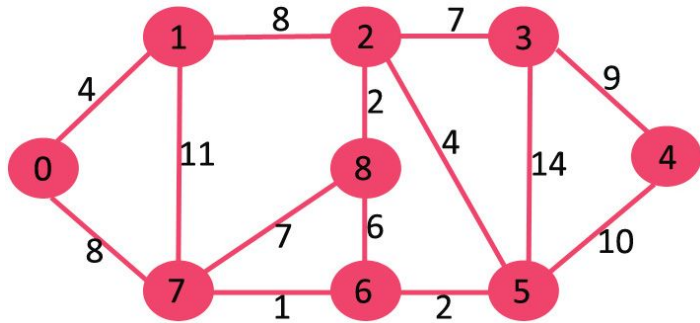
# Dijkstra's algorithm

The quest of the shortest path.

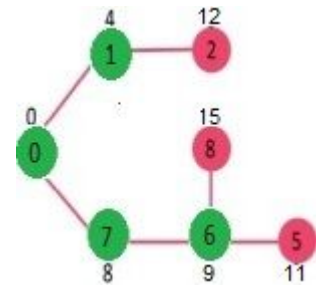
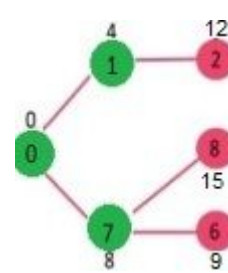
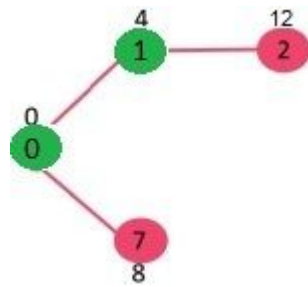
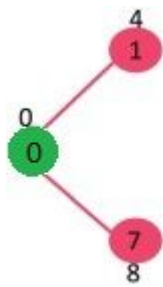


# What is Dijkstra's algorithm

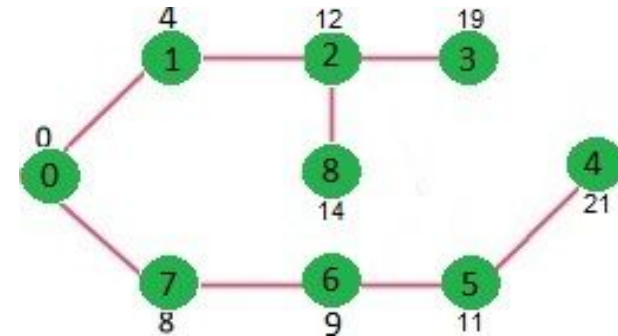
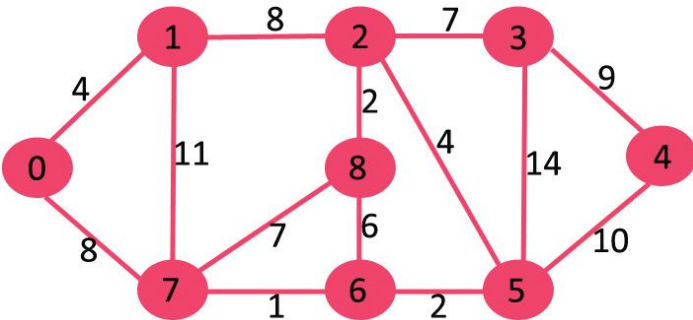
- A greedy algorithm
- Find the shortest path between two nodes or between one fixed node and all the other nodes.
- Works for positively weighted graphs.



source: GeeksForGeeks



- 1) Select the node with shortest distance that has not been 'approved' (bool = false)
- 2) Approve that node (to exclude in next iteration)
- 3) Update distance of neighbors in the distance array
- 4) Repeat until all nodes have been 'approved'/'visited'.



## GraphLibrary


> bin


> obj

 .gitignore

 Edge.cs

 Graph.cs

 GraphLibrary.csproj

 TVertex.cs

 Vertex.cs

I decided to reuse the GraphLibrary created in class and add a method in graph.

I added a non-static method because it makes sense for me to call it on an instance of my Graph class.

Only change is the Edge properties with the addition of uint weight.

```
public uint Weight;
```

```

// create dictionary to store Vertex-bool
Dictionary<Vertex<TVertex>, bool> boolDic = new Dictionary<Vertex<TVertex>, bool>();
for (int i = 0; i < _nVertices; i++) {
    boolDic.Add(_vertices.ElementAt(i), false);
}

// create dictionary to store Vertex-int (dist)
Dictionary<Vertex<TVertex>, uint> distDic = new Dictionary<Vertex<TVertex>, uint>();
for (int i = 0; i < _nVertices; i++) {
    distDic.Add(_vertices.ElementAt(i), int.MaxValue);
}

// set the distance of startVertex to itself to 0
distDic[startVertex] = 0;

// Create a Parent array of Vertices that store an ID
Dictionary<Vertex<TVertex>, Vertex<TVertex>?> parentDic =
new Dictionary<Vertex<TVertex>, Vertex<TVertex>?>();
for (int i = 0; i < _nVertices; i++) {
    parentDic.Add(_vertices.ElementAt(i), null);
}

```

Use of Dictionary instead of array

Update the distance Dictionary to 0 for the starting point.

Create a Dictionary to return a Path

**For the rest let's check the code.**

# Challenges



Adapt my code to the existing library

Understand the algorithm logic

Taming my ego (2 hours vs 20 min)

Using the right Type/collection

# Reflection



## Learning:

- Type / Collection / Generics
- Big-O Notation

## Adapt my code to the existing library

- Could be improved with fibonacci heap and priority queue