Graph Representation

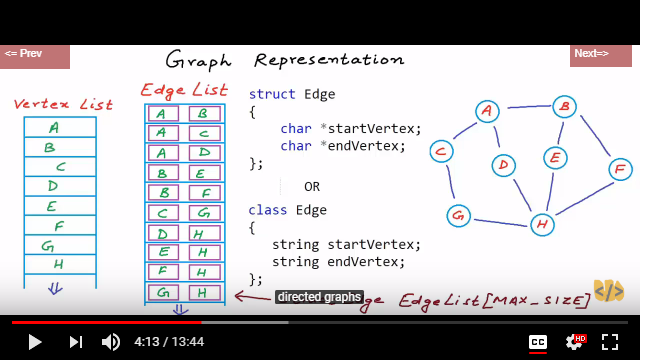
\*The Simplest way to represent a graph would be, we can create two lists one for vertices and another for edges.

For list we can use array of an appropriate size or we can use an implementation of a dynamic list like vector in c++ or ArrayList in Java.

\*Vertex list= A vertex is identified by its name for list of vertices will be list of names or string.

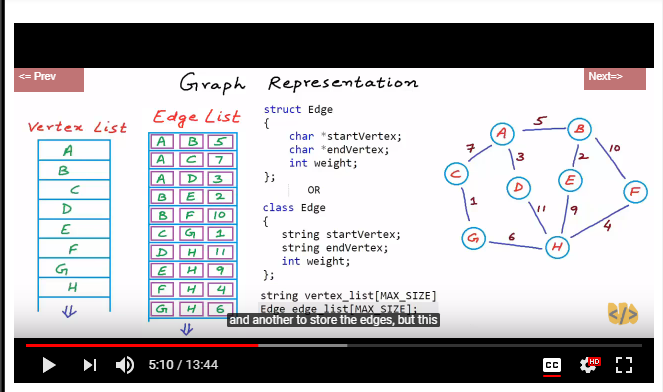
\*Edge List= An edge is identified by its two end points so we can create edge as an object with two fields, one to store start and another to store end of the edge.

Look at the image here



If the graph is weighted then we can have one more field in the edge object. Now an entry in edge list will have three fields. One to store the start vertex, one to store end vertex and one to store weight of the edge.

Look at the image here



\*Let’s discuss the structure in terms of memory and space required for representation.

->Space complexity

For vertex list space complexity will be O(|V|) where |V| number of vertices in list V and for Edge list it will O(|E|).

So total space complexity will be O(|V|+|E|)

->Time complexity

One of the most frequently performed operations while working with graph would be finding all nodes adjacent to a give node. We have to scan the whole edge list.Time complexity will be O(|E|)

In a graph with simple edge means a graph with no self loop if number of element is |V|=n

If |V| = v

Then 0<=|E|<=v(v-1) if directed

0<=|E|<=v(v-1)/2 if undirected

\*So in graph any operation running in order or number of edges would be considered very costly. We try to keep things in order of number of vertices.

\*Representing Graph as vertex list and edge list is not very time efficient design