## Graph Algorithm

Graph Algorithm Outline

1. Discuss how to represent one graph, and discuss *BFS and DFS algorithm*.
2. Discuss how to calculate the *Minimum Generated Tree of Graph*.
3. Discuss how to calculate the *Shortest Path of Two Nodes within Weight Graph*.
4. Discuss how to calculate the *Maximum Flow in Flow Network*.

*Representation in Graph Algorithm*

* For the given graph *G = ( V, E )*, when represent run time of the algorithm, normally we use the Node Number | V | of Graph and the Edge Number | E | as input, which is to say we use two parameters but not one parameter to describe the scale of input.
* Graph Search Skill is the Core in the whole Graph Algorithm.

*Representation of Graph*

For Graph = (V, E), there have two standard method to represent graph.

1. *Adjacent List Method*
2. *Adjacent Matrix Method*

These two method all can be used to present *Undirected Graph*, and *Directed Graph*. Most of algorithms use the Adjacent Matrix Method as input, but when we need to check if there has edge between two nodes, we may need to use Adjacent Matrix Method.

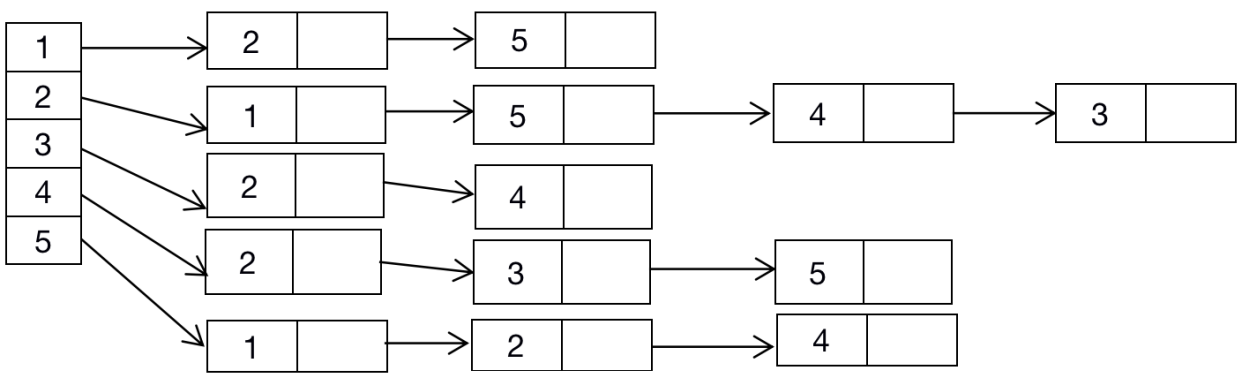
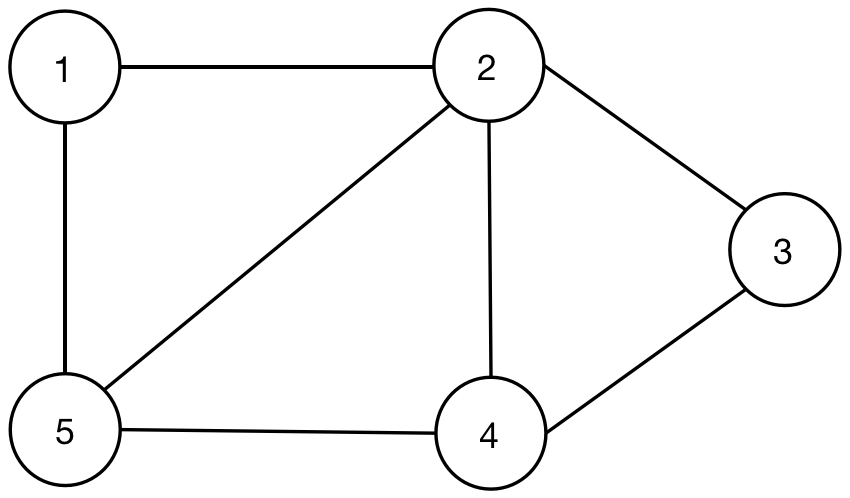
* *Adjacent Linked List Representation Method*

For Graph G = (V, E), Adjacent List Representation consists of array Adj which includes | V | Linked List, each node however has one Linked List.

For each node u belongs to collection V, Adjacent List Adj[u] includes all nodes v which has connected with node u --- Adj[u] includes all nodes v which are adjacent with node u.

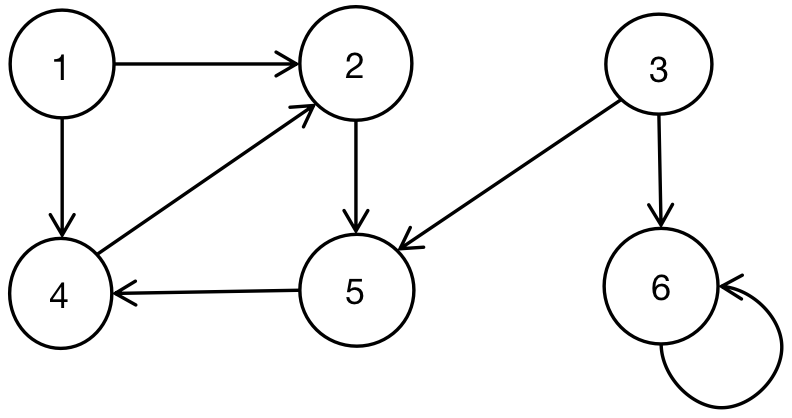
*Undirected Graph*

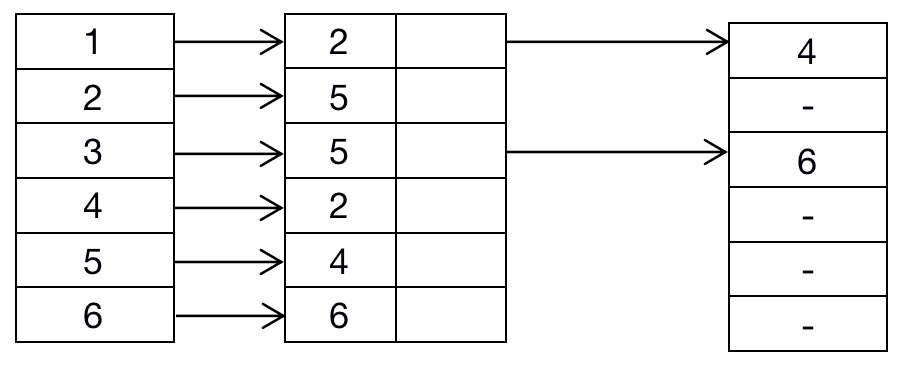
If Graph is *Undirected Graph*, then for edge ( u, v ), the node v appears in Linked List Adj[u], also the node u would appear in Linked List Adj[v]. Therefore, the length sum of all Linked List equals to 2 | E |.



*Directed Graph*

If Graph is *Directed Graph*, then for the edge ( u, v ), the node v appears in the Linked List Adj[u], so the length sum of all Linked List equals to | E |.





But no matter representing by Undirected Graph or Directed Graph, the storage space of Linked List is *O ( V + E )*.

Modify the Linked List for a while, then it can be used to present the *Wight Graph*. The Weight Graph stands for each Edge with weight in Graph. The Weight can be calculated by w: the weight function E -> R. If G = ( V, E ) is one Weight Graph, it’s Weight Function is w, then the weight value can be saved in Adjacent List of node u.

From this point of view, the robustness of Adjacent List is high, and by simple modification, it can be used to support many variants.

*Deficiency of Adjacent Linked List*

The deficiency of Adjacent Linked List is that we can not judge whether there has one edge between two nodes, that is to say, we can not get the edge ( u, v ) is one of edge in the graph. The only method is to search for the node v in Adjacent Linked List Adj[u].

However, Adjacent Matrix Representation can overcome this kind of deficiency, the paid cost is much more storage space.

* *Adjacent Matrix Representation Method*

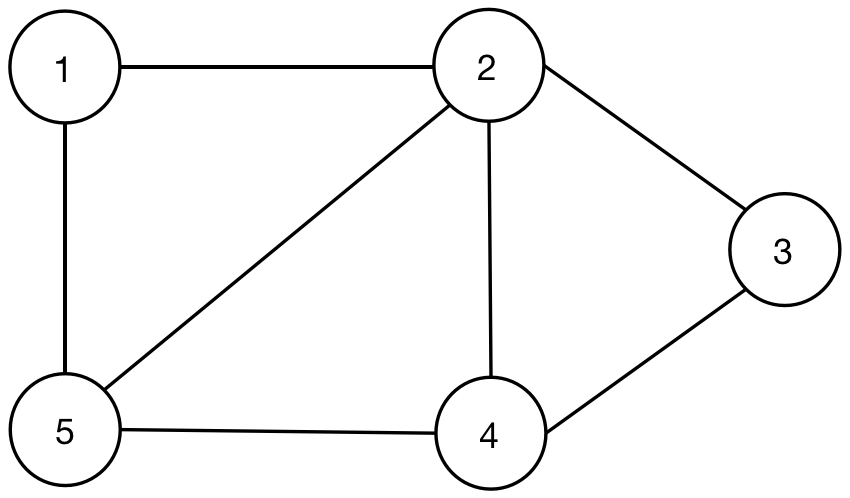
For Adjacent Matrix Representation, we normally encode the node in Graph as 1, 2, 3, ... | V |, these number can be random. After proceeding this kind of encode, the Adjacent Matrix of Graph G consists of one | V | \* | V | Matrix, and this matrix satisfies below condition:

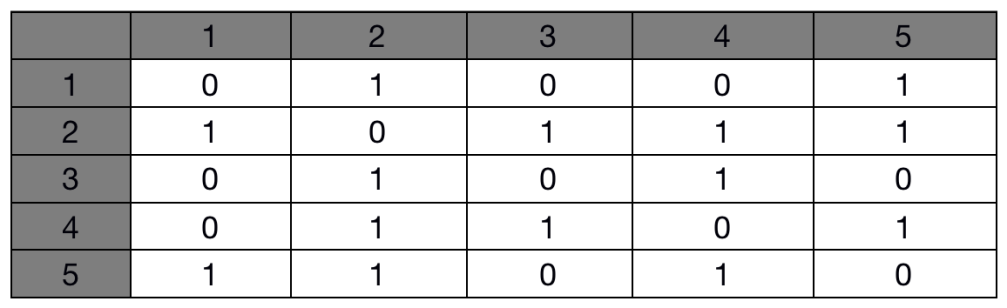
*Aij = 1, if ( i, j ) belongs to E*

*Aij = 0, otherwise*

No matter how many edges in one Graph, then the space of Adjacent Matrix is always O( V ).

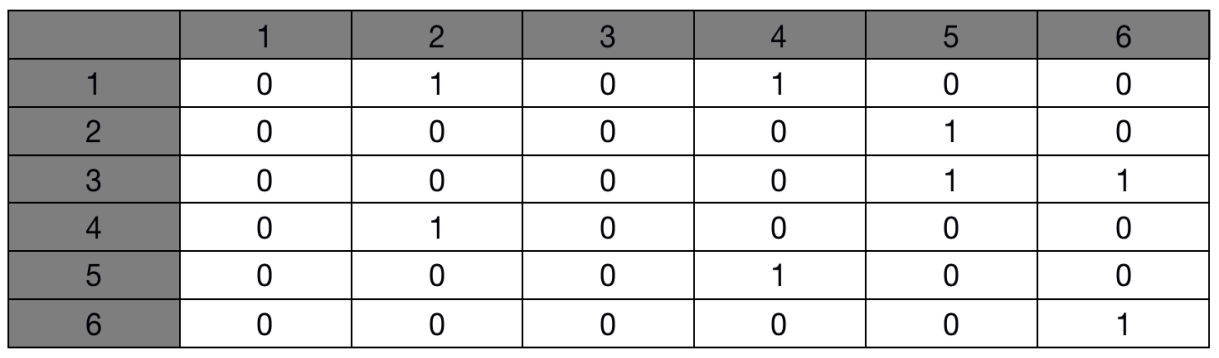
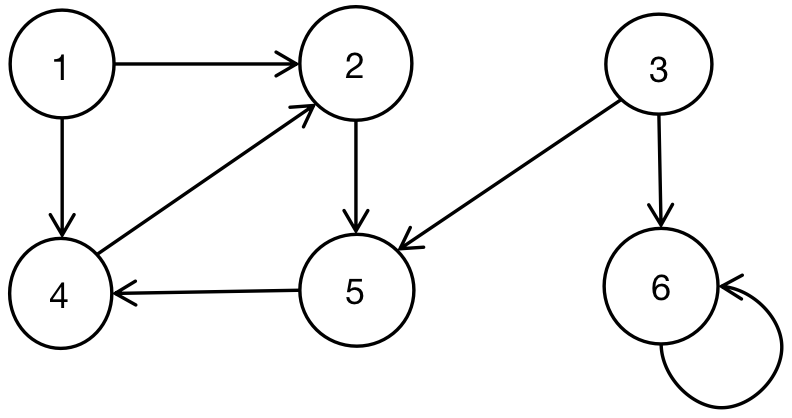
*Undirected Graph*





The Adjacent Matrix of Undirected Graph is the *Symmetric Matrix*, because in the Undirected Graph, the edge ( u, v ) and ( v, u ) is one edge. So we can save half of Adjacent Matrix of Undirected Graph which would be helpful to decrease Graph Storage Space.

*Directed Graph*



Same as Adjacent Linked List Representation, Adjacent Matrix Representation can be used to represent *Weight Graph*.

*For example*

If Graph G = ( V, E ) is the *Weight Graph*, and its weight function is w, then we can directly save *weight w( u, v ) in the uth row vth column* in Adjacent Matrix. For the non - existent edge, we can save NIL directly in the corresponding unit. But for many problems, it is much more convenient to use 0 or infinite to represent the non - existent edge.

There has one more advantage for Adjacent Matrix, which is to say that each record has only one byte space.

*Attribution of Graph*

Most of operations to Graph needs us to maintain some attributions of Graph Node or Edge. These attributions can be presented by v.d, which stands for the attribution d of v. If there has one property of edge f, then the property of edge (u, v) can be represented as (u, v).f, for representing and understanding this algorithm, such property representation is clear.