

### Three Ways to represent a graph:

### We analyze the 3 critiria:

1、Space

2、 how long it takes to determine whether a given edge is in the graph.

3、The other is how long it takes to find the neighbors of a given vertex.

1、Using Array

[ [0,1], [0,6], [0,8], [1,4], [1,6], [1,9], [2,4], [2,6], [3,4], [3,5],

[3,8], [4,5], [4,9], [7,8], [7,9] ]

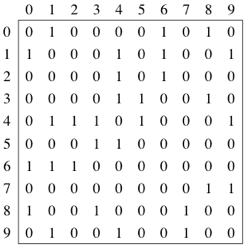
If edges have weights, add either a third element to the array or more information to the object, giving the edge's weight.

1、Total Space: Θ(*E*)

2、If a given edge exists? O(n)

3、

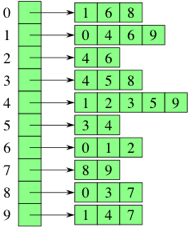
### 2、Adjacency matrice



1. space :Θ(*V*2)space, even if the graph is **sparse**:
2. find out whether an edge is present in constant time
3. if you want to find out which vertices are adjacent to a given vertex i:O(V)

For an undirected graph, the adjacency matrix is **symmetric**

### 3、Adjacency lists



[ [1, 6, 8],

[0, 4, 6, 9],

[4, 6],

[4, 5, 8],

[1, 2, 3, 5, 9],

[3, 4],

[0, 1, 2],

[8, 9],

[0, 3, 7],

[1, 4, 7] ]

We can get to each vertex's adjacency list in constant time, because we just have to index into an array. To find out whether an edge (i,j) is present in the graph, How long does that take in the worst case? The answer is Θ(d), where d is the degree of vertex i, can be from low as 0—high as V-1

If the graph is weighted, then each item in each adjacency list is either a two-item array or an object, giving the vertex number and the edge weight.

How to initial:

for (var j = 0; j < graph[i].length; j++) {

doStuff(graph[i][j]);

}

var vertex = graph[i];

for (var j = 0; j < vertex.length; j++) {

doStuff(vertex[j]);

}

The space:

the adjacency lists for an undirected graph contain 2∣E. Why 2∣E∣? Each edge (i,j) appears exactly twice in the adjacency lists, For a directed graph, the adjacency lists contain a total of ∣E∣,one element per directed edge.