Lists

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
b	a	b	c	a	a	d	g	e	i	g	p	i	i	k	o
f	c	d	g	i	c	h	o	j	f	o			o	h	l
e		f			j	k		n	g					p	
						j		m							

AdjacenyLists are good for their outEdges O (n) time, and they also are very flexible in having many implmentation possibilities

As I understand, the traversal algorithms are more straightforward for list representations as well

Matrix

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
a	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
b	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
c	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
d	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
e	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
f	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
g	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0
h	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
i	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	0
j	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0
k	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
l	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
m	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
n	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
o	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1
p	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

AdjacencyMatrix are great in respect to addEdge, removeEdge and hasEdge taking o(1) time

They also can compute shortest path between all pairs of vertices in O (log n) time