Adjacency Matrix Implementation

graph stores
- a list of vertices
- a list of edges

} **doubly-linked list allows for O(1) removal given reference to list node**

- 2D array, indexed by vertex key

vertex stores
- the associated object
- degree of the vertex
- **reference to the vertex's location in the list of vertices**
- distinct integer key in the range 0..n-1

edge stores
- the associated object
- endpoint vertices
- **reference to the edge's location in the list of edges**

array stores
- A[i][j] holds the edge from vertex with index i to vertex with index j (null if no edge)

Adjacency List Implementation

graph stores
- a list of vertices
- a list of edges

} **doubly-linked list allows for O(1) removal given reference to list node**

vertex stores
- the associated object
- degree of the vertex
- **reference to the vertex's location in the list of vertices**
- list of incident edges

} **doubly-linked list allows for O(1) removal given reference to list node**

edge stores
- the associated object
- endpoint vertices
- **reference to the edge's location in the list of edges**
- **references to the edge's location in the incidence lists for its endpoint vertices**

	adjacency list	adjacency matrix
numVertices(), numEdges()	O(1)	O(1)
vertices(), edges()	O(1) per element	O(1) per element
aVertex()	O(1)	O(1)
degree(v)	O(1)	O(1)
adjacentVertices(v)	O(1) per element	O(n) – to scan row/column of array
incidentEdges(v)	O(1) per element	O(n) – to scan row/column of array
endVertices(e)	O(1)	O(1)
opposite(v,e)	O(1)	O(1)
areAdjacent(v,w)	O(min(deg(v,w))) – search list for vertex with smaller degree	O(1)
insertEdge(v,w,o)	O(1)	O(1)
insertVertex(o)	O(1)	O(n) – to initialize row/col of array $O(n^2)$ – if array needs to grow
removeVertex(v)	O(deg(v)) – to remove each incident edge	O(1) – with clever bookkeeping (and wasted space) $O(n^2)$ – shifting in array
removeEdge(e)	O(1)	O(1)
space	O(n+m)	$O(n^2)$

Comparison

Adjacency matrix –
- **very time-efficient for `isAdjacent` – O(1)**
- very space-inefficient for sparse graphs
- time-inefficient for traversing edges incident on a vertex – O(n)
- time-inefficient for insert/remove vertex

Adjacency list –
- **space-efficient except for the most dense graphs**
- **time-efficient for traversing edges incident on a vertex – O(deg)**
- `isAdjacent` is O(deg) rather than O(1)