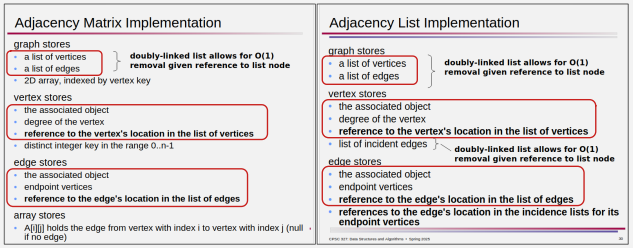
Graph ADT Implementation

- code organization
  - `AbstractGraph`, `AbstractVertex`, and `AbstractEdge` should contain instance variables common to both adjacency list and adjacency matrix implementations



  - `AbstractGraph` should contain bodies for those methods that can be implemented there
  - helper classes such as `ListNode` should be inner classes

Graph ADT Implementation

- running times
  - be sure to achieve the running times discussed in class
    - avoid unnecessary loops, searching

	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)

  - to achieve O(1) removal from a list (and for full credit), use your own doubly-linked list implementation for lists and store the reference to the list node
    - can't just add prev, next pointers to AbstractVertex/AbstractEdge because in the adjacency list implementation, an edge will appear in three lists – the list of edges in the graph and the list of incident edges for each of the edge's vertices
    - no need to search for the list node if there's a reference stored!

Graph ADT Implementation

- Java things
  - `.equals()` vs ==
    - `a == b` is for checking if a and b are the same object (same location in memory)
      - e.g. list nodes
      - use == when *object identity* matters – different objects are different things even if they have the same state
    - `a.equals(b)` is for checking if a and b are equivalent objects (different locations in memory but viewed as interchangeable)
      - e.g. `String`
  - only cast if you need methods of the more specific type