1 Graphs (abstract)

- Represent pieces of information and connections between them
- Verticies (or nodes) are connected with edges
- Edges may have additional information, such as a weight
- Can represent everyday information – road maps, networks, inheritance hierarchy, schedule planning

2 Graphs

- Contain sets of verticies and edges. A graph $G = (V, E)$, where $V$ is the set of verticies and $E$ is the set of edges
- Edges may be bidirectional, in which case the graph is said to be undirected. Otherwise, edges only may be followed in one direction, in which case the graph is said to be directed (a digraph)
- The degree of a node in an undirected graph is the number of edges incident upon it
- In an undirected graph, the in-degree of a node is the number of edges coming into it and the out-degree is the number of edges leaving
- A path is some sequence of edges from one node to another
- A sparse graph is one with much less than $|V|^2$ edges
- A dense graph is one with nearly $|V|^2$ edges

3 Connectivity and Cycles

- An set of verticies in an undirected graph is considered to be connected and a set of verticies in a digraph is considered to be strongly connected if there exists a path from every node to every other node in the set
- If there is only one connected or strongly connected component (set of verticies) that includes all verticies in the graph, then the graph is said to be connected or strongly connected
- If a graph has a cycle (a path from a node to itself), it is said to be cyclic, or otherwise, acyclic
- It follows that connected graphs are cyclic, since if there is a path from every node to every other node, there must be a path from every node to itself
- A tree is a directed acyclic graph (a DAG) with $|E| = |V| - 1$
4 Graph Representation – Adjacency Matrix
- Store information about the edges of a graph in a $|V| \times |V|$ matrix $M$ (array)
- If $G$ is unweighted, store a binary digit to indicate a connection
- If $G$ is weighted with numeric weights, store the numeric weights in the matrix
- Can perform fast lookups to check for existence of edges between two nodes
- Wastes space when representing sparse graphs – good for dense graphs
- $M^n$ produces a matrix representing all paths of length exactly $n$ in $G$

5 Graph Representation – Adjacency Lists
- Store information about the edges incident upon each node in a list
- Takes some time to perform lookups about edges incident upon a node
- Conserves space when representing sparse graphs

6 Graph Representation – Edges Lists
- Store a (unordered) list of edges in $G$
- Is the most space efficient graph representation, but slowest with which to perform any operations

7 Graph Traversals
Clear all node marks
Insert starting node into a (stack, queue)
While the (stack, queue) is not empty

- Remove a node $n$ from the (stack, queue)
- If $n$ is marked, continue the loop
- Mark $n$
- For each edge incident on $n$, add its other incident node to the (stack, queue)

- If we use a stack in the above algorithm, we have a depth first search (DFS) of the graph
- If we use a FIFO queue in the above, we have a breadth first search (BFS)

8 Simple garbage collection
- Java provides garbage collection for allocated memory – that is, when objects are no longer in use, the virtual machine calls the garbage collector to free the memory for future use
- To simply detect which pieces of memory are in use, each allocated piece of memory is a node in the graph, and a graph traversal is run with starting nodes at all declared variables, following all references as edges, marking nodes that are visited
• Once the traversals end, nodes (pieces of memory) that are still unmarked are deallocated

• This is known as mark-and-sweep

• To improve performance, generational mark-and-sweep divides pieces of memory into “generations” – the more times an object survives garbage collection, the older it gets. The older generations have garbage collection run on them less often. The idea behind this is that the longer an object has survived, the longer it is likely to survive.