CSCI 136
Data Structures &
Advanced Programming

Describing Graphs

Describing Graphs

There are many ways to describe a graph G = (V,E)

(other than by drawing a picture)

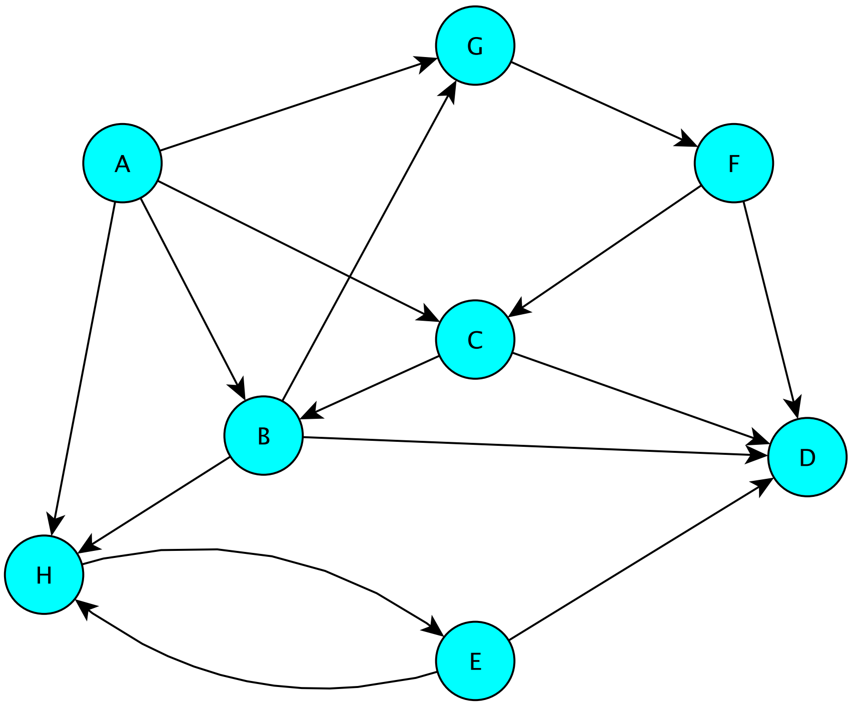
- A list of all vertices followed by a list of all edges
  - Note; If every vertex is incident to at least one edge, the list of vertices can be inferred from the list of edges
- A matrix (2-dimensional array) M such that
  - Each row of M corresponds to a vertex v
  - Each column of M corresponds to an edge e
  - If v is incident with e, entry M[v,e] = 1; else M[v,e] = 0
  - M is called the *incidence matrix* of the graph
  - Note the abuse of notation M[v,e]
    - Neither v nor e might be ints---but we could encode them as ints!

Describing Graphs

The two most frequently used approaches are

- The *Adjacency Matrix* A of the graph G = (V,E)
  - Each row of A corresponds to a vertex of G
  - Each column of A corresponds to a vertex of G
  - A(u,v) = 1 if {u,v} is in E; A(u,v) = 0 otherwise
- The *Adjacency Lists* AL of the graph G = (V,E)
  - AL is a 1-dimensional array indexed by V
  - The entry AL[v] is a list of all neighbors of v
    - If G is directed, AL[v] is a list of all *out-neighbors* of v
- Again: We encode vertices as ints

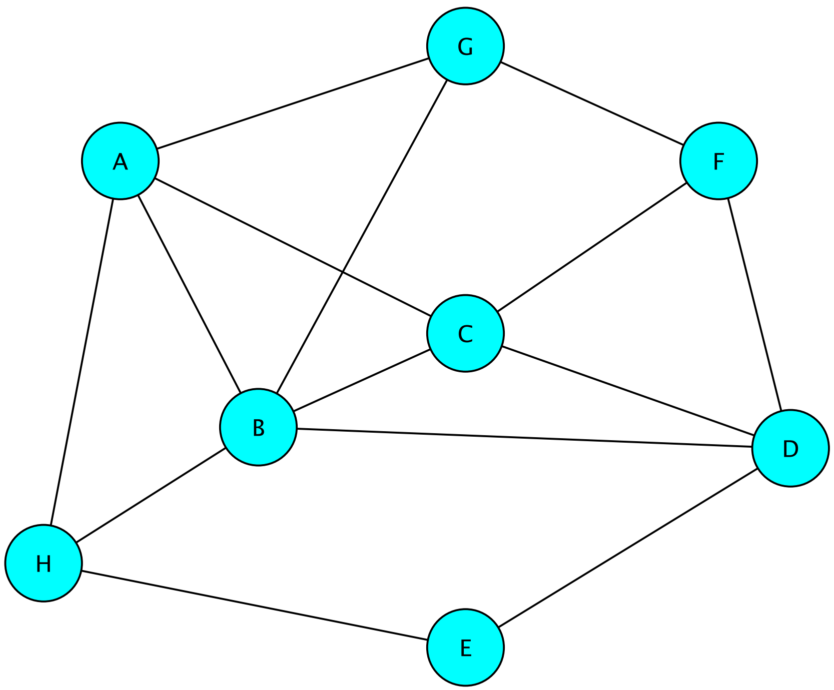
Adjacency Array: Directed Graph



	A	B	C	D	E	F	G	H
A	0	1	1	0	0	0	1	1
B	0	0	0	1	0	0	1	1
C	0	1	0	1	0	0	0	0
D	0	0	0	0	0	0	0	0
E	0	0	0	1	0	0	0	1
F	0	0	1	1	0	0	0	0
G	0	0	0	0	0	1	0	0
H	0	0	0	0	1	0	0	0

Entry (i,j) stores 1 if there is an edge from i to j; 0 otherwise
For example: edges(B,C) = 0 but edges(C,B) = 1

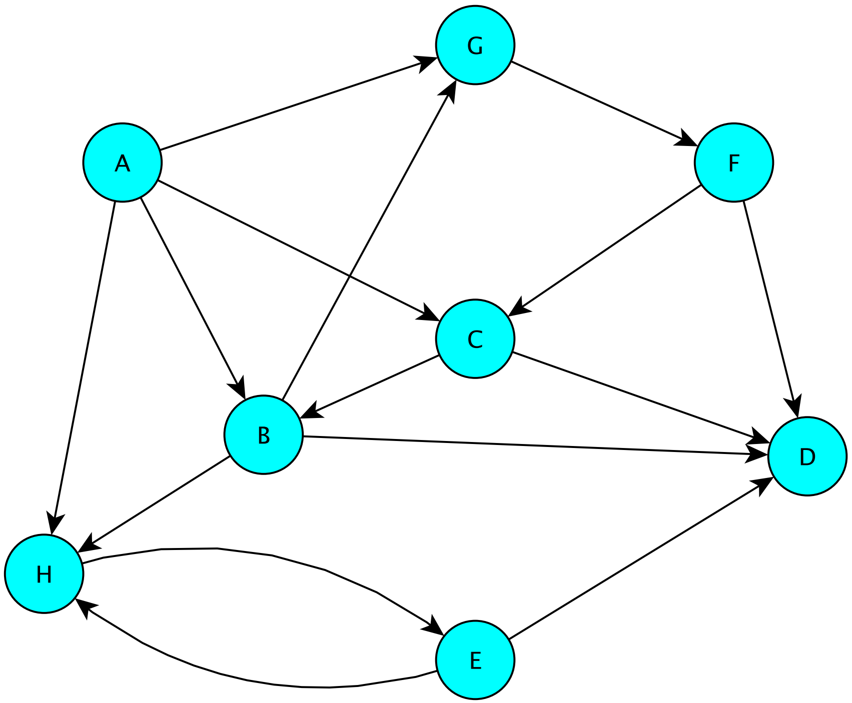
Adjacency Array: Undirected Graph

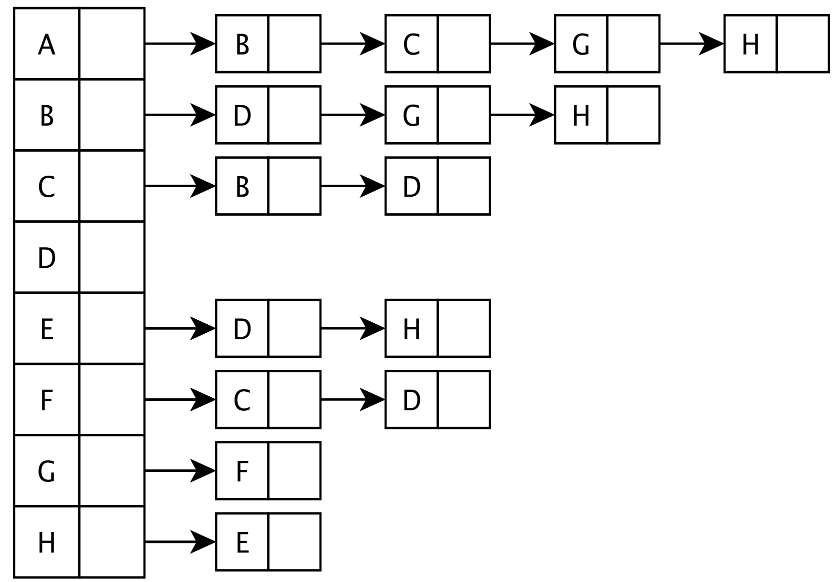


	A	B	C	D	E	F	G	H
A	0	1	1	0	0	0	1	1
B	1	0	1	1	0	0	1	1
C	1	1	0	1	0	1	0	0
D	0	1	1	0	1	1	0	0
E	0	0	0	1	0	0	0	1
F	0	0	1	1	0	0	1	0
G	1	1	0	0	0	1	0	0
H	1	1	0	0	1	0	0	0

Entry (i,j) store 1 if there is an edge between i and j; else 0
For example: edges(B,C) = 1 = edges(C,B)

Adjacency List : Directed Graph

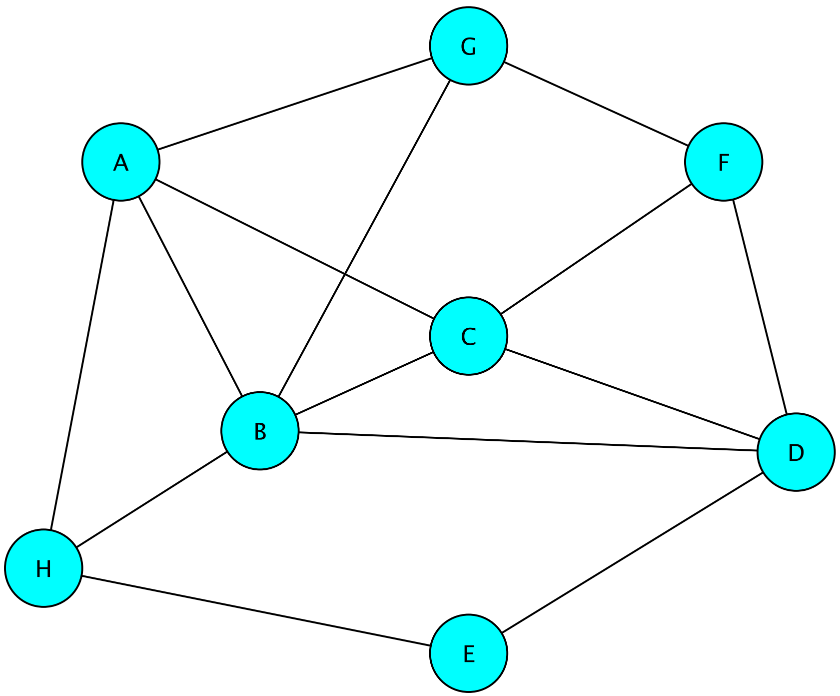


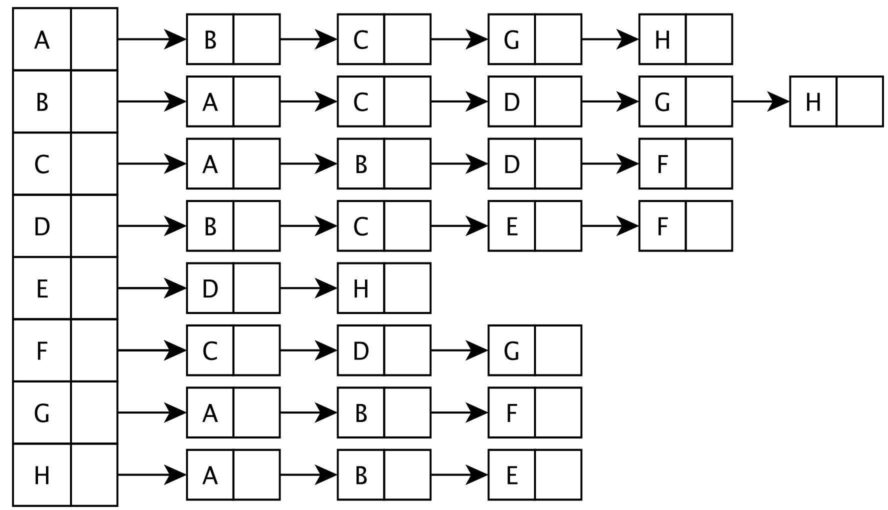


The vertices are stored in an array V[]

V[] contains a linked list of edges having a given source

Adjacency List : Undirected Graph





The vertices are stored in an array V[]

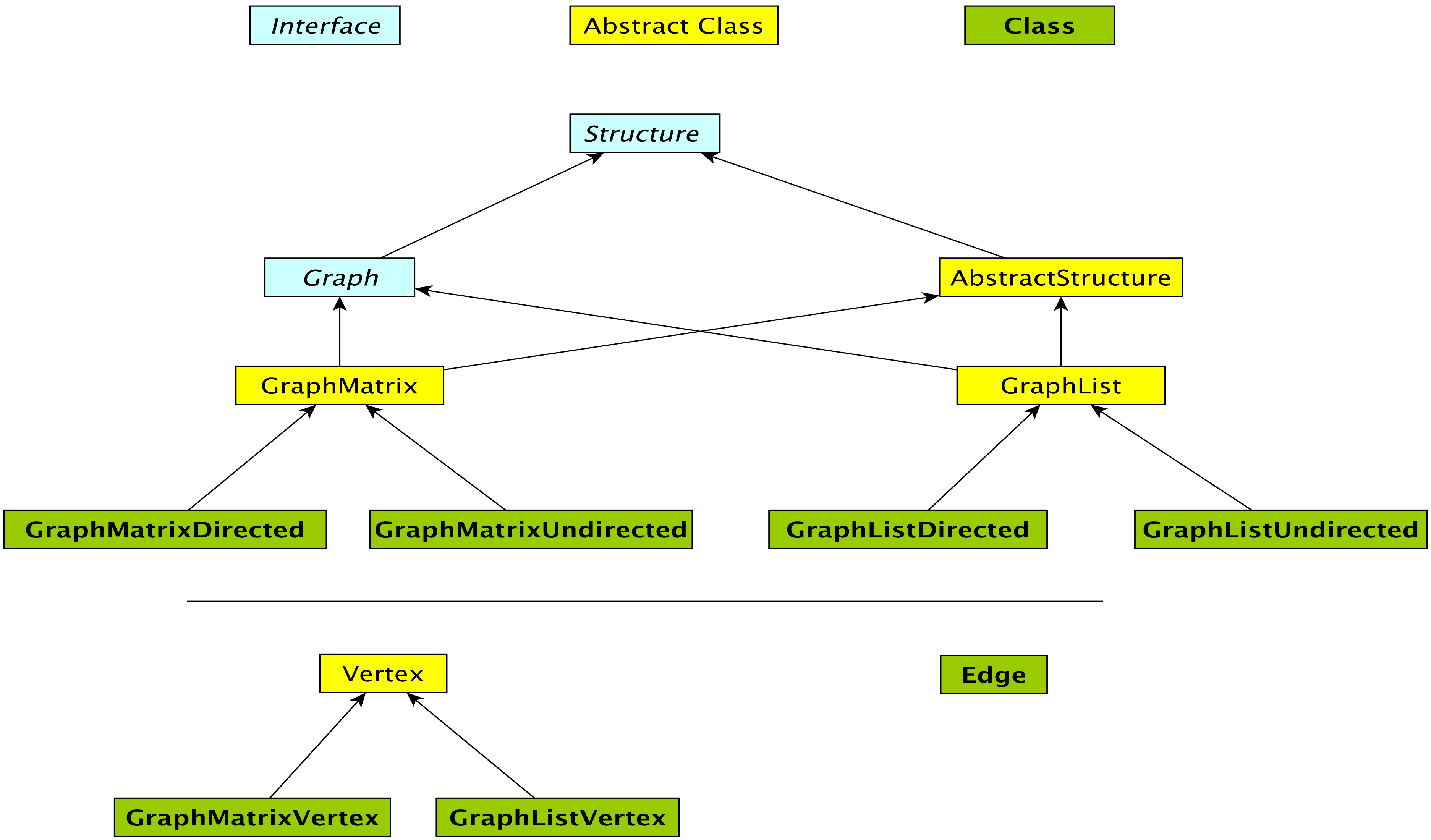
V[] contains a linked list of edges incident to a given vertex

Graph Data Structures

What we want

- Represent both directed and undirected graphs
- Have option of array-based or list-based
    - Lists are more compact for *sparse graphs* (few edges)
- Ability to store application-specific data at vertices and edges
- Most frequently used methods are most efficient
    - Spoiler: Implementations will have different performance characteristics

Graph Classes in structure5



Graph Classes in structure5

Why so many?!

- There are two types of graphs: undirected & directed
- There are two implementations: arrays and lists
- We want to be able to avoid large amounts of identical code in multiple classes
- We abstract out features of implementation common to both directed and undirected graphs

These implementations will be the focus of the next few presentations….